Bitcoin Hyper
Technical due diligence review

An editorial review of Hacken's final security assessment of the **giga-svm validator client**, dated 24 November 2026.

Bitcoin Hyper describes its project as a Bitcoin Layer 2 with a Solana Virtual Machine (SVM) execution layer. It submitted giga-svm, a fork of the Anza Agave v1.18.21 validator, as its core node software. The fork shortens the nominal slot target from 400 to 32 milliseconds. It also introduces a path that lets a node process RPC submissions locally during its own leader slots. In the corrected design, this path operates alongside a service that forwards transactions and retries delivery.

Assessment record

Item	As recorded in the English report
Client	Bitcoin Hyper
Report	Blockchain Protocol Review and Security Analysis Report for Bitcoin Hyper
Audit and approval	Auditor: Sofiane Akermoun. Approved by Tanuj Soni. Hacken acknowledges the Bitcoin Hyper team's cooperation.
Report history	Preliminary report: 18 November 2026 Final report: 24 November 2026
Platform and language	Solana · Rust. Report tags: Layer 1, Solana.
Code under review	Yona-Labs/giga-svm bf6eec5134af396445f959458b3e2182ae5df258
Source documents	Original Hacken report, 57 pages · Hacken methodology

ABOUT THIS REVIEW

MS Digital Asset Academy prepared this reading on 25 November 2026. Hacken is the author of the underlying audit. Its findings, severity ratings and fix references are attributed to it throughout. For technical tests, this edition reports the outcome and the limits of the evidence. Page references refer to Hacken's English report. Editorial interpretation appears in reading notes.

Findings register and contents.

RESULTS AND CONTENTS

Hacken report · pp. 3 and 5

Findings register

Hacken records **nine security findings, all fixed**. Its summary lists zero accepted and zero mitigated. Two additional informational observations are also marked fixed and are counted separately.

Initial severity: **three critical, four high, zero medium and two low**. Hacken also directs users to the general risks on page 7 of its report. Those cover matters that are not resolved simply by closing a code finding.

Reference	Finding or observation	Severity	Pages
F-2026-20534	Signature verification on the direct RPC path	Critical	8–15
F-2026-20580	Validator availability and malformed RPC input	Critical	16–21
F-2026-20671	Missing upstream security and stability fixes	Critical	22–25
F-2026-20541	Transaction expiry on non-leader RPC nodes	High	26–31
F-2026-20548	Unbounded RPC queue growth	High	32–36
F-2026-20579	Leader discards transactions marked for retry	High	37–42
F-2026-20593	Recorded transactions reference unusable lookup tables	High	43–45
F-2026-20635	RPC worker prevents graceful shutdown	Low	46–47
F-2026-20650	RPC worker fails to respect vote-only mode	Low	48–49
F-2026-20669	Missing and broken tests for fork-specific changes	Info	50–51
F-2026-20670	Proof of History can outpace the nominal slot target	Info	52–53

Other sections

Documentation, architecture and testing (p. 4) · System overview and launch risks (pp. 6–7).

Assurance and limitations (p. 54) · Severity definitions (p. 55) · Technical scope (p. 56) · Source files and methodology (pp. 56–57).

Documentation, architecture and testing

Documentation

The repository includes Agave's guides for building, testing and operating validators. Its README describes the upstream Solana implementation. It does not document the fork-specific RPC transaction architecture or explain the new clock design. Comments alongside the relevant constants identify a rate of 2,000 ticks per second and a nominal slot duration of 32 milliseconds.

Code quality

The code retains Agave's modular Rust structure. In its overview, Hacken identifies Rust 1.85.0 as the pinned toolchain. In the corrected design, transaction submission verifies signatures and passes transactions to a forwarding and retry service. The direct path has a bounded queue, checks address lookup tables against the bank that will execute the transaction, respects vote-only mode and shuts its worker down when the channel closes.

