bsc-invariants

An open-source invariant-testing library for the BNB Smart Chain DeFi stack.

Michael Moffett, operator. CaliperForge. 2026-06-27. Contact: michael@caliperforge.com, team@caliperforge.com. Public artifact: `github.com/caliperforge/bsc-invariants` (Apache-2.0). This document: caliperforge.com/bsc-invariants (forthcoming inline + PDF). Companion technical brief: caliperforge.com/methodology (the cross-VM method this paper applies to BSC).

Reading order

---

§1. The problem

§1.1. BSC DeFi is a large, exploited surface

BNB Smart Chain hosts one of the deepest DeFi stacks outside Ethereum mainnet. PancakeSwap is the canonical BSC AMM and the highest-TVL DEX on the chain by a wide margin; Venus is the canonical BSC lending market; Stargate is the canonical cross-chain liquidity-bridge presence on BSC. These three protocols together intermediate billions of dollars of user value across swaps, lending positions, and cross-chain transfers at any given block.

That surface is actively exploited today, not historically. In the last twelve months Venus Protocol (the M2 target of this library) absorbed two distinct on-chain incidents on BNB Chain. In March 2026 a supply-cap-bypass attack on the Venus Core Pool drained roughly $3.6M after the attacker accumulated a large THE-token position over months, transferred tokens directly into the lending contract to bypass the typical deposit flow, then ran a cyclical deposit-borrow-buy strategy that inflated the THE oracle price and let the on-chain collateral position drift to roughly 53.2M THE, about 3.7x the token's actual circulating supply (`https://ambcrypto.com/inside-the-3-6mln-venus-protocol-exploit-on-bnb-chain/` , 2026-03-16). In September 2025 the Venus Core Pool Comptroller contract was updated to a malicious address and the attacker siphoned roughly $27M in vUSDC and vETH from the protocol (`https://www.coindesk.com/tech/2025/09/02/bnb-chain-based-venus-protocol-drained-of-usd27m-on-suspected-contract-compromise` , 2025-09-02). The March 2026 incident sits squarely in the V-1 / V-2 invariant surface this library's M2 milestone targets: an attacker-controlled deposit path that lets the on-chain collateral position drift out of sync with the supply cap and the tracked balance is the planted-twin shape, line for line.

The same window saw the broader DeFi corpus hit again and again at the same invariant classes this library targets. The Cetus CLMM `checked_shlw` arithmetic overflow on Sui (May 2025, ~$223M) shaved a Q64.64 mask wrong and credited huge liquidity for near-zero tokens. The Loopscale lending market on Solana (April 2025, ~$5.8M) priced RateX PT collateral off a single-source on-chain oracle and absorbed manipulation. The zkLend empty-market accumulator inflation on Cairo (February 2025, ~$10M) round-truncated subsequent deposits to zero shares while still crediting the underlying. The New Market Trading confused-deputy access-control failure across 88 Gnosis Safes on Ethereum / Base / Arbitrum (May 2026, ~$3.98M) accepted a caller-supplied delegate address as proof-of-authorization without binding it to `msg.sender`. The JELLY mark-price manipulation on HyperEVM (March 2025) consumed a thin-liquidity HIP-1 perp mark unchecked by a downstream lender and cascaded into unsound liquidations; the CaliperForge `hyperevm-safety` library reproduces it as a clean / planted twin in-tree. Each of these is a 2025 or 2026 incident, not a 2022 retrospective, and each one's bug class maps to a property the BSC stack must assert against on its own surface.

The pattern is not new. The historical record (the Oct 2022 BNB Chain cross-chain bridge incident, the 2021 Venus XVS-collateral liquidation episode, and the long tail of PancakeSwap v2 fork exploits) is preserved in the public post-mortem trackers and informs the `invariant-atlas` corpus alongside the 2025-2026 cases. The point is that the 2025-2026 cadence shows the same bug classes are still being shipped, with material dollar losses, on BSC and on chains adjacent to it. Forks of PancakeSwap v2 and v3 across BSC continue to mis-port the Uniswap-v3 fee-growth accounting math, the canonical bug class that surfaces every time a downstream team refactors `feeGrowthGlobalX128 += FullMath.mulDiv(...)` and accidentally writes `=` instead of `+=`.

