decurity

Smart Contract Security Audit Report

Socket (Mayan Integration)

1. Contents

2. General Information

This report contains information about the results of the security audit of the Socket.Tech (hereafter referred to as "Customer") Mayan route smart contract, conducted by Decurity in the period from 01/09/2025 to 01/10/2024.

2.1. Introduction

Tasks solved during the work are:

- Review the protocol design and the usage of 3rd party dependencies,
- Audit the contracts implementation,
- Develop the recommendations and suggestions to improve the security of the contracts.

2.2. Scope of Work

The audit scope included the following smart contracts (commitcafe429c105670dca112b0e9aab7e1ab5db537a6):

- https://github.com/SocketDotTech/socket-ll-contracts/pull/257

2.3. Threat Model

The assessment presumes the actions of an intruder who might have the capabilities of any role (an external user, token owner, token service owner, or a contract).

The main possible threat actors are:

- User,
- Protocol owner,
- Relayer,
- Token owner/contract.

2.4. Weakness Scoring

An expert evaluation scores the findings in this report, and the impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2.5. Disclaimer

Due to the intrinsic nature of the software and vulnerabilities and the changing threat landscape, it cannot be generally guaranteed that a certain security property of a program holds.

Therefore, this report is provided “as is” and is not a guarantee that the analyzed system does not contain any other security weaknesses or vulnerabilities. Furthermore, this report is not an endorsement of the Customer’s project, nor is it an investment advice.

That being said, Decurity exercises the best effort to perform its contractual obligations and follow the industry methodologies to discover as many weaknesses as possible and maximize the audit coverage using limited resources.

3. Summary

As a result of this work, we haven’t discovered any exploitable security issues.

The other suggestions included fixing the low-risk issues and some best practices (see Security Process Improvement).

3.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of 14 January, 2025.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
Incorrect calldata parsing in `replaceMiddleAmount`	src/bridges/mayan/MayanBridge.sol	**High**	Acknowledged
Wrong events could be emitted	src/bridges/mayan/MayanBridge.sol	**Low**	Acknowledged

4. General Recommendations

This section contains general recommendations on how to improve the overall security level.

The Findings section contains technical recommendations for each discovered issue.

4.1. Security Process Improvement

The following is a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level:

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new contracts and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the contracts.

5. Findings

5.1. Incorrect calldata parsing in `replaceMiddleAmount`

Risk Level: High

Status: Comment from the developer: The data is coming from their SDK, so it won't be malformed. Plus, we have added checks on the backend so that it does not have the issue.

Contracts:

- src/bridges/mayan/MayanBridge.sol

Description:

`replaceMiddleAmount` doesn't account for all the calldata variants. For example, calldata from the test `test/solidity/bridges/mayan/PolygonNativeToSolanaSOL.t.sol` will be processed incorrectly when `bridgeAfterSwap` or `swapAndBridge` is called.

Currently, the test code directly calls `bridgeNativeTo`, so `replaceMiddleAmount` and `replaceMiddleAmountMemory` are not invoked.

Remediation:

Limit the function selectors allowed to call and make sure calldata processing is done correctly for each of them. For example, `wrapAndSwapETH` currently will not work (see https://github.com/mayan-finance/swap-bridge/blob/4277ba9275d7911ffeb73a8eca6fc8c8718d145b/src/MayanSwap.sol#L111).

5.2. Wrong events could be emitted

Risk Level: Low

Status: Comment from the developer: we verify the receiver on the backend for the emitted one.

Contracts:

- src/bridges/mayan/MayanBridge.sol

Description:

In the emitted events, the recipient address is taken from `mayanBridgeData.nonEvmAddress` or `mayanBridgeData.receiver`. However, the actual recipient address stored in `mayanBridgeData.protocolData` might be different. There are no checks for this discrepancy.

Remediation:

Consider parsing the calldata to fill in the event data.

6. Appendix

6.1. About us

The Decurity team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we’ve gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.