Security Review Report
NM-0359 Berachain-Governance

NETHERMIND
SECURITY

(Nov 13, 2024)

Contents

1 Executive Summary

This document presents the security review performed by Nethermind Security for Berachain Governance smart contracts. The Berachain team is implementing an on-chain governance system where holders of `$BGT` tokens are allowed to make important decisions about the core functions of Proof of liquidity and Berachain's core decentralized applications.

This security review focuses exclusively on the smart contracts listed in Section 2 (*Audited Files*).

The audit was performed using (a) manual analysis of the codebase and (b) creation of test cases. **Along this document, we report** 2 points of attention, where one is classified as `Low` and one is classified as `Informational` or `Best Practice`. The issues are summarized in Fig. 1.

This document is organized as follows. Section 2 presents the files in the scope. Section 3 summarizes the issues. Section 4 presents the system overview. Section 5 discusses the risk rating methodology. Section 6 details the issues. Section 7 discusses the documentation provided by the client for this audit. Section 8 presents the compilation, tests, and automated tests. Section 9 concludes the document.

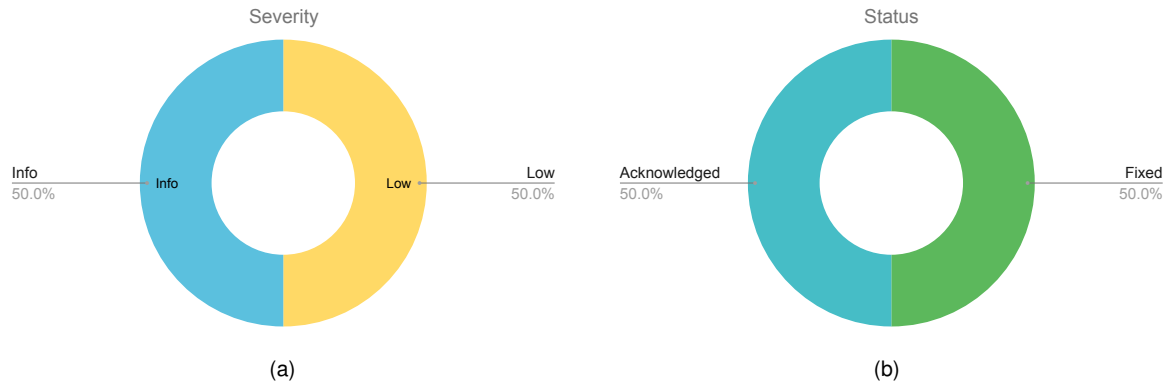


Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (1), **Undetermined** (0), **Informational** (1), **Best Practices** (0). **Distribution of status: Fixed** (1), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	Nov 4, 2024
Response from Client	Regular responses during audit engagement
Final Report	Nov 13, 2024
Repository	contracts-monorepo
Commit (Audit)	12d75bdc26195faa5b4af8cd7b995f33ef9d3a1a
Commit (Final)	30e7641819602c5ceec61e4c79043a547ab65249
Documentation Assessment	Medium
Test Suite Assessment	Medium

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	Timelock.sol	9	5	55%	4	18
2	BerachainGovernance.sol	157	24	15%	21	202
3	GovDeployer.sol	56	35	62%	16	107
	Total	**222**	**64**	**28.8%**	**41**	**327**

3 Summary of Issues

	Finding	Severity	Update
1	Voting design favors proposal execution	Low	Acknowledged
2	Incorrect status for queued accelerated proposals	Info	Fixed

4 System Overview

The Berachain team introduces an on-chain governance system where BGT token holders participate in critical decision-making processes about its core decentralized applications and the core function of its Proof of Liquidity.