A reviewer who has watched BSC DeFi long enough recognizes the shape: high TVL, fast-moving forks of well-known reference implementations, and a recurring exploit class at the pool-accounting, oracle-consumption, and bridge-credit boundaries. The marginal dollar of safety work on BSC pays back in dollars of user value protected when the next mis-ported `+=` ships, the next supply-cap bypass lands, or the next confused-deputy module deploys.

§1.2. Pre-deploy invariant testing is the layer this paper targets

A protocol team has several lines of defense before its code reaches mainnet: code review, unit tests, audits, formal verification of specific modules, and integration testing on a fork. Each catches a different class of bug. The line this paper targets sits inside the protocol's own development loop, before audit, in the form of property tests with stateful fuzzing.

A property test states an invariant the protocol must maintain across any sequence of state-mutating calls and fuzzes the input space against that invariant. When the property is sharp (it actually distinguishes correct code from the specific bug class), and when the harness is wired (the property runs in CI on every push), a refactor that breaks the invariant is caught at PR time. Foundry, Echidna, Medusa, Halmos, and Certora are the engines that run these properties on Solidity. They are mature. They are good.

The differentiated work is not the engine. It is the layer above: which property to assert, on which surface, against which historical failure mode, paired to a same-source twin that demonstrates the property would have caught it. That is the layer `bsc-invariants` ships.

§1.3. Why BSC specifically (not Ethereum mainnet)

Several pre-deploy property-testing libraries exist for Ethereum mainnet protocols. The BSC ecosystem has comparatively few. The PancakeSwap-fork pattern, the Venus / Compound-v2-fork pattern, and the Stargate-on-BSC bridge layer each carry chain-specific surface that an Ethereum-mainnet library does not cover by accident. PancakeSwap v3 ships protocol-fee bps values different from Uniswap v3; Venus's interest-rate model and Comptroller have BSC-specific governance settings; Stargate's pool-delta accounting threads through the LayerZero message protocol with BSC-specific chain-id mappings. A library that targets these surfaces directly removes a class of "we ported the Ethereum library and it does not quite fit" friction.

The BNB Chain 2026 Tech Roadmap (https://www.bnbchain.org/en/blog/tech-roadmap-2026) also names a new execution engine with register-based interpretation and AOT/JIT, EIP-7928 BAL-based parallel execution, a re-architected storage layer, and an AI agent registry with verifiable capabilities. Every one of those priorities introduces new state-transition surface where invariant testing is the natural pre-deploy gate. A library that establishes the BSC-targeting invariant-testing discipline today is the foundation downstream tooling against the 2026 surface will build on.

§2. The approach

§2.1. Planted-twin CLEAN/PLANTED CI, in one paragraph

The unit of work is a pair. A clean reference implementation where the asserted property holds under fuzz, and a same-source twin where one localized hunk introduces the canonical bug-class change the property is designed to catch. CI runs both legs on every push. The clean leg must exit zero with no `INVARIANT VIOLATED` marker on stdout. The planted leg must exit non-zero with the marker present. A property that cannot be falsified by any same-source twin we can construct does not ship. A property that the planted leg does not actually surface does not ship. Both legs are receipts the reviewer can read directly from the CI run, not narrative claims.

This is a falsifiability gate. Every clean claim ships with a planted control that must fire. The CI gate asserts both verdicts explicitly. A green clean leg alone proves only that the asserted predicate survives the harness; it does not prove the predicate is sharp enough to discriminate bug-class code from canonical code. The planted twin is the executable proof of the discrimination. (The full method, including the same-source discipline that prevents "firing for the wrong reason," the audit handle on every artifact, and the catch / false-positive / precision matrix the method imports from the Trace2Inv academic line, is set out in the companion CaliperForge methodology brief: `caliperforge.com/methodology`.)

