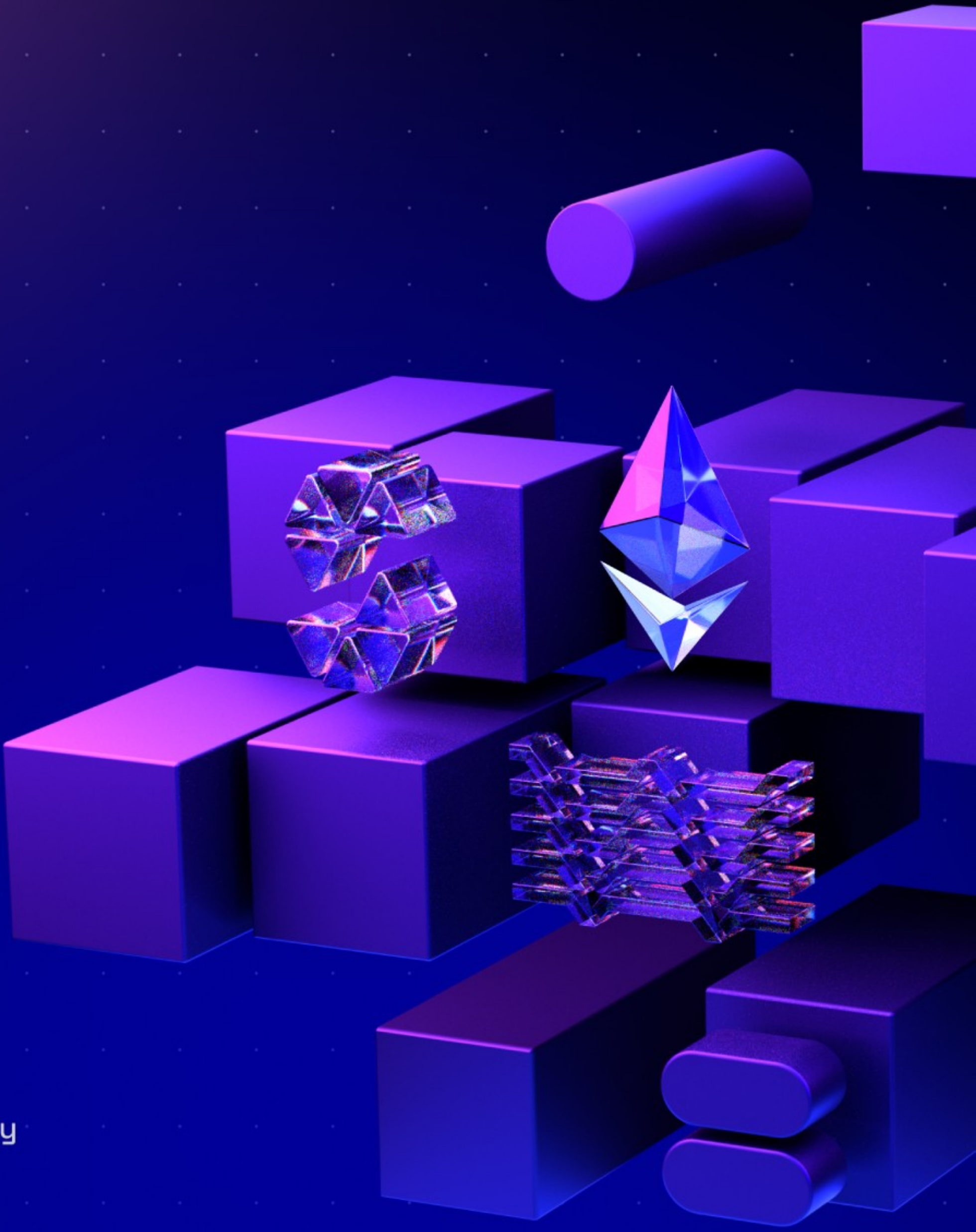
Safe

Allowance module

9.9.2024



Ackee Blockchain Security

Contents

1. Document Revisions

1.0	Final Report	05.09.2024
1.1	Fix Review	09.09.2024

2. Overview

This document presents our findings in reviewed contracts.

2.1. Ackee Blockchain Security

Ackee Blockchain Security is an auditing company based in Prague, Czech Republic, specializing in audits and security assessments. Our mission is to build a stronger blockchain community by sharing knowledge – we run free certification courses School of Solana, Summer School of Solidity and teach at the Czech Technical University in Prague. Ackee Blockchain Security is backed by the largest VC fund focused on blockchain and DeFi in Europe, RockawayX.

2.2. Audit Methodology

1. **Technical specification/documentation** - a brief overview of the system is requested from the client and the scope of the audit is defined.
2. **Tool-based analysis** - deep check with automated Solidity analysis tools and Wake is performed.
3. **Manual code review** - the code is checked line by line for common vulnerabilities, code duplication, best practices and the code architecture is reviewed.
4. **Local deployment + hacking** - the contracts are deployed locally and we try to attack the system and break it.
5. **Unit and fuzz testing** - run unit tests to ensure that the system works as expected, potentially write missing unit or fuzz tests.

2.3. Finding classification

A *Severity* rating of each finding is determined as a synthesis of two sub-ratings: *Impact* and *Likelihood*. It ranges from *Informational* to *Critical*.

If we have found a scenario in which an issue is exploitable, it will be assigned an impact rating of *High*, *Medium*, or *Low*, based on the direness of the consequences it has on the system. If we haven't found a way, or the issue is only exploitable given a change in configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.) or given a change in the codebase, then it will be assigned an impact rating of *Warning* or *Info*.

Low to *High* impact issues also have a *Likelihood*, which measures the probability of exploitability during runtime.

