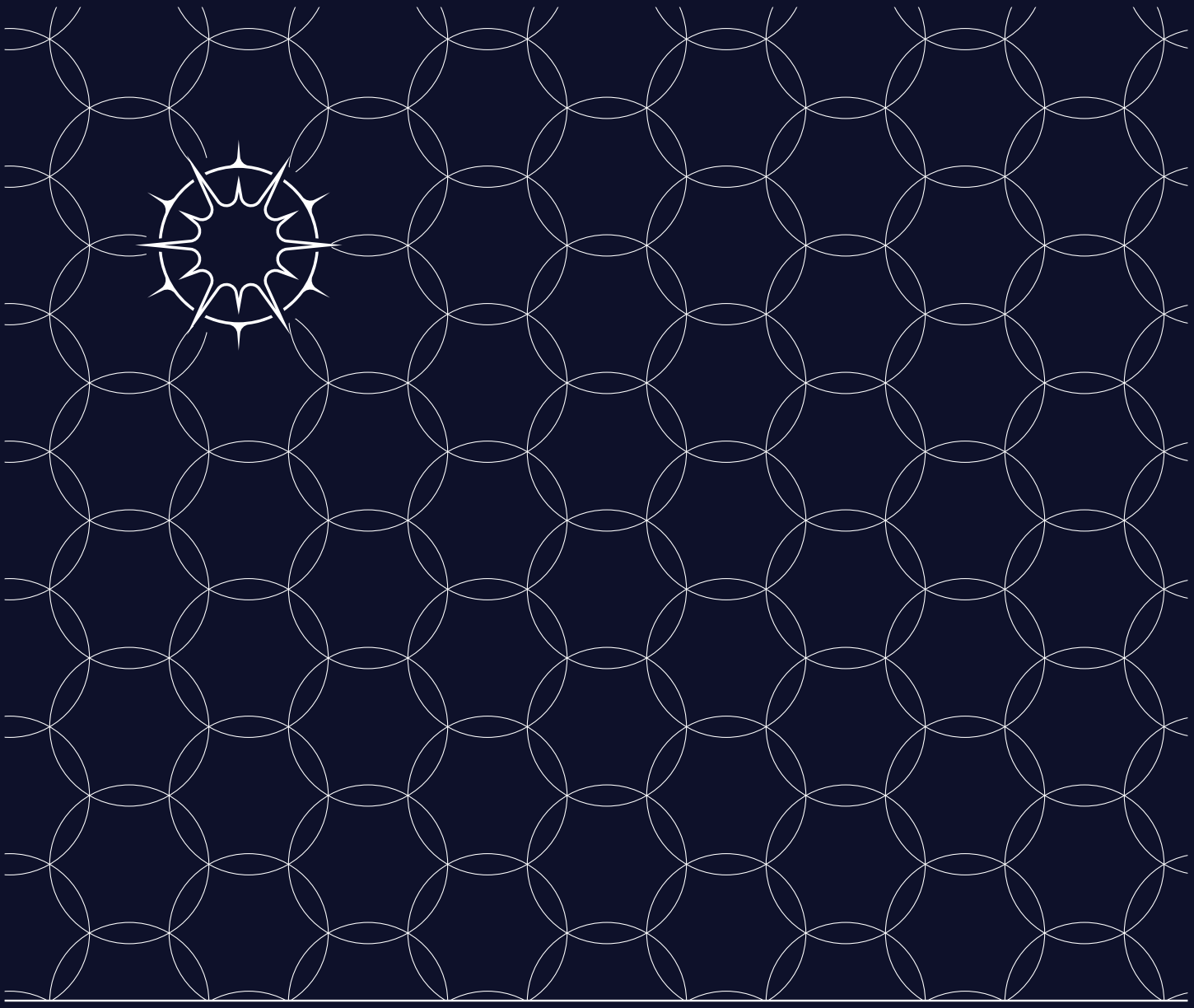

28.12.2024 **OP Labs**

DeputyGuardianModule Security Review

Version 1.0

Auditor:

EV_om

Table of Contents

1 Introduction

1.1 About OP Labs

OP Labs contributes to the Optimism protocol, an extension to Ethereum that scales both its technology and values. Optimism enables orders of magnitude of improved performance and scalability to Ethereum while doubling down on its commitment to public goods.

1.2 About Radiant Labs

Radiant Labs is a smart contract security firm providing bespoke security assessments and code reviews. Our team of specialized auditors combines deep technical expertise with practical blockchain security experience to deliver meticulously tailored security reviews.

1.3 About the Auditor

EV_om is an independent smart contract security researcher specialising in smart contract audits. They currently serve as a Zenith Researcher and Judge on Code4rena, where they ranked #8 overall as a solo researcher and led Radiant Labs to the #2 position in 2024. Their expertise has been demonstrated through consistent top performance in competitive security reviews across a diverse range of major blockchain protocols.

2 Engagement Details

2.1 Executive Summary

Radiant Labs conducted a security assessment of the DeputyPauseModule smart contract for OP Labs. The module enables authorised emergency pauses of the Optimism network through a signature-based mechanism, replacing the existing pre-signed transaction system.

The assessment found the module to be well-designed, with identified issues focusing on standard compliance and documentation accuracy. None of the findings compromise the module's core security properties or its ability to serve its intended purpose.