§2.2. What this brings to BSC

The methodology is general; this paper applies it to BSC's largest DeFi surfaces with three guarantees:

1. **Every invariant lands with a same-source planted twin.** The library does not ship a clean-only suite. The planted-twin discipline applies to PancakeSwap v3, Venus, and Stargate identically.
2. **Receipts, not badges.** Every CI run uploads its `forge test -vv` log; every milestone ships `receipts/forge-test-*.log` plus a `receipts/planted_demo/` directory with the explicit clean-passes + planted-fires pair. A reviewer reads the verdict from the log file directly.

3. **Open-source under Apache-2.0 from commit one.** The library is born public good. Any BNB Chain ecosystem developer can fork, reuse, extend, and adapt the harness to their own protocol without negotiating a license.

The discipline composes with the rest of the Solidity-side stack. The harness runs `forge invariant` under Foundry; the same property surface is portable to Echidna and Medusa via the Recon Chimera bundle (`chimera-template-pack`) once the Recon Chimera leg lands at M2 Week 3; properties that admit symbolic execution can be cross-checked with Halmos. The library is engine-pluralist by design. The differentiation is the property and the twin, not the engine.

§3. What `bsc-invariants` ships today

§3.1. The PancakeSwap v3 invariant surface

Five PancakeSwap v3 invariants are in-tree at HEAD, with a sixth shipping as a paired clean / planted CI demo. Each invariant is a precise mechanical statement on a same-source reference contract; each is exercised by unit tests plus stateful-fuzz invariant tests under Foundry.

ID	Invariant	What it asserts	Test file
P-1	FeeGrowthGlobalMonotonicity	`feeGrowthGlobal{0,1}X128` is non-decreasing across any non-protocol-fee swap with non-zero `feePips × amountIn`, in either direction.	`invariants/PancakeV3FeeGrowth.t.sol`
P-2	TickInBounds	After any state-mutating operation, `MIN_TICK ≤ pool.tick ≤ MAX_TICK` (the canonical Uniswap v3 bound, reproduced from `v3-core/contracts/libraries/TickMath.sol`).	`invariants/PancakeV3TickBounds.t.sol`
P-3	SqrtPriceX96InBounds	After any state-mutating operation, `MIN_SQRT_RATIO ≤ pool.sqrtPriceX96 < MAX_SQRT_RATIO`.	`invariants/PancakeV3TickBounds.t.sol`
P-4	LiquidityEventConsistency	`mint` and `burn` update active `liquidity` by exactly `±delta` for in-range positions; leave it unchanged for out-of-range positions.	`invariants/PancakeV3LiquidityEvents.t.sol`
P-5	FeeGrowthOutsideConsistency	Per-initialized-tick `feeGrowthOutside{0,1}X128 ≤ feeGrowthGlobal{0,1}X128` after any sequence of accrues and crosses (increment-only conservation form; the wrap-around variant is M2 Week 3 work).	`invariants/PancakeV3FeeGrowthOutside.t.sol`
P-1 pla	Clean + plante	The clean leg exits zero with no marker; the planted leg deterministically surfaces `INVARIANT VIOLATED feeGrowth_neverDecreases` and exits non-zero. The planted hunk replaces the canonical `+=` accumulator update	`invariants/planted/PancakeV3Fe`

ID	Invariant	What it asserts	Test file
nted	d CI pair	with a `=` assignment on the `zeroForOne` branch, the exact mis-port a Uniswap-v3 fork introduces when refactoring the pool.	`eGrowth.planted.t.sol`

The full prose statement of each invariant, with upstream PancakeSwap v3 and Uniswap v3 citations, lives in `docs/invariants.md`. The in-code NatSpec on every test is the single source of truth for the precise predicate.