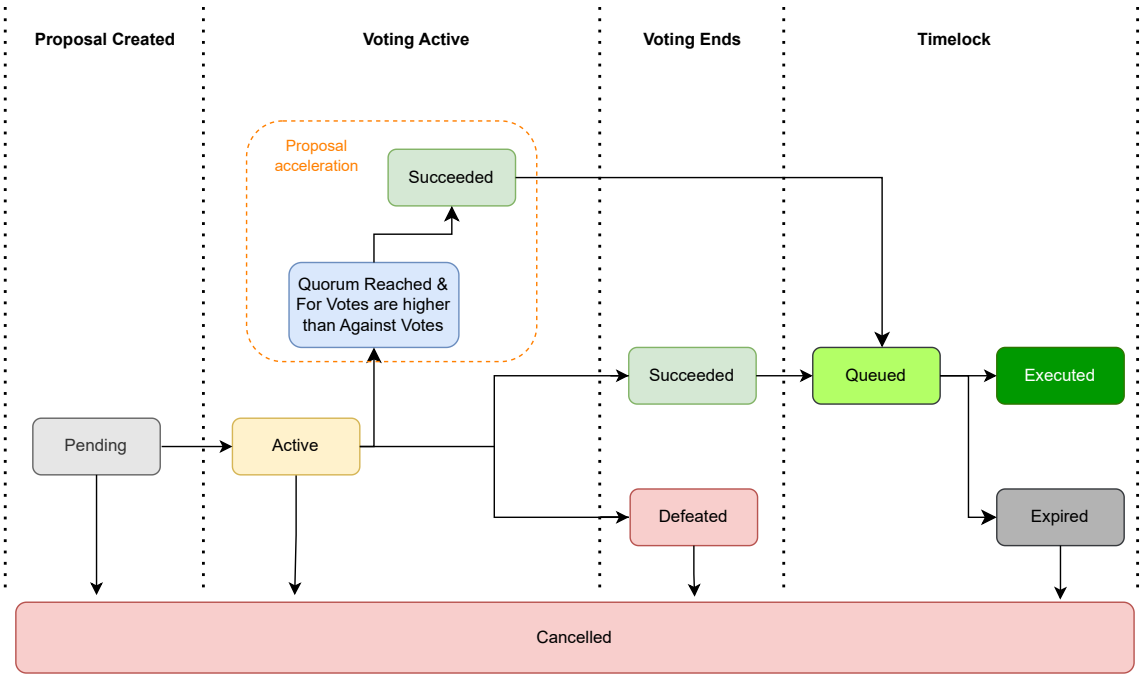
The governance module relies on Openzeppelin Governance contracts with an additional functionality to accelerate proposals execution. The system's main components include the following contracts:

- **BerachainGovernance.sol**: Main contract implementing the governance module.
- **Timelock.sol**: Inherits from OpenZeppelin `TimelockControllerUpgradeable` contract, implementing a timelock on the proposals execution.
- **GovDeployer.sol**: Helper contract to deploy the BerachainGovernance contract with pre-set parameters.

4.1 Proposal Lifecycle

There are several stages to the governance process in Berachain, including:

1. **Proposal Creation**: This is the first stage in the governance process. Any user with sufficient voting power can create a governance proposal.
2. **Pending State**: Once a proposal has been created, it enters a waiting period before it becomes active for voting.
3. **Active Voting**: When the voting period is ongoing, `$BGT` holders can cast their votes.
4. **Proposal Outcome**: The proposal is marked as `Succeeded` as soon as the quorum is reached and votes `for` the proposal are higher than those `against`. This acceleration process implemented by Berachain team, allows the early execution of proposals even when the voting period is not over yet.
5. **Timelock**: If the proposal succeeds, it is queued with a timelock delay.
6. **Execution**: Following the expiry of the timelock period, the proposal can be executed, implementing the changes described in the proposal.



4.2 Governance Deployment

Berachain utilizes the `GovDeployer` contract to deploy governance contracts with configurable parameters. This is aimed at preventing front-running related issues. The deployment process uses `CREATE2` to deploy `Timelock` and `BerachainGovernance` contracts deterministically.

Additionally, during deployment, initial contract parameters are set, including the `GOV_PROPOSAL_THRESHOLD`, voting delay, and quorum requirements.

