SPEARBIT

Llama v1.1.0 & Llama Token Governor Security Review

Auditors

Noah Marconi, Lead Security Researcher

Xmxanuel, Security Researcher

Report prepared by: Lucas Goiriz

January 5, 2024

Contents

1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

2 Introduction

Llama is an onchain governance and access control framework for smart contracts. Using Llama, teams can deploy fully independent instances that define granular roles and permissions for executing transactions, known as "*actions*".

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of Llama according to the specific commit. Any modifications to the code will require a new security review.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 3 days in total, Llama engaged with Spearbit to review the llama-periphery protocol.

*See **Llama v1.1.0 Comments** for more details on the review scope.*

In this period of time a total of **10** issues were found.

Summary

Project Name	Llama
Repository	llama-periphery
Commit	1ae0a7...c323c2
Type of Project	Governance, Voting
Audit Timeline	Dec 18 to Dec 20
Two week fix period	

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	3	2	1
Gas Optimizations	2	1	1
Informational	5	3	2
Total	**10**	**6**	**4**

5 Findings

5.1 Low Risk

5.1.1 Unreachable branch in `ILlamaCore.actionsCount` check

Severity: Low Risk

Context: LlamaTokenGovernor.sol#L249

Description: `ILlamaCore.actionsCount` is defined as returning a `uint256`. The check performed in the `LlamaTokenGovernor.initialize` function `if (_llamaCore.actionsCount() < 0) revert InvalidLlamaCoreAddress();` will never revert with the error `InvalidLlamaCoreAddress`.

In cases where `_llamaCore` refers to an address without an `actionsCount` function, the call will revert before the `< 0` check occurs.

In cases where a value less than 0 is returned (i.e. an `int256` is returned), the bytes will be interpreted as `uint256` and fail to revert. See example in `chisel`:

```
$ chisel
Welcome to Chisel! Type `!help` to show available commands.
 int256 x = -75
 x
Type: int
 Hex: 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffb5
 Decimal: -75
 uint256(x)
Type: uint
 Hex: 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffb5
 Decimal: 115792089237316195423570985008687907853269984665640564039457584007913129639861
```

Recommendation: In discussing with the Llama team they indicate the purpose of the statement above is as a way to check that `_llamaCore` is infact a `LlamaCore` address, and suggest an an ERC165 interface for `LlamaCore` would be one solution. We agree with this remediation.

In lieu of edits to `LlamaCore`, basic address validation and duck typing are helpful:

- Zero Address Check: `if (address(_llamaCore) == address(0)) revert ZeroAddress();`.
- Duck Type: execute the call to a `LlamaCore` function anyway, if a revert occurs, it is not likely a `LlamaCore`.

Llama: We are following this recommendation and addressing it in PR 87.

Spearbit: Confirmed.

5.1.2 `delayPeriod` only works if `isFixedLengthApprovalPeriod=true` or `Governor` is the only `policyholder`

Severity: Low Risk

Context: LlamaTokenGovernor.sol#L683

Description: The active voting period in Llama is divided into three different periods in the `Governor`. The first period is the `delayPeriod`, in this period it is not possible to vote and it can be used to delegate voting power to others by calling the underlying ERC20 token. However, this concept assumes that the active voting period in `LlamaCore` has always a fixed length.

This is defined by the `isFixedLengthApprovalPeriod` in the strategy. If it is `false` the action state can be changed to `queuing` as soon as the needed majority is reached. This can lead to edge cases, where the `Governor` can't approve or disapprove an action because it is stuck in the `delayPeriod` and the action has already been executed.

If the `Governor` is the only policyholder for a specific role it does not matter, because a majority by others can not be reached earlier.

Recommendation: Ensure the `delayPeriod` is only used for fixed length approval periods or document it clearly that the `Governor` should not be used together with not fixed length approval periods.

Llama: Llama didn't implement this recommendation because the absence of the `Governor`'s vote or veto would not change the voting outcome. The action would be approved or disapproved regardless of the `Governor`'s additional voting power.

Spearbit: Acknowledged.

5.1.3 `LlamaTokenGovernor` only supports one `role`

Severity: Low Risk

Context: LlamaTokenGovernor.sol#L199

Description: The role of the `LlamaTokenGovernor` is defined in the `initialize` function. In `Llama` policyholders such as the `Governor` contract can hold `roles`, which allows them to create action or participate in voting processes.

A `policyholder` is capable of holding multiple distinct roles. However, if the `Governor` is assigned more than one role, it becomes impossible to participate in the voting processes associated with the additional roles. In addition, the stored role must be able to create actions and requires the right to `approval` or `forceApproval`. Otherwise, the `castVote`/`castVeto` function would revert to the `strategy.checkIfApprovalEnabled` call.

Recommendation: Consider allowing the role to be passed as a parameter instead of being hardcoded in storage. This would enable the Governor to vote for multiple different roles.

Llama: We are following this recommendation and addressed it in commit 2eda6549.

Spearbit: Resolved.

5.2 Gas Optimization

5.2.1 `uint16` cannot be less than 0

Severity: Gas Optimization

Context: LlamaTokenGovernor.sol#L780-L781

Description: The vote and veto quorum percentage checks compare to 0 with ensuring they are not `<= 0`.

Recommendation: Given the `uint16` type can never be below 0, the check can be simplified to:

```diff
- if (_voteQuorumPct > ONE_HUNDRED_IN_BPS || _voteQuorumPct <= 0) revert
↪  InvalidVoteQuorumPct(_voteQuorumPct);
- if (_vetoQuorumPct > ONE_HUNDRED_IN_BPS || _vetoQuorumPct <= 0) revert
↪  InvalidVetoQuorumPct(_vetoQuorumPct);
+ if (_voteQuorumPct > ONE_HUNDRED_IN_BPS || _voteQuorumPct == 0) revert
↪  InvalidVoteQuorumPct(_voteQuorumPct);
+ if (_vetoQuorumPct > ONE_HUNDRED_IN_BPS || _vetoQuorumPct == 0) revert
↪  InvalidVetoQuorumPct(_vetoQuorumPct);
```

Llama: We are following this recommendation and addressing it in PR 87.