§3.2. The CI receipts

The library's run summary at `RUN_SUMMARY.md §3` and `§W2-3` records the local proof from the operator host on 2026-06-27:

- **`forge build`** clean (compiler run successful, advisory `unsafe-typecast` and `mixed-case-function` lints only; not errors). Full log: `receipts/forge-build-2026-06-27-week2.log`.
- **`forge test --match-path 'invariants/*.t.sol' --no-match-path 'invariants/planted/*' -vv`** (clean leg across all five PancakeSwap v3 invariants): **31 tests passed, 0 failed, 0 skipped** across 4 suites, in 1.32 seconds. Four stateful-fuzz invariants run 256 runs × 50 depth = 12,800 calls each, with zero reverts. Full log: `receipts/forge-test-2026-06-27-week2.log`.
- **`forge test --match-path 'invariants/PancakeV3FeeGrowth.t.sol' -vv`** (P-1 clean leg in isolation, the planted-twin gate's clean side): exit 0, 5 of 5 tests pass, zero `INVARIANT VIOLATED` markers on stdout. Receipt: `receipts/planted_demo/p1-clean-passes-2026-06-27.log`.
- **`forge test --match-path 'invariants/planted/PancakeV3FeeGrowth.planted.t.sol' -vv`** (P-1 planted leg, the planted-twin gate's planted side): exit 1, 2 of 2 tests fail, five `INVARIANT VIOLATED feeGrowth_neverDecreases` markers on stdout. Receipt: `receipts/planted_demo/p1-planted-fires-2026-06-27.log`.

Both CI invariants the `p1-feegrowth-clean-passes` and `p1-feegrowth-planted-fires` jobs gate on are satisfied at HEAD. A reviewer reads the verdict from the four log files directly; nothing in this paper is a claim the receipts do not back.

Provenance for reproducibility: `forge-std @ v1.9.4` (tag `1eea5bae12ae557d589f9f0f0edae2faa47cb262`), Foundry `1.7.1` (commit `4072e48705af9d93e3c0f6e29e93b5e9a40caed8`), solc `0.8.28`, macOS arm64, 2026-06-27. To reproduce:

```
git clone https://github.com/caliperforge/bsc-invariants
cd bsc-invariants
forge install foundry-rs/forge-std@v1.9.4
forge build
forge test -vv
```

§3.3. The repo and the license

The public artifact lives at `github.com/caliperforge/bsc-invariants` under Apache License 2.0 (born-with-license from commit one; `LICENSE` + `NOTICE` ship in the initial scaffold). The `NOTICE` file carries the upstream attributions for the v3 tick math constants reproduced from Uniswap v3 and the patterns ported from CaliperForge's `hyperevm-safety` and `invariant-atlas` artifacts. `AI_DISCLOSURE.md` per the CaliperForge register names the AI-touched surfaces and the independent review discipline; `SECURITY.md` carries the responsible-disclosure norm; `SCOPE.md` is the single source of truth for protocols × invariants × status × milestone.

The library composes on top of two existing CaliperForge public artifacts: `hyperevm-safety` (HyperEVM invariants plus the JELLY incident reproduction, the source of the planted-twin convention this library ports) and the EVM cases in `invariant-atlas` (the cross-VM defender-side exploit corpus). The methodology brief at `caliperforge.com/methodology` describes the full cross-VM context.

§4. The roadmap

The library is structured across three milestones, each one a named BSC DeFi protocol with a self-contained deliverable shape: invariants in-tree, planted-twin pair landed, public release. The milestone partition matches the SCOPE.md table; each milestone flips its CI jobs from path-gated `deferred` to `real-work-green` the moment the wiring file lands.

§4.1. M1, PancakeSwap v3 (core shipped and public; funded milestone completes coverage)

Status: harness core shipped and public; funded milestone completes the coverage. The PancakeSwap v3 harness is live and reachable today at `github.com/caliperforge/bsc-invariants` under Apache-2.0. P-1 through P-5 are in-tree at HEAD and CI-green (§3.1). The P-1 clean / planted CI pair is shipped and public (§3.2). Tests: 31 / 31 green. A reviewer can clone the repo, run `forge test -vv`, and read the four `receipts/planted_demo/` log files directly today.

