# Exec Sum

A time-boxed security review of the **Realms bridge** protocol was done by **Antoine M.**, with a focus on the security aspects of the application's implementation. No criticial issues were found apart from an issue in the bridge deployment flow.

# Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

# About me

I'm an independant security researcher for the Starknet ecosystem. Find me on Twitter@Meckerrr

# About Realms

Realms is an on-chain autonomous world. More info can be found here: https://scroll.bibliothecadao.xyz/game/realms and here: https://www.realmseternum.com/

Realms interacts with Ethereum for some components of the project like NFTs or tokens, hence it needs a bridge.

## Observations

Nothing particular.

# Threat Model

## Privileged Roles & Actors

No owner and contracts are not upgradable.

## Security Interview

# Severity classification - OWASP

| Severity | Impact: High | Impact: Medium | Impact: Low |
|---|---|---|---|
| **Likelihood: High** | Critical | High | Medium |
| **Likelihood: Medium** | High | Medium | Low |
| **Likelihood: Low** | Medium | Low | Low |

**Impact** - the technical, economic and reputation damage of a successful attack

**Likelihood/Difficulty** - likelihood or difficulty is a rough measure of how likely or difficult this particular vulnerability is to be uncovered and exploited by an attacker.

**Severity** - the overall criticality of the risk

# Security Assessment Summary

***review commit hash* - 0765cdd**

### Scope

The following smart contracts were in scope of the audit:

- `bridge.cairo`
- `token.cairo`
- `bridge.sol`

The following number of issues were found, categorized by their severity:

- Critical & High: x issues
- Medium: x issues
- Low: x issues
- Informational: x issues

# Findings Summary

| ID | Title | Severity |
|---|---|---|
| [H-01] | Unsafe set token function | High |
| [I-01] | Use of EthAddress | Informational |
| [I-02] | Use of a custom ERC20 | Informational |
| [I-03] | Missing parameter in event | Informational |
| [I-04] | Use of upgradable contract | Informational |

# Detailed Findings

# [H-01] {Unsafe set token function}

## Severity - High

## Description

contract: `bridge.cairo`

The token is not set in the constructor. Instead the function `set_l2_token_once` is used. There can be a potential big issue there. Contract is deployed and the bridge address is set in the constructor. A new function call has to be made to set the `l2_token`. There is no access control on the function so anyone could call it and set any token. A multicall could maybe save the situation but do we want to take the risk. You should either add access control and maybe give the `admin` role the 0 address later or simply set the token in the constructor.

# [I-01] {Use of EthAddress}

## Severity - Informational

## Description

contract: `bridge.cairo`

Type `felt252` is used to deal with L1 addresses instead of EthAddress. The felt prime is

used as a bound for the eth address number. The `EthAddress` type has an embedded bound check.

# [I-02] {Use of a custom ERC20}

## Severity - Informational

## Description

contract: `token.cairo`

I would advise to use the openzeppelin erc20 as a base. It is always safer and a good practice to use a well tested and widely used contract.

# [I-03] {Missing parameter in event}

## Severity - Informational

## Description

contract: `bridge.cairo`

```
fn WithdrawalInitiated(recipient: felt252, amount: u256) {}
```

In this event, the `sender` parameter is missing. It would clearer imo to have that one (for indexing for instance).

# [I-04] {Use of upgradable contract}

## Severity - Informational

## Description

I'm simply questionning the fact of not using upgradable contracts (with a safe governance) in the context of a constantly evolving Starknet/Cairo.