5 Risk Rating Methodology

The risk rating methodology used by Nethermind Security follows the principles established by the OWASP Foundation. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely the finding is to be uncovered and exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage, such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage, such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage, such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding, other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind Security also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to pass to the client formally;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

6 Issues

6.1 [Low] Voting design favors proposal execution

File(s): BerachainGovernance.sol

Description: The BerachainGovernance contract modifies the behavior of the state(...) function to introduce a mechanism for accelerating proposal execution upon a successful vote. A proposal is considered successful if two key conditions are met, even if the voting delay is not over:

- **Quorum is reached**: The total number of voters, including both *for* and *abstain* votes, exceeds a defined threshold.
- **The vote is successful**: The count of *for* votes is higher than the votes *against*.

However, this approach reveals drawbacks that need careful consideration, particularly given the potential implications for important governance actions.

- The modification creates a race condition as it incentivizes certain actors to vote early, especially those in favor of a proposal, to ensure that a quorum is reached before opposition votes can be cast.
- Participants are not given equal opportunities to vote. Those who are able to vote earlier hold more influence over the outcome, while some participants may miss their chance to vote due to the acceleration process.
- The results of a vote under the current design may diverge from its results if the voting period is completed. If users in favor of the proposal vote first, potentially front-running the votes against the proposal, the eventual outcome might not accurately reflect the community's consensus.

Recommendation(s): Revaluate the voting mechanism to address the mentioned drawbacks and ensure a fair process.

Status: Acknowledged

Update from the client:

6.2 [Info] Incorrect status for queued accelerated proposals

File(s): BerachainGovernance.sol

Description: The BerachainGovernance contract overrides the state(...) function to implement additional logic, allowing for proposal acceleration when the quorum is reached. In such cases, the state is set to Succeeded regardless of whether the voting period has ended.

However, this function does not check if the proposal was queued in the timelock contract. Consequently, an erroneous status of Succeeded might be returned when the proposal is actually Queued.

```
function state(uint256 proposalId)
    public
    view
    override(GovernorUpgradeable, GovernorTimelockControlUpgradeable)
    returns (ProposalState)
{
    ProposalState currentState = GovernorTimelockControlUpgradeable.state(proposalId);

    // Accelerate the proposal if it has reached the quorum; regardless of the deadline.
    if (currentState == ProposalState.Active && _quorumReached(proposalId) && _voteSucceeded(proposalId)) {
        return ProposalState.Succeeded;
    }

    return currentState;
}
```

Recommendation(s): To ensure an accurate state returned by the function, check for the queued state of the proposal.

Status: Fixed

Update from the client: Fixed with 30e7641819602c5ceec61e4c79043a547ab65249

7 Documentation Evaluation

Software documentation refers to the written or visual information that describes the functionality, architecture, design, and implementation of software. It provides a comprehensive overview of the software system and helps users, developers, and stakeholders understand how the software works, how to use it, and how to maintain it. Software documentation can take different forms, such as user manuals, system manuals, technical specifications, requirements documents, design documents, and code comments. Software documentation is critical in software development, enabling effective communication between developers, testers, users, and other stakeholders. It helps to ensure that everyone involved in the development process has a shared understanding of the software system and its functionality. Moreover, software documentation can improve software maintenance by providing a clear and complete understanding of the software system, making it easier for developers to maintain, modify, and update the software over time. Smart contracts can use various types of software documentation. Some of the most common types include:

- Technical whitepaper: A technical whitepaper is a comprehensive document describing the smart contract's design and technical details. It includes information about the purpose of the contract, its architecture, its components, and how they interact with each other;
- User manual: A user manual is a document that provides information about how to use the smart contract. It includes step-by-step instructions on how to perform various tasks and explains the different features and functionalities of the contract;
- Code documentation: Code documentation is a document that provides details about the code of the smart contract. It includes information about the functions, variables, and classes used in the code, as well as explanations of how they work;
- API documentation: API documentation is a document that provides information about the API (Application Programming Interface) of the smart contract. It includes details about the methods, parameters, and responses that can be used to interact with the contract;
- Testing documentation: Testing documentation is a document that provides information about how the smart contract was tested. It includes details about the test cases that were used, the results of the tests, and any issues that were identified during testing;
- Audit documentation: Audit documentation includes reports, notes, and other materials related to the security audit of the smart contract. This type of documentation is critical in ensuring that the smart contract is secure and free from vulnerabilities.

