Smart Contract Security Audit Report

Socket (Across Bridge 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”) Across Bridge route smart contract, conducted by Decurity in the period from 05/15/2024 to 05/16/2024.

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 contract https://github.com/SocketDotTech/socket-ll-contracts/blob/feat/across-v3-stack-fix/src/bridges/across/AcrossV3.sol. Initial review was done for the commit cd38dd980514c5e5b00e3e212a33bc2f2164f49f. The remediation review was done for the commit bf0e69f83c074b38791d849ead25064710f27c02

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 20 May, 2024.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
User cannot send ETH to a smart contract	src/bridges/across/AcrossV3.sol	**Info**	Fixed
Wrong or unused comments	src/bridges/across/AcrossV3.sol	**Info**	Fixed
UINT256_MAX can be private	src/bridges/across/AcrossV3.sol	**Info**	Fixed

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. Users cannot send ETH to a smart contract

Risk Level: Info

Status: Fixed in the commit bf0e69f8.

Contracts:

- src/bridges/across/AcrossV3.sol

Description:

According to the Across documentation, if a bridge transfer is sent to the contract, the contract will receive WETH (not ETH). Otherwise, if a bridge transfer is being sent to an EOA, the EOA will receive ETH (not WETH).

A user might think that the output token will be ETH and try to send it to a contract, which hasn't possibility to withdraw WETH. As a result, WETH will be deposited to the contract and locked.

It is important to note this behavior in the documentation to avoid such locked funds.

Remediation:

Note in the documentation the fact about WETH/ETH dependance on the output address type.

References:

- https://docs.across.to/introduction/developer-notes#what-is-the-behavior-of-eth-weth-in-transfers

5.2. Wrong or unused comments

Risk Level: Info

Status: Fixed in the commit f1b365a1.

Contracts:

- src/bridges/across/AcrossV3.sol

Description:

The line 64 of the contract `AcrossImpl` contains wrong comment. The structure `AcrossBridgeDataNoToken` has the variable `address outputToken`, however, the comment assumes that the variable is an array, like in the structure `AcrossBridgeData`.

In addition, the line `9` in the contract file contains commented import of the `console.sol` contract, which is better to be removed.

Remediation:

Fix and remove unused comments.

5.3. UINT256_MAX can be private

Risk Level: **Info**

Status: Fixed in the commit f1b365a1.

Contracts:

- src/bridges/across/AcrossV3.sol

Location: Lines: 25.

Description:

The variable `uint256 public immutable UINT256_MAX` can be private to save gas for the contract deployment.

Remediation:

Make the variable private.

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.