2.2 Scope

Overview

Project Name	Optimism DeputyPauseModule
Repository	https://github.com/ethereum-optimism/optimism
Commit hash	2f17e6b67c61de5d8073d556272796d201bc740b
Methods	Manual review
Duration	2 days

Contracts in Scope

packages/contracts-bedrock/src/safe/DeputyPauseModule.sol

Supporting documentation:

- Design document
- Component specification with system invariants
- Implementation tests

Audit goals:

- Identify any lingering security risks not already explicitly stated or understood
- Verify adherence to system invariants specified in the component specification
- As an additional measure, evaluated the module's interaction with existing governance infrastructure to ensure secure privilege flows and component boundaries

2.3 Risk Classification

Findings in this report are classified according to their severity:

High - direct theft/loss/freezing of funds, unauthorised control over critical functions, or breaking of core contract functionality
Medium - economic damage or manipulation under specific conditions or with limited impact
Low - non-critical implementation issues with negligible security impact
Informational - style suggestions, documentation improvements, and recommended best practices

3 Findings

Issues Found

High risk	0
Medium risk	0
Low risk	2
Informational	2

3.1 Low risk

3.1.1 Struct encoding does not follow EIP-712

Context: `DeputyPauseModule.sol`

Description: The contract currently encodes the struct data incorrectly in both the `pause()` and `_setDeputy()` functions. According to EIP-712, the 'data to sign' denoted by `hashStruct` should be computed as:

```
hashStruct(s : S) = keccak256(typeHash || encodeData(s))
```

...where `encodeData(s)` is the concatenation of the encoded member values in their order of appearance, each exactly 32 bytes long. However, the current implementation includes the struct itself in the encoding:

```
File: DeputyPauseModule.sol
164:     bytes32 digest = _hashTypedDataV4(keccak256(abi.encode(
    PAUSE_MESSAGE_TYPEHASH, PauseMessage(_nonce))));
```

```
File: DeputyPauseModule.sol
201:     bytes32 digest =
202:         _hashTypedDataV4(keccak256(abi.encode(DEPUTY_AUTH_MESSAGE_TYPEHASH,
    DeputyAuthMessage(_deputy))));
```

While this does not currently affect the resulting hash due to the simple nature of the structs (single-field structs with primitive types), it would lead to incorrect signature verification if the structs were to be extended with more complex data types in the future.

Recommendation: Remove the struct wrapper and directly encode the field values:

```
File: DeputyPauseModule.sol
164:     bytes32 digest = _hashTypedDataV4(keccak256(abi.encode(
    PAUSE_MESSAGE_TYPEHASH, _nonce)));
...
202:         _hashTypedDataV4(keccak256(abi.encode(DEPUTY_AUTH_MESSAGE_TYPEHASH,
    _deputy)));
```

3.1.2 Deputy key rotation process can be optimised for faster incident response

Context: `DeputyPauseModule.sol`

Description: The current design documentation suggests that in case of a deputy key compromise, the response would be to:

1. Deploy a new `DeputyPauseModule`
2. Configure a new deputy
3. Grant appropriate permissions

However, the module already implements `setDeputy()` which allows the Foundation Safe to directly rotate the deputy key with a new address and signature. This simpler approach aligns with the specification's invariant iDPM-005 which states that "the Foundation Safe must be able to change the Deputy account easily."

Using `setDeputy()` would provide a faster incident response path compared to deploying a new module, as it requires fewer steps while maintaining the same security properties.

Recommendation:
Update the design documentation to reflect `setDeputy()` as the primary mitigation strategy for deputy key compromise.
To further optimize the response time, consider:

- Pre-signing and testing a `DeputyAuthMessage` from a backup deputy address that can be immediately used if the primary deputy is compromised
- Adding an authenticated method to temporarily disable the current deputy while a new one is being configured, providing an intermediate safety state during rotation

3.2 Informational

3.2.1 Outdated documentation on privileged roles

Description: The documentation under docs.optimism.io/chain/security/privileged-roles contains outdated information regarding roles and addresses.

The listed roles and addresses in the documentation do not reflect the current state of the system, which could lead to confusion during future maintenance or security reviews.

For example:

- The System Config Owner address has changed and its description is outdated
- The description of the Guardian contract is outdated with the introduction of the Deputy Pause Module

Recommendation:

- Update the documentation with current role definitions
- Review and update all listed addresses
- Implement a regular documentation review process

3.2.2 Incorrect module installation description

Description: Documentation incorrectly describes the DeputyGuardianModule's installation location within the governance structure.

The Deputy Pause Module design document suggests that the DeputyGuardianModule is to be installed on the Security Council Safe, when it is actually designed to be installed on the Guardian which the Security Council controls.

Recommendation: Clarify the correct installation location. Additionally, consider documenting and maintaining an overview of the overall structure of governance and permissioned addresses e.g. with the help of a diagram.

4 Appendix

4.1 Methodology

Our manual review methodology emphasizes deep technical understanding of the target system. Initial documentation analysis guides a comprehensive code review process that examines both individual components and system-wide interactions. The review process progresses through multiple phases of increasing depth, tracking coverage while maintaining flexibility to investigate emerging concerns. We investigate unfamiliar patterns through reference implementations and technical literature, ensuring thorough comprehension of all security-relevant aspects.

4.2 Disclaimer

This report is not an endorsement or indictment of any particular project or team, and the report does not guarantee the security of any particular project. This report represents our best effort to identify potential security issues within the smart contracts based on the information available at their time of writing. The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status.