Spearbit: Confirmed.

5.2.2 Save deploy cost and reduce maintenance overhead by using existing `Checkpoint` lib

Severity: Gas Optimization

Context: PeriodPctCheckpoints.sol, QuorumCheckpoints.sol

Description: `PeriodPctCheckpoints` and `QuorumCheckpoints` are drafted to allow for checkpointing of data structures not supported by the original OpenZeppelin Checkpoints lib. There is, however, support for Checkpoint208 as defined below:

```
struct Checkpoint208 {
    uint48 _key;
    uint208 _value;
}
```

Recommendation: Packing data before writing a checkpoint, and unpacking after reading a checkpoint, would allow for use of the unmodified Checkpoints lib:

```
function _packPeriods(
    uint16 delayPeriodPct,
    uint16 castingPeriodPct,
    uint16 submissionPeriodPct
)
    internal pure returns (uint208)
{
    return uint208(delayPeriodPct) << 32 | uint208(castingPeriodPct) << 16 |
↪   uint208(submissionPeriodPct);
}

function _unpackDelayPeriodPct(uint208 periods) internal pure returns (uint16) {
    return uint16(periods >> 32);
}

function _unpackCastingPeriodPct(uint208 periods) internal pure returns (uint16) {
    return uint16(periods >> 16);
}

function _unpackSubmissionPeriodPct(uint208 periods) internal pure returns (uint16) {
    return uint16(periods);
}
```

LLama: We have decided against implementing this to keep the checkpoints lib we're familiar with. And the gas saving is only at deploy time of the implementation (logic) contract which doesn't matter to us much anyway.

Spearbit: Acknowledged.

5.3 Informational

5.3.1 Document strategy compatibility and config requirements

Severity: Informational

Context: General scope

Description: Not all strategies and configurations are compatible with the `LlamaTokenGovernor`.

- Some strategies require that creator not be the approver.
- Strategy must have a queuing period and approval period.
- Interface must have `approvalPeriod` (not part of the core `ILlamaStrategy` interface).
- Etc...

Recommendation: Document and note the strategies and configurations the `LlamaTokenGovernor` is intended to work with and emphasize that using other configurations or strategies may lead to unexpected behavior.

Llama: We will update our documentation in the repo and in our docs to reflect these best practices.

Spearbit: Acknowledged.

5.3.2 Add NatSpec comments for parameters

Severity: Informational

Context: LlamaTokenGovernor.sol#L605

Description: Additional NatSpec comments for parameters would provide more clarity. For example:

```
/// @notice Returns the period pct checkpoints array from a given set of indices.
function getPeriodPctCheckpoints(uint256 start, uint256 end)
```

The `start` parameter here is inclusive and the `end` parameter is exclusive. This information should be ideally provided in the comments.

Recommendation: Add NatSpec comments to describe the different parameters of a function.

Llama: We are following this recommendation and addressing it in PR 87.

Spearbit: Resolved.

5.3.3 Typo in comment

Severity: Informational

Context: PeriodPctCheckpoints.sol#L85

Description:

```
  /**
   * @dev Returns whether there is a checkpoint in the structure (i.e. it is not empty), and if so the
↪  timestamp and
-  * peiod in the most recent checkpoint.
+  * period in the most recent checkpoint.
   */
```

Recommendation: Run a spellchecker within the code editor or as part of the linting.

Llama: We are following this recommendation and addressing it in PR 87.

Spearbit: Confirmed.

5.3.4 Adding safety checks for ERC20 voting token in `TokenAdapter` `initialize`

Severity: Informational

Context: LlamaTokenAdapterVotesTimestamp.sol#L54

Description: Not all ERC20 tokens can be used as voting tokens for the Governor contract.

Recommendation: Consider adding additional safety checks if the underlying ERC20 is suitable for the Governor's contract. For example, the total token supply is not higher than `type(uint128).max`.

Llama: We decided that this check wasn't necessary because the only time `uint128` is even considered is for votes/vetoes.

Spearbit: Acknowledged.

5.3.5 `submissionPeriodPct` value is not used in the `Governor` contract

Severity: Informational

Context: LlamaTokenGovernor.sol#L532

Description: The `submissionPeriodPct` can be set with the `setPeriodPct` together with `delayPeriodPct` and `castingPeriodPct` but is not used afterward.

In the `Governor` contract, the active voting period from `LlamaCore` is divided into three parts. The delay period can be used to delegate voting power, `castingPeriod` for active voting and the `submissionPeriod` to submit the voting result.

Recommendation: Consider only defining the `delayPeriodPct` and `castingPeriodPct`. The `submissionPeriodPct` would be defined implicitly as `submissionPeriodPct = ONE_HUNDRED_IN_BPS - castingPeriodPct - delayPeriodPct;`

Llama: We addressed this finding and removed the explicit reference to the submission period in commit 7534727d.

Spearbit: Resolved.

6 Llama v1.1.0 Comments

A diff review of the core Llama repo between the preview review commit hash (3c5fae52496f4342d35de6ce0cb7cf3be18281f1) and the latest (bcf23310549098c88745a1c9c352b6fe1eb0bf1b) was included. Specific focus included: LlamaGovernanceScript.sol, PR 497, PR 492, along with the `LlamaAbsoluteStrategyBase.sol` updates and associated script modifications. No findings were identified.