The funded M1 milestone closes the gap between the harness core and full PancakeSwap v3 coverage. The work that has not yet landed and that the milestone funds is: P-6 (protocol-fee accrual bound), P-7 (oracle observation cardinality monotonicity), planted-twin CI pairs for P-2 and P-3, the wrap-around variant of P-5, the first BSC-mainnet fork test against a real PancakeSwap v3 pool at a pinned block, and the Recon Chimera `Properties.sol` plus `CryticTester.sol` scaffold (the bundle that lets the same property surface run under Echidna and Medusa stateful fuzz alongside Foundry). The milestone is not "started from zero" work; it is the completion pass on a harness whose core a reviewer can read out of the public repo right now.

The Week-3 fork-test landing is the highest-value remaining add. A property that holds against a same-source reference under fuzz is a useful claim. A property that holds against the actual deployed PancakeSwap v3 contracts at BSC block N, sampled over a window of real swap activity, is a stronger claim. The fork test is what closes that gap; M1 is complete when it lands alongside the remaining P-6/P-7 invariants and the P-2/P-3 planted twins.

§4.2. M2, Venus lending

Status: not started. Venus is the canonical BSC lending protocol, a Compound-v2-class fork with BSC-specific governance settings and the XVS reward-accrual surface. The M2 invariant set targets the highest-impact failure modes the protocol has historically been exposed to:

- **V-1 Account liquidity correctness.** `getAccountLiquidity(user)` shortfall is consistent with on-chain collateral × `collateralFactor` minus borrows × price. The property catches the bug class where a refactor of the price oracle, the collateral factor cache, or the borrow accrual silently drifts the shortfall computation out of sync with the actual on-chain state.
- **V-2 Collateral factor bounds.** Per-market `collateralFactorMantissa ≤ 0.9e18` (the Venus governance cap). The property catches governance-action misconfiguration and any code path that overwrites the cap.
- **V-3 Liquidation incentive monotonicity.** `liquidationIncentiveMantissa ≥ 1e18` and seized collateral honors the configured incentive. The property catches the bug class where a math refactor under-credits the liquidator and creates a liveness gap (no liquidator profits to call `liquidateBorrow`, bad debt accumulates).
- **V-4 XVS reward accrual bound.** XVS reward accrual per market is bounded by `venusSpeed × blockDelta`. The property catches the accrual-runaway bug class that, in extreme form, is the same shape as the 2021 XVS-collateral incident on Venus.

Each invariant ships with a planted-twin pair. Each pair surfaces `INVARIANT VIOLATED` on the planted leg and exits cleanly on the clean leg. M2 closes when all four invariants and all four planted twins are in-tree, CI-green, and documented under `docs/invariants.md`.

§4.3. M3, Stargate-on-BSC bridge

Status: not started. Stargate is the canonical cross-chain bridge presence on BSC, threading the LayerZero message protocol with BSC-specific chain-id mappings and pool-delta accounting that has historically been the bridge-class exploit surface in the broader DeFi ecosystem. The M3 invariant set targets the credit-accounting and conversion safety boundaries:

- **S-1 Pool delta accounting.** `deltaCredit + lkb` (locally booked balance) reconciles with the sum of remote-chain credits the pool has received messages for. The property catches the credit-tracking drift bug class that, in extreme form, has been the failure mode of cross-chain bridge exploits across multiple ecosystems.
- **S-2 LD / SD conversion safety.** `amountSD = amountLD / convertRate`, with round-trip loss bounded by `convertRate`. The property catches the precision-loss bug class that surfaces when a downstream integration assumes lossless conversion at non-trivial scale.
- **S-3 Credit tracking monotonicity.** `credits[chainId][poolId]` is monotonic between credit-chain-path calls. The property catches the bug class where a re-entry or a message-replay path inadvertently double-credits a remote chain.

Each invariant ships with a planted-twin pair. M3 closes when all three invariants and all three planted twins are in-tree, CI-green, and documented.