The client is based on Agave's unmaintained 1.18 branch. Security and stability patches developed later upstream have been backported. Keeping that branch secure requires the fork's maintainers to continue identifying and adapting relevant fixes.

Architecture

Scalability. The transaction processing unit (TPU) supports parallel execution. The separate RPC fast path uses one worker and a bounded queue. The software targets 32 ms slots.

Decentralisation. Leader selection and voting are stake-weighted. The project intends to restrict admission to the validator set.

Performance. Each RPC submission enters the forwarding and retry service. The node can also queue a local copy to execute if it is the leader.

Resilience. The fork retains Agave's replay and repair mechanisms. Signature and lookup-table checks, bounded queues, retry handling, worker shutdown and upstream patches address risks identified during the assessment.

Test coverage

Agave's unit and integration tests are present. The report also lists coverage for the fork's RPC path, transaction forwarding and retries, queue limits, lookup tables, vote-only behaviour, PoH pacing and backported fixes.

> **READING NOTE**
>
> The overview describes the corrected software. The detailed finding descriptions examine defects in the initially audited revision. This difference in time explains why a safeguard can appear in the overview and be missing in the finding that follows.

SYSTEM OVERVIEW AND RISKS

Hacken report · pp. 6–7

How the validator works

Giga-svm executes transactions using the SVM. A stake-weighted leader schedule determines which validator is responsible for producing blocks at a given time. Other validators replay the transactions against their own state before voting. Bitcoin Hyper told Hacken that admission to the live validator set would be permissioned.

From submission to a block

Clients submit signed transactions through RPC or the TPU. The TPU verifies signatures and BankingStage schedules transactions for processing. In the corrected RPC design, the node verifies the signature and hands the transaction to the forwarding service. The service may retry delivery subject to retry and expiry limits.

The node also attempts to queue a copy for BankingStage. Local execution requires a leader bank, the working state used by the node during its turn to produce blocks. Other validators check the resulting block by replaying it independently.

Clock and configuration

At 2,000 ticks per second and 64 ticks per slot, a slot is nominally 32 ms. Four consecutive slots assigned to one leader therefore span 128 ms on paper. Genesis selects the Proof of History (PoH) tick mode. The fork also adds a native mint account at genesis, changes a snapshot slot-delta check and emits latency metrics for selected RPC transaction signatures.

The bridge, Bitcoin anchoring, explorer and wallet fall outside the audited repository.

Three risks Hacken identifies separately

Maintenance. Agave v1.18.21 belongs to a release line that no longer receives upstream fixes. Maintaining the fork requires continuing backports of relevant security and stability updates.

Launch configuration. Bitcoin Hyper plans to start from a fresh genesis. Hacken did not receive the final genesis configuration or the planned permissioned validator set. Supply, feature activations, PoH settings, validator count and stake distribution need to be assessed for launch. Validator count and voting concentration affect both liveness and consensus safety.

Performance. Release benchmarks had not established how the software would behave on the intended hardware and network topology. The 32 ms figure is a slot target; network-wide observations are needed to establish actual timing.

CRITICAL FINDINGS

Hacken report · pp. 8–15

Signature verification on the direct RPC path

F-2026-20534 · Severity: Critical · Impact 5/5 · Likelihood 5/5 · Fixed

The leader could record a transfer without a valid authorising signature. A second validator detected the invalid signature and rejected the block during replay.

Underlying issue

A transaction can be well formed without being authorised. Structural checks establish that the fields can be processed. A cryptographic signature check establishes that the holder of the relevant key authorised the operation. The latter must hold on every route into execution.

In upstream Agave, the transaction passed through TPU signature verification before BankingStage. The fork's direct RPC path introduced a route where the same guarantee did not always hold. The leader could therefore debit a transfer locally even though the account holder had not signed it.

What the test established

Hacken used two validators in a local test. It distinguished three observations: the RPC acknowledgement, the credit to the recipient in the leader's local state, and the transaction's presence in the leader's recorded block. The peer received the slot, identified the invalid signature and marked the slot dead. The peer's background replay thread later encountered an unrecoverable error while attempting repair.