The full definitions are as follows:

Severity

		Likelihood			
		High	**Medium**	**Low**	**N/A**
Impact	**High**	Critical	High	Medium	-
	Medium	High	Medium	Low	-
	Low	Medium	Low	Low	-
	Warning	-	-	-	Warning
	Info	-	-	-	Info

Table 1. Severity of findings

Impact

- **High** - Code that activates the issue will lead to undefined or catastrophic consequences for the system.
- **Medium** - Code that activates the issue will result in consequences of serious substance.
- **Low** - Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.
- **Warning** - The issue cannot be exploited given the current code and/or configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.), but could be a security vulnerability if these were to change slightly. If we haven't found a way to exploit the issue given the time constraints, it might be marked as a "Warning" or higher, based on our best estimate of whether it is currently exploitable.
- **Info** - The issue is on the borderline between code quality and security. Examples include insufficient logging for critical operations. Another example is that the issue would be security-related if code or configuration (see above) was to change.

Likelihood

- **High** - The issue is exploitable by virtually anyone under virtually any circumstance.
- **Medium** - Exploiting the issue currently requires non-trivial preconditions.
- **Low** - Exploiting the issue requires strict preconditions.

2.4. Review team

The following table lists all contributors to this report. For authors of the specific revision, see the “Revision team” section in the respective “Report revision” chapter.

Member’s Name	Position
Jan Kalivoda	Lead Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

2.5. Disclaimer

We’ve put our best effort to find all vulnerabilities in the system, however our findings shouldn’t be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

3. Executive Summary

Safe is a smart account that allows to enable various modules. The Allowance module manages transfer allowances for any tokens.

Revision 1.0

Safe engaged Ackee Blockchain Security to perform a security review of the changes to the Allowance module with a total time donation of 1 engineering day in a period between September 4 and September 5, 2024, with Jan Kalivoda as the lead auditor.

The incremental audit was performed on the commit `bc76ff`[1] and the scope was the changed `ALLOWANCE_TRANSFER_TYPEHASH`.

During the review, we paid special attention to:

- change of the typehash doesn't affect negatively the module's functionality and security,
- past audit report for the fork of the module to find applicable issues to the current scope.

Our review resulted in 1 findings.

Ackee Blockchain Security recommends Safe:

- address all reported issues.

See Report revision 1.0 for the system overview and trust model.

Revision 1.1

The fix review was done on the given commit `549ea1`[2]. All of the findings from the previous revision were fixed.

[1] full commit hash: `bc76ffd27a8a0c15c255ffcb61707bf7a30085eb`

[2] full commit hash: `549ea1605538a4aa0571fa8e4d44e0e762e60720`

4. Summary of Findings

The following table summarizes the findings we identified during our review. Unless overridden for purposes of readability, each finding contains:

- a *Description*,
- an *Exploit scenario*,
- a *Recommendation* and if applicable
- a *Fix*.

There might often be multiple ways to solve or alleviate the issue, with varying requirements regarding the necessary changes to the codebase. In that case, we will try to enumerate them all, clarifying which solves the underlying issue better (albeit possibly only with architectural changes) than others.

Critical	High	Medium	Low	Warning	Info	Total
0	0	0	1	0	0	1

Table 2. Findings Count by Severity

Finding title	Severity	Reported	Status
L1: Possible division by zero in modulo operation	Low	1.0	Fixed

Table 3. Table of Findings

Report revision 1.0

Revision team

Member's Name	Position
Jan Kalivoda	Lead Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

System Overview

The contract is a Safe module that manages transfer allowances for any tokens.

Trust Model

The contract can be considered trustless.

L1: Possible division by zero in modulo operation

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	AllowanceModule.sol	Type:	Arithmetics

Description

It is possible to set the `resetTimeMin` to 0. When the `resetBaseMin` is bigger than 0 and the `resetTimeMin` is 0, then the branch with modulo operation is triggered and it will cause a division by zero. Since it is also in Solidity version <0.8.0, it consumes all gas.

Listing 1. Excerpt from AllowanceModule

```
function setAllowance(address delegate, address token, uint96
allowanceAmount, uint16 resetTimeMin, uint32 resetBaseMin) public {
    require(delegate != address(0), "delegate != address(0)");
    require(
        delegates[msg.sender][uint48(delegate)].delegate == delegate,
        "delegates[msg.sender][uint48(delegate)].delegate == delegate"
    );
    Allowance memory allowance = getAllowance(msg.sender, delegate, token);
    if (allowance.nonce == 0) {
        // New token
        // Nonce should never be 0 once allowance has been activated
        allowance.nonce = 1;
        tokens[msg.sender][delegate].push(token);
    }
    // Divide by 60 to get current time in minutes
    // solium-disable-next-line security/no-block-members
    uint32 currentMin = uint32(block.timestamp / 60);
    if (resetBaseMin > 0) {
        require(resetBaseMin <= currentMin, "resetBaseMin <= currentMin");
        allowance.lastResetMin = currentMin - ((currentMin - resetBaseMin) %
resetTimeMin);
    } else if (allowance.lastResetMin == 0) {
```

Exploit scenario

The The `setAllowance` function is called wit the following parameters and unexpectedly reverts.

```
module.setAllowance(
    delegate=delegate_address,
    token=token_address,
    allowanceAmount=10,
    resetTimeMin=0,
    resetBaseMin=1,
)
```

Recommendation

Add check for `resetTimeMin` to be bigger than 0.

Fix 1.1

The issue is fixed by adding the data validation.

Go back to Findings Summary

Appendix A: How to cite

Please cite this document as:

Ackee Blockchain Security, Safe: Allowance module, 9.9.2024.

Thank You

Ackee Blockchain a.s.

Rohanske nabrezi 717/4
186 00 Prague
Czech Republic

hello@ackee.xyz