§4.4. Cross-milestone discipline

Three discipline elements are constant across M1, M2, and M3:

1. **Public release on completion.** Each milestone's deliverable is an Apache-2.0 release tag with a versioned `RUN_SUMMARY.md`, a frozen `SCOPE.md` snapshot, and a `receipts/` directory committed to the repo. The release is the audit trail.
2. **Independent code review before public release.** The CaliperForge `code_quality_reviewer` runs the §4b independent read against the canonical code authoring standard (`agents/ai_ops/policies/code_authoring_standard.md`) before any milestone is tagged. Author ≠ reviewer is a structural gate, not an editorial nicety.
3. **No claim before build.** This paper, the README, the SCOPE table, and every external surface cite only what ships at HEAD. Properties listed in §4.2 and §4.3 are roadmap commitments stated as future work, not as features the library currently has. When V-1 ships, the SCOPE row flips from ⬜ to ✅, the RUN_SUMMARY records the test count, the receipt is committed, and the paper's next revision moves the V-1 row out of §4.2 and into §3.1.

The roadmap is shipped in the same shape as the M1 work the reviewer can already audit. That is the strongest evidence the M2 and M3 work will land in the same shape.

§5. Ecosystem benefit to BNB Chain

§5.1. A reusable public good for BNB Chain developers

`bsc-invariants` is Apache-2.0 from commit one. Every line of source, every test, every receipt, every doc is freely usable by any BNB Chain ecosystem developer for any purpose, including commercial integration into a downstream product. The library is shipped as a public good in the literal license sense.

Concretely, three reuse paths the library enables:

1. **A protocol team forking `bsc-invariants` for their own PancakeSwap-v3-fork.** Every BSC AMM that forks Uniswap v3 / PancakeSwap v3 source inherits the same fee-growth accounting math, the same tick-bound surface, and the same per-tick `feeGrowthOutside` conservation property. A fork team adopts `bsc-invariants` by pointing the harness at their own reference contract and running `forge test`. The planted-twin discipline scales: if the team modifies the fee-growth math, the planted leg flags whether the modification weakens the invariant the same way the canonical bug class would.
2. **A Risk Scoring Frameworks grantee building on the safety-rating primitive layer.** The BNB Chain H2 2025 wishlist (`bnb-chain/community-contributions/2025/2025h2.md`) names "Risk Scoring Frameworks: Standardized safety ratings (audit status, TVL stability, centralization risks)" as a category-2 priority. A standardized safety rating that asserts "this protocol passes a sharp set of invariants" needs the sharp set to exist as a reusable, auditable artifact first. `bsc-invariants` ships that artifact for PancakeSwap v3 today, and for Venus and Stargate at M2 and M3. A Risk Scoring Frameworks grantee can call into `bsc-invariants` invariants directly, or fork the property surface into their own evaluator.

3. **A future BNB Chain agent-registry tool integrating the verifiable-capability discipline.** The BNB Chain 2026 Tech Roadmap (https://www.bnbchain.org/en/blog/tech-roadmap-2026) names "an agent registry supporting identity, reputation scoring, and verifiable capabilities" as a flagship roadmap priority. A verifiable capability is, mechanically, an invariant the agent asserts about itself, and the harness that falsifies it. The planted-twin discipline `bsc-invariants` ships on DeFi properties today is the same discipline that the agent-registry's verifiable-capability layer will need to assert against agent behavior. The library is upstream of that future tool.

§5.2. Direct alignment to published BNB Chain priorities

The library is built against named items on the BNB Chain wishlist and the BNB Chain 2026 Tech Roadmap. Two of those alignments are direct:

- **H2 2025 wishlist, category 2 (Risk Scoring Frameworks).** The safety-rating primitive layer the wishlist asks for is exactly what a sharp invariant suite plus a planted-twin discipline provides. `bsc-invariants` ships the primitive; downstream Risk Scoring tools compose on top of it.
- **2026 Tech Roadmap, AI agent registry with verifiable capabilities.** The discipline `bsc-invariants` applies to DeFi protocols today is the same falsifiability primitive the agent registry's verifiable-capability claim will need. The library establishes the BSC-targeting discipline now, in shipped form, so the agent-registry integration at M3 (gated on the spec landing publicly) is a discipline port, not a discipline invention.