The test passed because it observed the assertions demonstrating the defect. The replay-thread error occurred in the test environment. The production executable has process-level error handling that can terminate the validator.

Consensus implications

The consequences depend on the voting stake of the affected leader. With limited stake behind it, the other validators can reject the block. Greater concentration can turn disagreement and unsuccessful repair into a loss of liveness. A leader holding a voting supermajority could, in the scenario examined by Hacken, finalise the transfer on its branch. The experiment established local execution, recording and peer rejection; it did not demonstrate finality on a live public network.

Recommended remediation

Require cryptographic signature verification before admission to any execution path, including when preliminary simulation is optional. Maintain equivalent verification guarantees across the direct and relayed paths. Regression tests should exclude local execution, block recording and propagation of transfers with invalid signatures.

Fix recorded by Hacken 3ddddd5d.

CRITICAL FINDINGS

Validator availability and malformed RPC input

F-2026-20580 · Severity: Critical · Impact 5/5 · Likelihood 5/5 · Fixed

An input-handling error could bring down a validator. Hacken observed the process terminate when running the actual validator executable locally.

Underlying issue

The fork added latency measurements for RPC transactions. The measurement path made an assumption about data that had not yet been validated. As a result, malformed input could cause an unrecoverable runtime error before the RPC service could return a normal rejection.

The upstream Agave baseline rejected invalid input during validation. The repository contained a test for that behaviour. A change to the RPC module’s initialisation interface had left the inherited test unable to compile, so it could not identify the regression.

What the test established

Hacken reports two observations. In an isolated test a child process terminated with exit status 1. It then tested the fork’s validator executable on a local machine using the project’s own operating tools. Before the test, the validator was producing slots, voting and answering queries. After the error the process had exited and the RPC service no longer responded.

Effect on availability

Loss of one validator does not always imply loss of chain progress. With a single block producer, the loss of that process halts block production. In a larger validator set, the result depends on which nodes are reachable and how much voting stake is lost. If the remaining validators cannot reach the supermajority required by the protocol, new blocks can no longer attain the required confirmation or finality.

Recommended remediation

Collect the metric only after validating the data, handle missing values explicitly and avoid using helpers that assume well-formed input on unvalidated network requests. Restore the inherited unit test and keep an integration test that checks the service rejects invalid input while the validator stays available.

Fix recorded by Hacken 3ddddd5d.

CRITICAL FINDINGS

Hacken report · pp. 22–25

Missing upstream security fixes

F-2026-20671 · Severity: Critical · Impact 5/5 · Likelihood 4/5 · Fixed

Hacken grouped several independent defects because their fixes were available in upstream software or dependencies but missing from the audited fork. It rated the aggregate critical for deployments whose TPU transport service is reachable by untrusted clients. Network access controls affect likelihood. The underlying issues have individual ratings.

Six security issues

Component and rating	Evidence and limits
TPU QUIC transport Critical	Invalid transport data could terminate the networking process. Hacken observed the real TPU service exit in a child process using production error handling. The test covered the service rather than a complete validator. Reachability on a live network depends on deployment.
rBPF JIT compiler High	The compiler used for SBF programs could encounter a memory fault on x86-64 hosts. A fix had already been backported to Agave v1.18.22. Hacken had not established the CPU architecture of the intended Bitcoin Hyper validators.
Duplicate block fragments High	A buffer-handling error could stop the component that processes duplicate fragments. A direct component test reproduced the error. It did not test admission of messages through live gossip.
QUIC connection identifiers Medium	Retirement of connection identifiers could cause an internal queue to grow without a bound. The report explains the defect in the dependency. It did not measure memory growth on a validator running the fork.
Zero-knowledge proofs Medium	Some values were not correctly bound into equality and fee proofs. A local equality-proof test accepted inconsistent amounts before the upstream fix and rejected them afterwards. The local test covered equality only. Live impact depends on the features and proof contexts in use. No asset transfer was demonstrated.
Account-data budget Medium	The cost model could reserve the full account-data budget of a block for a failed transaction. A subsequent valid account creation was rejected by the cost tracker in a test. Execution in the bank and inclusion in a block were not tested.

