BREWTFORCE

HotShot Internal Review

Summary

Project Name	Espresso
Language	Rust
Codebase	https://github.com/EspressoSystems/HotShot
Delivery Date	29/07/2024
Team	0xKato, Jarred Parr
Commit(s)	9cd03d3caddbfe6b353aeb291fed312add9a8208

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	2
High	1	0	0	0	0	1
Medium	3	0	0	0	0	3
Low	3	0	0	0	0	3

Findings & Resolutions

ID	Title	Category	Severity	Status
TIQP-1	QuorumProposalTask Is Vulnerable To Certificate Forgery Attacks	Censorship Attack, Liveness Violation	Critical	Confirmed
TIQV-1	QuorumVoteTask Is Vulnerable To Certificate Forgery Attacks	Censorship Attack, Liveness Violation	Critical	Confirmed
O-1	Orchestrator Slots May Be Trivially Exhausted	Liveness Violation	High	Confirmed
QPRM-1	The Liveness Branch Of The QuorumProposalRecv Task Will Arbitrarily Block Voting	State Violation	Medium	Confirmed
QVM-1	VoteNow Uses An Invalid Leaf	State Violation	Medium	Confirmed
TIST-1	Back-To-Back Leaders May Not Build Blocks	State Violation	Medium	Confirmed
TIST-2	Race Condition in Validated State Handling: Update vs Read Conflict	Race Condit	Low	Confirmed
HSL-1	HotShot does not alert when running with dependency tasks.	Typo	Low	Confirmed
Global-1	Insufficient validation in create_vote_accumulator	State Violation	Low	Confirmed

TIQP-1 | QuorumProposalTask Is Vulnerable To Certificate Forgery Attacks

Category	Severity	Location	Status
Censorship Attack, Liveness Violation	● Critical	crates/task-impls/quorum_proposal	Confirmed

Description

Due to the lack of validation in the `QuorumProposalTask`, it is vulnerable to a certificate forgery attack in which any staked attacker may arbitrarily grief the entire network by forging a signature-valid, but content-invalid quorum proposal, and spam this message arbitrarily to all other nodes within the network. This is a two-pronged attack, with two principal vectors of exploitation:

1. Exhaustion of available dependency task slots in the quorum proposal, resulting in timeouts and view sync triggers.
2. An overflow of quorum proposals resulting in induced timeouts for the service and overall low throughput.

However, there is an even more insidious quality to this type of attack. The choice to cancel the tasks on any quorum proposal results in the circumstance in which valid quorum proposals are cancelled, resulting in the malicious attacker always having the ability to arbitrarily censor any proposal that they choose by simply forging a certificate for view `n+1` and spamming it. Since the tasks are pipelined, they do not complete processing before handling another, and can be cancelled at will, even when they'd have otherwise validly proposed.

Recommendation

Resolution

Fixed in 3495

TIQV-1 | QuorumVoteTask Is Vulnerable To Certificate Forgery Attacks

Category	Severity	Location	Status
Censorship Attack, Liveness Violation	Critical	crates/task-impls/quorum_vote	Confirmed

Description

Due to the lack of validation in the `QuorumVoteTask` for the `DaCertificateRecv` event, the task is vulnerable to a certificate forgery attack in which any staked attacker may arbitrarily grief the entire network by forging a signature-valid, but content-invalid Da Certificate, and spam this message arbitrarily to all other nodes within the network. This can allow any staked attacker to DDoS the network arbitrarily, and inject a malicious payload into the other task's states, blocking voting.

Recommendation

Resolution

Fixed in 3584

O-1 | Orchestrator Slots May Be Trivially Exhausted

Category	Severity	Location	Status
Liveness Violation	High	crates/orchestrator	Confirmed

Description

Orchestrator is susceptible to resource exhaustion on startup for a malicious attacker. Within the orchestrator's `post_ready` method, there is a limited number of available slots (also, it logs using println, which is not good) and a motivated attacker with even a modest internet connection can easily absorb all of the slots as the increment for the connected nodes is unauthenticated and has no validation. This can only occur when the server is in manual mode.

Recommendation

Validate API access, especially since we know ahead of time who the keys are.

Resolution

Fixed in 3621

QPRM-1 | The Liveness Branch Of The QuorumProposalRecv Task Will Arbitrarily Block Voting

Category	Severity	Location	Status
State Violation	Medium	crates/task-impls/src/quorum_proposal_recv/mod.rs	Confirmed

Description

When the parent of a proposal does not exist in memory, but we still want to try and vote, the `QuorumProposalRecv` task initiates a lookup of the parent by calling `parent_leaf_and_state`, this lookup uses code that had originally been written for a different code path and, as a result will check if the caller is the leader since we are voting, and not forming a QC, this check does not apply to this block of code and, as a result, we may be arbitrarily blocked from voting due to this misplaced check, so the Liveness branch which initiates `VoteNow` is, essentially, non functional in its current state.