Two secondary alignments are indirect but real:

- **2026 Tech Roadmap, new execution engine and EIP-7928 BAL parallel execution.** Any new execution engine introduces new state-transition surface where invariant testing is the natural pre-deploy gate. A library that has shipped the discipline on the existing surface is the foundation downstream tooling against the new surface will build on.
- **2026 Tech Roadmap, re-architected storage layer.** The same shape applies: invariant testing is the natural pre-deploy gate for any storage-layer redesign that touches the contract-state representation. The library's discipline ports.

§5.3. The compounding effect

A single Apache-2.0 library that ships invariants for three of the highest-TVL BSC DeFi surfaces (PancakeSwap, Venus, Stargate) is a direct ecosystem benefit. A library that also establishes the planted-twin discipline as a reusable pattern across BSC is a larger benefit, because the next BSC DeFi protocol to adopt invariant testing inherits the pattern rather than re-inventing it. The library's M2 and M3 work compounds the M1 release: each milestone adds a new protocol's surface and demonstrates the pattern's portability.

The CaliperForge organization that builds `bsc-invariants` has prior public Apache-2.0 artifacts on adjacent chains (`hyperevm-safety` on HyperEVM, `invariant-atlas` cross-VM, `cf-invariants` on Cairo, `cf-modeleval` on LLM safety properties). The pattern of "ship public, ship under Apache-2.0, ship with receipts" is the operating model, not a one-off. BNB Chain's investment in `bsc-invariants` joins a multi-VM artifact line, not a single-grant deliverable.

§6. Honest scope

`bsc-invariants` is a pre-deploy property-testing library shipped as Apache-2.0 open source. It is not a security audit, not a runtime monitoring service, and not exhaustive coverage of the BSC DeFi stack at v0.0.2: PancakeSwap v3 is M1, Venus is M2, Stargate is M3. The properties it ships catch the specific bug classes the planted twins encode within the case timeout; they do not prove the absence of those classes on a protocol's full deployed code. The library composes with formal-verification lines (Halmos, Certora) and with runtime-monitoring lines (Forta, Hypernative, third-party BSC monitors) where those exist; it does not replace either. The honest-scope statement also appears verbatim in the public repo's README, in `SCOPE.md`, and in `RUN_SUMMARY.md`'s known-gaps section, where a reviewer can read each milestone's documented limitations directly.

§7. References

§7.1. The artifact

- **`bsc-invariants`** (this paper's subject; PancakeSwap v3 invariants P-1 through P-5 plus the P-1 planted-twin CI demo; Weeks 1+2 in-tree at HEAD; Apache-2.0). `github.com/caliperforge/bsc-invariants`. Reproduction receipts in `receipts/`.

§7.2. The CaliperForge artifact line (Apache-2.0 public good)

- **`hyperevm-safety`** (HyperEVM-specific invariants plus the JELLY incident reproduction; Apache-2.0; CI-green). `github.com/caliperforge/hyperevm-safety`. The source of the planted-twin convention `bsc-invariants` ports.
- **`invariant-atlas`** (cross-VM defender-side exploit corpus, v0.1, seven cases; Apache-2.0; CI-green). `github.com/caliperforge/invariant-atlas`. The EVM cases C6 and C7 are the closest cross-VM siblings of `bsc-invariants`.
- **`cf-invariants`** (Cairo lending-market property suite; Apache-2.0; CI-green). Upstream of Atlas case C1.
- **`cf-modeleval`** (planted-twin discipline applied to LLM safety properties, v1; Apache-2.0; CI-green; twelve-cell matrix, eleven hold, one honest miss documented in `DISCLOSURE.md §L6`). `github.com/caliperforge/cf-modeleval`.

§7.3. Companion technical brief

- **The CaliperForge Methodology.** Technical brief on planted-twin CI as a falsifiability primitive for invariant authoring. `caliperforge.com/methodology` (Apache-2.0). The cross-VM method this paper applies to BSC.