These ratings describe components within F-2026-20671. The main register counts this group once, as a critical finding.

FINDING F-2026-20671 / CONTINUED

Hacken report · pp. 23–25

Stability fixes and ongoing maintenance

Three stability issues

Component and rating	Evidence and limits
Startup account verification Medium	Cleanup of saved state could conflict with an outstanding epoch accounts hash request during startup. A service-level test reproduced the failure. It did not include a full validator restart.
Replay on restart Medium	A stored duplicate proof relating to the root slot could cause an unrecoverable replay error when the relevant feature was active. The outcome depends on persisted state and feature configuration. No restart test reproduced this condition on the fork.
Replay batches Low	Encountering a previously confirmed slot could cause later slots in the same batch to be skipped. The behaviour was identified in the code. Its ultimate effect on fork choice was not reproduced.

Why the overall rating is critical

The highest rating rests on the TPU QUIC issue. Hacken directly observed the isolated service process terminate. In a deployment, loss of enough voting power could affect consensus availability. Availability of an entire network was not measured by this component test.

Recommended remediation and maintenance

Backport the relevant fixes from the 1.18 series and adapt subsequent changes in Quinn, gossip, zero-knowledge proofs and the accounts service to the fork. Exercise patched components against the retained regression tests. To assess deployment-specific exposure, confirm TPU and gossip reachability, CPU architecture and activated features.

Hacken identifies v1.18.26 as the final release of the Agave 1.18 line. The fork’s maintainers need to follow Anza and dependency advisories, select the changes relevant to their branch, adapt them and keep the resulting behaviour under regression testing.

Fix recorded by Hacken 23cedf83.

> READING NOTE
>
> The zero-knowledge issue concerned specific proof verifiers present in the validator repository. It provides no assessment of the full Bitcoin anchoring mechanism, which was outside the audit scope.

HIGH-SEVERITY FINDINGS

Hacken report · pp. 26–31

Transaction expiry on non-leader RPC nodes

F-2026-20541 · Severity: High · Impact 3/5 · Likelihood 4/5 · Fixed

An RPC node could return a signature even though it had not passed the transaction to a validator that could include it in a block.

How expiry arose

The direct RPC service added a transaction to a local queue and returned its signature to the client. The local worker then waited for its node to become leader. An unstaked RPC node has no leader slots, so a valid submission could expire in the queue without ever reaching a block producer. The project's configuration provided for unstaked nodes offering RPC.

For a staked node outside its leader slot, inclusion depended on how soon its turn arrived and whether the recent blockhash was still valid. Hacken cites a validity window of 1,875 blocks, nominally around 60 seconds at 32 ms. This off-turn case follows from code review. The test itself used an unstaked node.

Agave's submission service forwards transactions to current or upcoming leaders. The fork retained that service in the repository, but its original direct RPC path did not use it.

Two transfers, two outcomes

Submission route	Observed by Hacken
Leader	A control transfer of 1,000,000 lamports was credited to Bob. The lamport is the unit used throughout the test.
Unstaked RPC node	The node returned a signature for a second transfer of 2,000,000 lamports. After the blockhash expired, the transaction had no recorded status and Bob's balance was still 1,000,000.

Recommended remediation

Forward submissions when the node is outside a leader slot. A leader-only service should reject such requests explicitly and direct the client to a suitable producer. Return an error if no route accepts the transaction. Preserve signature checks on every route and test unstaked nodes, off-turn staked nodes and changes of leader.

Fix recorded by Hacken 3ddddd5d.

Reading note. An RPC acknowledgement is evidence that a node received the request. Block inclusion and transaction finality each require further evidence.