Recommendation

Change the `parent_leaf_and_state` method to assume that we're the leader of the next view, and leave the caller to verify. This is just a state lookup, so it shouldn't matter outside of potentially slightly more-frequent lock acquisition.

Resolution

Fixed in 3553

QVM-1 | VoteNow Uses An Invalid Leaf

Category	Severity	Location	Status
State Violation	Medium	crates/task-impls/src/quorum_vote/mod.rs	Confirmed

Description

The `VoteNow` task uses the wrong data in `VoteDependencyData` which uses the parent leaf instead of the proposal leaf, leading to a break in the chain in the vote for the received quorum proposal.

```
leaf = Some(vote_dependency_data.parent_leaf.clone());
```

Recommendation

Change the type to use the leaf itself, and update the comment as well. Then go into the `QuorumProposalRecv` task and update the creation of the object.

Resolution

Fixed in 3553

TIST-1 | Back-To-Back Leaders May Not Build Blocks

Category	Severity	Location	Status
State Violation	Medium	crates/task-impls/src/transactions.rs	Confirmed

Description

Within the transactions task, there is a less-than-ideally commented block of logic which is critical for gating off whether or not a block is built for a particular view. There exist two problems with this particular region of the code:

1. It is missing comments that explain the purpose of the checks and the motivation for the code as a whole.
2. Due to these checks, there is a circumstance where a leader will fail to build a block, even when they were otherwise allowed to.

Suppose we're currently in view 10, we are the leader for view 10 and view 11, and we missed view 9, meaning that self.cur_view is 8. Upon receipt of the ViewChange event, make_block is set to true, and we go on and create a block. However, on the next ViewChange event, we are also the leader, so for view 11 make_block is false, and self.cur_view is updated to view 11, so this if statement fails Because we are not the leader of the next view, but we are the leader of this view, so this check should pass and we should be able to make the block.

Recommendation

Ensure that the code is properly documented as well as allowing the leader that to build blocks in the case described above.

Resolution

Fixed in 3491

TIST-2 | Race Condition in Validated State Handling: Update vs Read Conflict

Category	Severity	Location	Status
Race Condition	Low	crates/task-impls/src/transactions.rs	Confirmed

Description

Hotshot initiates a block build request when a node is the leader of the next view. This process is handled via the `wait_for_block` function call. However, a race condition exists concerning the timing of reads and writes to the `validated_state_map`.

When `validate_proposal_safety_and_liveness` is called, it can update the `validated_state_map` before the `ViewChange` is received from the `update_view` function is processed. This results in `wait_for_block` reading the already updated value.

Alternatively, if the `ViewChange` is received before `validate_proposal_safety_and_liveness` updates the `validated_state_map`, `wait_for_block` will read a non-updated value, corresponding to the view prior to the proposed view.

Recommendation

Resolution

Fixed in 3508

HSL-1 | HotShot does not alert when running with dependency tasks

Category	Severity	Location	Status
Typo	Low	crates/hotshot/src/lib.rs:303	Confirmed

Description

During startup in HotShot, there is a log to alert the user if they're potentially unintentionally running the dependency tasks. There is a typo that was introduced when adding this log and, as a result, it will never alert the user, potentially making debugging difficult.

Recommendation

Fix the typo.

Resolution

Fixed in 3508

Global-1 | Insufficient validation in create_vote_accumulator

Category	Severity	Location	Status
State Violation	Low	Global	Confirmed

Description

When the leader of view is collecting votes to create a quorum certificate the leader will call `handle_quorum_vote_recv` each time it receives a vote, which will then handle the vote sent in until it have a threshold of votes.

When the first vote is received, the leader will create a vote accumulator task by calling `create_vote_accumulator`.

The problem is in the way we validate the first vote's view number. This is done by erroring `if vote.view_number() != info.view`. But this validation will not do anything since `info.view` will be set to `vote.view_number()`.

Recommendation

Resolution

Fixed in 3508

Disclaimer

This report is an internal review and should not be considered an “endorsement” or “disapproval” of any particular part of the codebase. It does not provide any warranty or guarantee regarding the absolute bug-free nature of the analyzed technology, nor does it reflect the economics, value, business model, or legal compliance.

This report should not be used to make investment or involvement decisions. It is not a substitute for external reviews and should not be taken as investment advice. Instead, it serves as part of an internal assessment process aimed at helping improve the quality of the code.

The goal is to help reduce attack vectors and risks associated with evolving technologies, we do not claim any guarantees regarding security or functionality of the technology analyzed. We do not guarantee the explicit security of the audited code, regardless of the findings.