§7.4. BNB Chain official surfaces

- **BNB Chain Grants program.** `https://www.bnbchain.org/en/grants`. The program this paper's Field 4 points to.
- **BNB Chain community-contributions wishlist.** `https://github.com/bnb-chain/community-contributions/blob/main/2025/2025h2.md`. The H2 2025 wishlist, including category 2 "Risk

Scoring Frameworks: Standardized safety ratings (audit status, TVL stability, centralization risks)" referenced in §5.

- **BNB Chain 2026 Tech Roadmap.** `https://www.bnbchain.org/en/blog/tech-roadmap-2026`. The roadmap naming the AI agent registry with verifiable capabilities, EIP-7928 BAL parallel execution, the new execution engine, and the storage layer redesign referenced in §1.3 and §5.

§7.5. BSC DeFi protocols this library targets

- **PancakeSwap v3.** `https://github.com/pancakeswap/pancake-v3-contracts`. The canonical BSC DEX; M1 target.
- **Venus Protocol.** `https://github.com/VenusProtocol/venus-protocol`. The canonical BSC lending market; M2 target.
- **Stargate.** `https://github.com/stargate-protocol/stargate`. The canonical cross-chain bridge with BSC presence; M3 target.

§7.6. Upstream references reproduced under attribution

- **Uniswap v3 `TickMath.sol`.** `https://github.com/Uniswap/v3-core/blob/main/contracts/libraries/TickMath.sol`. Source of `MIN_TICK`, `MAX_TICK`, `MIN_SQRT_RATIO`, `MAX_SQRT_RATIO` constants reproduced under attribution in `src/lib/TickMath.sol` and credited in `NOTICE`.

§7.7. Engines composed

- **Foundry** (`forge`, `1.7.x`). Pinned in `foundry.toml`.
- **forge-std** (`v1.9.4`). Pinned in `remappings.txt`.
- **Recon Chimera** (`chimera-template-pack`). Echidna + Medusa stateful-fuzz wrapper that the M2 Week-3 bundle lands.
- **Halmos**. Bounded symbolic execution; cross-checks for properties that admit it.

§7.8. Related work the library composes with

- **Trace2Inv** (Chen et al., FSE 2024, `arXiv:2404.14580`). Defender-side runtime invariant benchmark on twenty-seven historical EVM exploits. The peer-reviewed self-measurement methodology the CaliperForge artifact line imports as its discipline posture (set out in §3 of the companion methodology brief).
- **DeFiHackLabs** (`github.com/SunWeb3Sec/DeFiHackLabs`). Attacker-side Foundry-PoC reproductions of historical EVM exploits, including multiple BSC incidents. A reviewer cross-referencing `bsc-invariants` planted twins against published BSC bug classes can use DeFiHackLabs as the public bug-class index.

---

§8. Provenance

- **Author.** Michael Moffett (operator), with the CaliperForge research_lead specialist (Opus tier) and the solidity_specialist (`bsc-invariants` build engineer).
- **Date.** 2026-06-27.

- **Sources of evidence.** The shipped `bsc-invariants` repository (`github.com/caliperforge/bsc-invariants`), its in-tree `RUN_SUMMARY.md` Weeks 1+2 dated 2026-06-27, and the companion CaliperForge methodology brief. The CaliperForge prior-art artifact line referenced in §5.3 and §7.2.
- **Independent review.** §4a content QA gate and §4b independent code-quality review (CaliperForge `code_quality_reviewer`, not the build engineer) before any external publication, per the CaliperForge organizational discipline.
- **License.** This white paper is Apache-2.0 alongside the artifacts it references.
- **Disclosure.** AI-augmented authoring under the CaliperForge operating model; the operator reviews and edits. Per-repository AI-touched surfaces are named in each repo's `AI_DISCLOSURE.md`. The organization-level register is at `caliperforge.com/ai-disclosure`.

Contact. michael@caliperforge.com, team@caliperforge.com.

End of paper.