These types of documentation are essential for smart contract development and maintenance. They help ensure that the contract is properly designed, implemented, and tested, and they provide a reference for developers who need to modify or maintain the contract in the future.

Remarks about Berachain documentation

The Berachain team provided comprehensive documentation outlining the protocol's functionalities. Additionally, they effectively addressed concerns and questions raised by the Nethermind Security team during their regular calls.

8 Test Suite Evaluation

8.1 Compilation Output

```
> forge build

[] Compiling...
[] Compiling 31 files with Solc 0.8.26
[] Solc 0.8.26 finished in 12.51s
Compiler run successful!
```

8.2 Tests Output

```
> forge test

Ran 27 tests for test/gov/BerachainGovernance.t.sol:BerachainGovernanceTest
[PASS] testFuzz_UpgradeGovernance_FailsIfNotOwner(address) (runs: 1024, : 5795323, ~: 5795323)
[PASS] testFuzz_UpgradeTimelock_FailsIfNotOwner(address) (runs: 1024, : 2334611, ~: 2334611)
[PASS] test_AcceleratedProposal() (gas: 976936)
[PASS] test_CancelProposal() (gas: 463168)
[PASS] test_CancelProposalFailsIfCallerNotProposer() (gas: 453725)
[PASS] test_CancelProposalFailsIfProposalActive() (gas: 469939)
[PASS] test_CreateFailedProposal() (gas: 523932)
[PASS] test_CreatePendingProposal() (gas: 471557)
[PASS] test_CreateSucceededProposal() (gas: 526475)
[PASS] test_ExecuteProposal() (gas: 738060)
[PASS] test_ExecuteProposal_FailsIfProposalNonExistent() (gas: 55312)
[PASS] test_ExecuteProposal_FailsIfProposalNotQueued() (gas: 452179)
[PASS] test_ExecuteProposal_FailsIfSucceededButNotQueued() (gas: 574155)
[PASS] test_GuardianCanCancelProposalWhenReady() (gas: 675824)
[PASS] test_GuardianCanCancelProposalWhenWaiting() (gas: 675293)
[PASS] test_GuardianCancelProposalFailsIfAddressDoesNotHaveRole() (gas: 661040)
[PASS] test_ProposeFailsIfInsufficienVotes() (gas: 65656)
[PASS] test_ProposeFailsWithInvalidDescription() (gas: 56398)
[PASS] test_QueueProposal() (gas: 649555)
[PASS] test_QueueProposalFailsIfProposalCanceled() (gas: 468268)
[PASS] test_QueueProposalFailsIfProposalNonExistent() (gas: 55167)
[PASS] test_QueueProposalFailsIfProposalNotSuccess() (gas: 452109)
[PASS] test_UpgradeGovernanceViaVoting() (gas: 6330371)
[PASS] test_UpgradeGovernance_FailsIfNotOwner() (gas: 5795189)
[PASS] test_UpgradeTimelockViaVoting() (gas: 2850600)
[PASS] test_UpgradeTimelockWithCallerAsTimelock() (gas: 2339925)
[PASS] test_VoteCastingFailsIfMissingVotingPower() (gas: 482365)
Suite result: ok. 27 passed; 0 failed; 0 skipped; finished in 264.48ms (259.90ms CPU time)
```

9 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation.

We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client. Written in Golang and open-sourced from the outset, Juno verifies the validity of the data received from Starknet by comparing it to proofs retrieved from Ethereum, thus maintaining the integrity and security of the entire ecosystem.

Learn more about us at nethermind.io.

General Advisory to Clients

As auditors, we recommend that any changes or updates made to the audited codebase undergo a re-audit or security review to address potential vulnerabilities or risks introduced by the modifications. By conducting a re-audit or security review of the modified codebase, you can significantly enhance the overall security of your system and reduce the likelihood of exploitation. However, we do not possess the authority or right to impose obligations or restrictions on our clients regarding codebase updates, modifications, or subsequent audits. Accordingly, the decision to seek a re-audit or security review lies solely with you.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.