HIGH-SEVERITY FINDINGS

Hacken report · pp. 32–36

Unbounded growth of the RPC queue

F-2026-20548 · Severity: High · Impact 3/5 · Likelihood 4/5 · Fixed

Transactions could remain in the local queue after expiry. Hacken measured how many entries the worker retained.

What allowed the queue to grow

The local RPC transaction channel had no capacity limit. The worker took an initial batch and waited for a leader bank before reading the next one. On an unstaked node the wait never ended. Further submissions built up behind the first batch, including transactions whose blockhash had expired.

A staked node could also accumulate requests between turns. A separate issue arose when the queue was empty: the worker repeatedly polled it instead of waiting for a notification or timeout. Agave v1.18.21 also used an unbounded intake channel, but its submission service could drain the channel outside local leader slots, cap its retry pool at 10,000 and remove expired transactions.

What the evidence supports

In Hacken's component test, **9,872 transactions** remained queued after the worker took its first batch. The count stayed at 9,872 after their blockhash expired. When a second group arrived, the queue reached **19,872**. The size of the worker's initial batch can vary with processing speed.

The observation establishes that expired entries were retained. The test did not measure memory or CPU consumption, exhaust resources, or exercise the public HTTP service. Memory pressure and eventual loss of availability were projected consequences.

Recommended remediation

Coordinate queue management with transaction forwarding. The service should drain the queue independently of the local leader schedule or reject a submission before enqueueing it. Bound the queue, signal overload to clients, expire stale submissions and wait efficiently when idle. A retained local execution path still needs mandatory signature checks.

Fix recorded by Hacken 3ddddd5d · completed in 23cedf83.

HIGH-SEVERITY FINDINGS

Hacken report · pp. 37–42

Leader discards transactions marked for retry

F-2026-20579 · Severity: High · Impact 3/5 · Likelihood 4/5 · Fixed

A valid transaction may need another attempt. BankingStage's consumer identified these transactions, but the original direct RPC worker cleared the batch without keeping them. Clients had already received a signature. Simulating each request separately cannot anticipate all conflicts when a leader processes several requests together.

When another attempt may be needed

Condition	Why execution may be deferred
Account contention	Transactions may compete for write access to an account or find it locked by another worker. Hacken exercised contention within one batch.
Slot cost limit	The remaining compute, vote, account or account-data budget may be too small. A transaction can still be eligible for a later slot.
Slot boundary	A slot can finish before the full batch is processed and recorded. Valid unprocessed transactions need to survive the transition.

Agave retains eligible transactions for bounded retry. The fork's initial direct route bypassed both retry mechanisms. Frequent writes to the same account make this especially relevant. As a hypothetical application consequence, Hacken cites an order that remains executable after its cancellation request is lost.

Test evidence and limits

Alice submitted two valid transfers of 1,000 lamports. The transfer to Bob was recorded. The transfer to Carol was missing and the worker's queue was empty. Resubmission of the same signed transaction credited Carol. The test used the worker directly and an extended slot to isolate account contention. It did not reproduce the cost-limit or end-of-slot cases.

Recommended remediation

Keep eligible transactions for a local retry or forward them to the next leader. Bound the retry queue and discard expired entries. Form batches with compatible writes, and test intra-batch contention, locks held by other workers and handover between slots. Track whether each eligible transaction ultimately lands or expires.

Fix recorded by Hacken 3ddddd5d · completed in 5c1a5821.

HIGH-SEVERITY FINDINGS

Hacken report · pp. 43–45

Lookup tables could become unusable before execution

F-2026-20593 · Severity: High · Impact 4/5 · Likelihood 3/5 · Fixed

The leader could record a transaction with a valid signature even though a peer replaying the block could no longer resolve the same account addresses.

Where the states diverged

Versioned transactions can refer to accounts through an address lookup table. The RPC service may prepare a transaction against an earlier bank than the one the leader will use for execution. A table that was usable in the earlier state can be unavailable in the later one.

