Smart Contract Security Audit Report

Socket (Stargate v2 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") Stargate v2 route smart contract, conducted by Decurity in the period from 06/03/2024 to 06/04/2024.

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 (commit 5d417ae21f385423777234ae8365cfab58b6d3cc):

- https://github.com/SocketDotTech/socket-ll-contracts/blob/feat/stargate-v2/src/bridges/stargate/StargateV2.sol
- https://github.com/SocketDotTech/socket-ll-contracts/blob/feat/stargate-v2/src/receivers/stargate.sol

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 23 August, 2024.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
Lack of events emitting in swapAndBridge	src/bridges/stargate/StargateV2.sol	**Low**	Fixed
Incorrect use of safeDecreaseAllowance	src/receivers/stargate.sol	**Info**	Fixed
Absence of selectors	src/bridges/stargate/StargateV2.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. Lack of events emitting in swapAndBridge

Risk Level: **Low**

Status: Fixed in commit bf1bc3.

Contracts:

- src/bridges/stargate/StargateV2.sol

Location: Lines: `251`. Function: `swapAndBridge`.

Description:

The `swapAndBridge` function does not emit `SocketBridge` event, like other bridge implementations. In plus, it does not emits `NativeBridgeFee` event, like other functions of this contracts.

As a result, the analytical tools may report wrong information. Note that these events do not affect bridging.

Remediation:

Consider adding event emitting.

5.2. Incorrect use of safeDecreaseAllowance

Risk Level: **Info**

Status: Fixed in commit bf1bc3.

Contracts:

- src/receivers/stargate.sol

Location: Lines: `161-165`. Function: `perfomAction`.

Description:

It seems like the `performAction` function is supposed to give a temporary allowance to the `toAddress` recipient on lines 161-165, but it is implemented incorrectly.

```
File: stargate.sol
148:     function perfomAction(
             address token,
             uint256 amountLD,
             address payable toAddress,
             bytes memory dataPayload
         ) private notBlocked(toAddress) {
             if (token == NATIVE_TOKEN_ADDRESS) {
                 (bool success, ) = toAddress.call{
                     gas: gasleft() - defaultGas,
                     value: amountLD
                 }(dataPayload);
                 require(success, "StargateReceiver: Failed to call");
             } else {
                 IERC20(token).safeIncreaseAllowance(toAddress, amountLD);
                 (bool success, ) = toAddress.call{gas: gasleft() -
defaultGas}(
                     dataPayload
                 );
                 IERC20(token).safeDecreaseAllowance(toAddress, 0);
                 require(success, "StargateReceiver: Failed to call");
             }
             emit PayloadExecuted(toAddress, amountLD, token);
         }
```

Instead of setting the allowance to zero, the function call `safeDecreaseAllowance(toAddress, 0)` is simply maintaining the current allowance, as it sets the allowance to `oldAllowance - 0`.

```
    function safeDecreaseAllowance(
        IERC20 token,
        address spender,
        uint256 value
    ) internal {
        unchecked {
            uint256 oldAllowance = token.allowance(address(this), spender);
            require(oldAllowance >= value, "SafeERC20: decreased allowance
below zero");
            uint256 newAllowance = oldAllowance - value;
            _callOptionalReturn(token,
abi.encodeWithSelector(token.approve.selector, spender, newAllowance));
        }
    }
```

As a result, if the allowance is not fully spent by `toAddress`, it remains after the action execution.

However, this logic only gives `toAddress` the right to take back unspent tokens from the Stargate contract.

Remediation:

Consider fixing the allowance decrease:

```
-165:               IERC20(token).safeDecreaseAllowance(toAddress, 0);
+166:               IERC20(token).safeDecreaseAllowance(toAddress, 
IERC20(token).allowance(address(this), toAddress));
```

Another variant is using `safeApprove` when setting an initial allowance or resetting it to zero.

5.3. Absence of selectors

Risk Level: Info

Status: Fixed in commit 85af02.

Contracts:

- src/bridges/stargate/StargateV2.sol

Description:

The `StargateV2` contract, unlike all other bridge implementations, does not define function selectors for its bridging functions. It may be beneficial to add function selectors to contracts for consistency.

Remediation:

Consider adding selectors.

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.