The direct RPC path resolved addresses on admission and queued the result. At execution the worker checked conditions such as blockhash validity but did not recheck the table's availability. A peer replaying the block would resolve addresses against the relevant bank. Agave's reference path performs preparation or revalidation against the execution state.

What Hacken observed

A transfer from Alice to Bob resolved against the first bank. A fresh check in the leader bank reported the lookup table unavailable. The local worker nevertheless credited Bob with 1,000 lamports and recorded the signed transfer in a PoH entry. The verifier used for replay subsequently reported that it could not resolve the table.

The test controlled the two slot states and exercised the worker directly. It did not exercise HTTP bank selection, run a second validator or measure the timing window on a production network. Peer rejection of the slot and consequences for continued block production were inferred from code review.

Recommended remediation

Revalidate table availability against the bank that will execute the transaction and discard stale references. An alternative is to prepare the transaction again at execution as the TPU path does. A regression test should confirm that an unusable table prevents execution and keeps the transfer out of the leader's recorded entries.

Fix recorded by Hacken 23cedf83.

LOW-SEVERITY FINDINGS

Hacken report · pp. 46–49

Graceful shutdown and vote-only mode

RPC worker prevents graceful shutdown

F-2026-20635 · Severity: Low · Impact 1/5 · Likelihood 4/5 · Fixed · pp. 46–47

When the RPC transaction channel closed, the worker left its inner receive loop and entered the outer loop again. It did not observe the validator's shutdown signal. BankingStage then waited indefinitely for the worker to finish.

The validator's shutdown command could eventually trigger Agave's five-second fallback, which terminates the process before normal cleanup. Tests that shut down a validator could hang. A node without RPC could also be left waiting when a fatal internal error, such as a snapshot archiving failure, requested shutdown.

Recommended remediation. Exit the worker's main loop when the channel closes or the validator signals shutdown, including while waiting for a leader slot. Verify that shutdown of a single-node test cluster returns control.

Fix recorded by Hacken 3ddddd5d.

Direct RPC path bypasses vote-only mode

F-2026-20650 · Severity: Low · Impact 2/5 · Likelihood 2/5 · Fixed · pp. 48–49

Agave enables vote-only mode when the local root falls more than 400 slots behind the slot the node is about to produce. Restricting the workload gives it room to vote and catch up. Both TPU schedulers enforced the rule.

The direct RPC worker continued executing and recording ordinary valid transactions. Peers do not enforce a producer's local vote-only policy during replay, so the block can remain valid. The risk concerns capacity available for recovery and voting.

At the fork's nominal rate, 400 slots span 12.8 seconds. At Agave's 400 ms slot target they span 160 seconds. Hacken did not quantify the impact on vote inclusion or catch-up time.

Recommended remediation. Apply the TPU path's vote-only selection to direct RPC batches before execution.

Fix recorded by Hacken 23cedf83.

INFORMATIONAL OBSERVATIONS

Missing and broken tests for fork-specific changes

F-2026-20669 · Informational observation · Fixed

The fork introduced a transaction path and a new clock configuration. The inherited test suite had not kept pace with these changes.

Comparison with upstream

Across the first six fork revisions reviewed, Hacken found no new test files, additional unit cases or updates to existing test bodies. The inherited modules were still present, but test targets failed to compile in three Rust packages.

Package	Issue reported
solana-core	Tests still called an earlier BankingStage constructor and referenced a timing constant that had been removed.
solana-rpc	Tests used older RPC interfaces. Five also called an unimplemented helper, leaving a potential runtime failure even after compilation was repaired.
solana-program	Clock tests still expected the old constant values.

All three test targets passed the corresponding build check on the upstream revision. Separately, a snapshot unit test compiled but failed after the fork changed a limit.

What this meant for assurance

A test for invalid RPC requests no longer compiled after an argument changed. It could have flagged the critical availability issue. Another integration test made a valid transfer and then shut its validator down. Hacken inferred that shutdown would hang in the initial code. It did not run that particular test.

The default coverage workflow needed to compile the affected packages first, so it yielded no coverage measurement at the initial revision. Actual coverage and any separate testing performed by the developers were unknown.

Recommended remediation

Restore the inherited tests, update constructors, helpers and clock expectations, then execute the suite. Address the shutdown hang and add coverage for the new RPC and BankingStage behaviour.

Fix recorded by Hacken 23cedf83.

INFORMATIONAL OBSERVATIONS

Proof of History and the 32 ms slot target

F-2026-20670 · Informational observation · Fixed

The fork specifies 2,000 ticks per second and 64 ticks per slot. This yields a nominal 500 microseconds per tick and 32 milliseconds per slot. Genesis selects how validators generate Proof of History (PoH) ticks.

Timer-based mode

Agave designates this low-power mode for development. The fork's demonstration configuration selected it. A timer paces ticks while a node operates normally. During replay there is no cryptographic work requirement that independently establishes elapsed time, so pacing depends on the producer.

Bitcoin Hyper told Hacken that production validators would be permissioned and that the production genesis would not be generated with the demonstration script. Permissioning reduces the risk associated with a participant running modified software.

Hash-based mode

Replay checks the hash count. Under the timing constants reviewed, the adjustment intended to pace faster validators could leave no meaningful wait. The report discusses 1,000 hashes per tick, sized around a baseline of 2 million hashes per second.

Hacken cites a July 2023 Agave sample in which hardware representing 96.2% of stake could exceed 10 million hashes per second. At that rate 64 ticks would take about **6.4 ms of hashing alone**. This is an arithmetic illustration using an external historical sample. Actual Bitcoin Hyper block times require measurements from its own validators.

Evidence needed at launch

Obtain the deployed hashes per tick, target tick duration and ticks per slot. Benchmark release builds on the intended hardware and network. Measure tick and slot duration on faster and slower machines, block production across leader transitions, and peer replay.

For hash-based PoH, scale the timing adjustment to the slot duration and size the workload against the slowest intended validator with headroom. For timer-based operation, document what replay can verify under the permissioned trust model.

Fix recorded by Hacken 23cedf83.

Hacken did not receive the production PoH mode or parameters. It did not run a release benchmark for this observation.

Terms and limits of Hacken's assessment

How Hacken defines its assurance

Hacken says it assessed the protocol submitted to it using the industry practices available at the time. Its report considers the source-code security issues it describes, together with compilation, deployment and the code's ability to perform its intended functions.

The report gives no warranty that every vulnerability was found or that the code is secure. Its conclusions apply to the submitted revision. Later modifications can affect whether a finding or conclusion still applies. Hacken cautions against treating this assessment alone as sufficient assurance of the protocol's utility, safety or freedom from defects.

Hacken recommends multiple independent audits and a public bug bounty programme to broaden review and encourage responsible disclosure. Its authoritative text is in English, and it assumes no responsibility for renderings into other languages.

Quality assurance and authoritative versions

Hacken may independently reassess selected projects as part of its internal quality programme. Updated reports are shared privately with the respective clients. Publication on Hacken's website requires their explicit consent. Hacken identifies its own audit archive as the authoritative source for final and current reports. Hacken audit archive.

Software dependencies

A blockchain protocol runs on a wider software platform. Weaknesses in that platform, its language implementation or other dependencies can compromise an application running on it. A protocol review therefore cannot establish absolute security.

> **SOURCE AND DATE**
>
> This editorial review uses Hacken's English report dated 24 November 2026, supplied on 25 November. The date printed on the report and the date on which a particular copy first became public are separate facts. English source report.

APPENDIX 1

How to read the severity ratings

These are Hacken's general severity categories. The examples explain what each rating means across blockchain audits.

Critical

A critical weakness can compromise security, confidentiality, integrity or availability across a blockchain network. Consensus manipulation or control of a majority of network participants can have effects comparable to a 51% attack.

The category also covers widespread node failures leading to a substantial or complete network outage, and the potential for large-scale asset theft. These issues call for immediate attention and mitigation.

High

High-severity weaknesses can cause substantial harm. Examples include several nodes becoming unavailable, a temporary service interruption, or manipulation of part of the consensus process short of majority control.

Hacken also includes partial privacy breaches, limited unauthorised access to sensitive information and defects that materially undermine reliable smart contract execution.

Medium

Medium-severity issues can damage the protocol, usually with more limited consequences. Examples include limited effects on user privacy, slower transaction processing or relatively modest financial losses. Particular circumstances or elevated access may be required.

Low

Low-severity defects include small inefficiencies in transaction processing, minor block-propagation delays and isolated disruption. Some arise only under rare conditions. Hacken recommends addressing these defects even when they pose little immediate threat.

APPENDIX 2

Technical scope of the engagement

Hacken defines the scope by the repository, its initial revision and the specific components submitted for review.

Repository Yona-Labs/giga-svm · **Revision reviewed**
bf6eec5134af396445f959458b3e2182ae5df258
White paper referenced in the report.

Components listed in scope

Area	Items listed by Hacken
RPC submission	Direct delivery to local block production, signature checks in preflight and on the production path, acknowledgement during queueing and per-transaction latency measurements.
Block production	Batching, inclusion during local leader slots, queue growth, arrival order, absence of fee-based ordering on the direct route, and shared local commit and recording stages.
Network clock	2,000 ticks per second, 32 ms slots, epoch duration, blockhash validity window and a 200 ms repair delay.
Genesis setup	Wrapped SOL mint, its authority, supply and 1 SOL balance. A single upgrade authority for 17 initial programs. Timer-based PoH in the supplied cluster startup configuration.
State snapshots	A historical limit of 4,000 entries against an operating window of 3,750.
Supporting changes	Preallocation for balance collection and local reset scripts.

Configuration code and deployment evidence

This appendix lists configuration code present in the repository. On page 7, Hacken says that it had not received the final genesis values or membership of the planned permissioned validator set. The supplied setup therefore required confirmation against the actual launch configuration.

Hacken’s overview excludes the bridge, Bitcoin anchoring, explorer and wallet from the audited repository.

The next page lists all 18 files in the stated scope and the source-control revisions linked to remediation.

APPENDICES 2 AND 3

Hacken report · pp. 56–57

Source files and methodology

Files in the stated scope

Repository files · 1–9	Repository files · 10–18
core/src/banking_stage.rs	multinode-demo/setup.sh
core/src/banking_stage/consumer.rs	programs-genesis.sh
core/src/banking_stage/decision_maker.rs	reset_and_restart_metrics.sh
core/src/banking_stage/leader_slot_metrics.rs	reset_validator.sh
core/src/repair/repair_service.rs	rpc/src/rpc.rs
core/src/tpu.rs	rpc/src/rpc_service.rs
core/src/validator.rs	runtime/src/bank.rs
genesis/src/main.rs	runtime/src/snapshot_bank_utils.rs
ledger/src/token_balances.rs	sdk/program/src/clock.rs

Remediation revisions

The finding sheets use the first eight characters of each revision. Hacken gives these full commit identifiers in the English report.

```
3ddddd5d3053501b5c7be58216c929aa52d1179c
23cedf8322c305a969518be2949c0f9a941506be
5c1a5821169bc5c3475b80297fbf70f17a9c5967
```

Assessment frameworks

Hacken cites the NIST Technical Guide to Information Security Testing and Assessment (SP 800-115) and the Penetration Testing Execution Standard (PTES). These frameworks guide the planning, conduct and documentation of technical assessments, including review of source code and validation of security impact.

Hacken applies its own methodology for blockchain protocols to support consistent, repeatable assessments across Web2 and Web3. The English report is the source for the auditors' observations, recommendations and remediation status.

References listed by Hacken: NIST SP 800-115 · PTES · Hacken methodology.

Return to the findings register and contents.