Hypersign

# SMART CONTRACT AUDIT

ZOKYO.

May 5, 2021 | v. 1.0

## PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

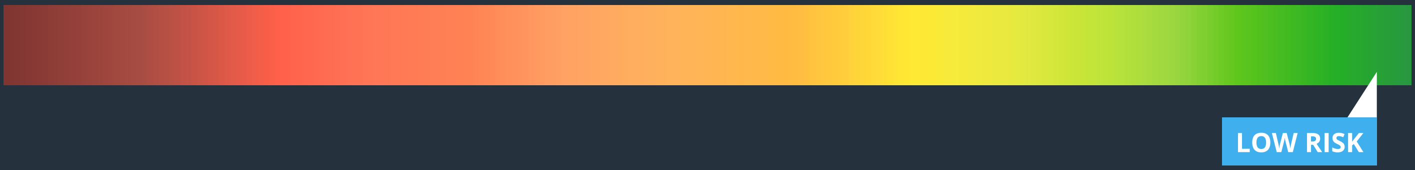
EXCELLENT

# TECHNICAL SUMMARY

This document outlines the overall security of the Hypersign smart contracts, evaluated by Zokyo's Blockchain Security team.

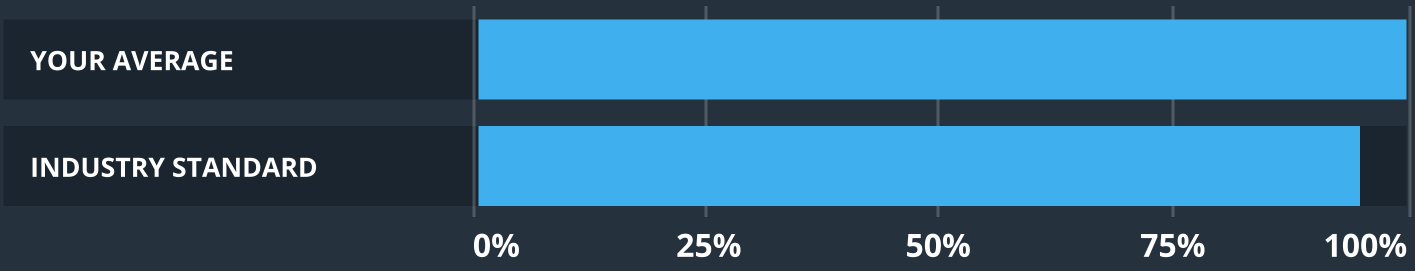
The scope of this audit was to analyze and document the Hypersign smart contract codebase for quality, security, and correctness.

## Contract Status



There were no critical issues found during the audit.

## Testable Code



The testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that’s able to withstand the Ethereum network’s fast-paced and rapidly changing environment, we at Zokyo recommend that the Hypersign team put in place a bug bounty program to encourage further and active analysis of the smart contract.

# TABLE OF CONTENTS

# AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Hypersign repository – https://github.com/hypersign-protocol/hid/.
Commit id – ead3c023c623947659ee03d6fef80844bfe4db1f.

**Throughout the review process, care was taken to ensure that the token contract:**

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Hypersign smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.

# EXECUTIVE SUMMARY

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

# STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

### ● Critical

The issue affects the ability of the contract to compile or operate in a significant way.

### ● High

The issue affects the ability of the contract to compile or operate in a significant way.

### ● Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

### ● Low

The issue has minimal impact on the contract’s ability to operate.

### ● Informational

The issue has no impact on the contract’s ability to operate.

# MANUAL REVIEW

## No vesting or farming logic found

**INFORMATIONAL** | UNRESOLVED

In Hypersign pitch deck was mentioned:

- Token sale functions;
- DeFi farming contracts.

None of those wasn’t submitted for audit.

**Recommendation:**
Submit for auditing farming and vesting contracts (if any).

# CODE COVERAGE AND TEST RESULTS FOR ALL FILES

## Tests written by Hypersign team

```
Contract: HID
    Success scenarios
        ✓ Name of token should be set (294ms)
        ✓ Symbol of token should be set (237ms)
        ✓ totalSupply of token should be 50 million (336ms)
        ✓ balance of admin should be 50 million (145ms)
        ✓ admin should send 100 token to alice (606ms)
        ✓ alice should send 10 token to bob (897ms)
        ✓ alice should send 1.5 token to Dany (522ms)
        ✓ bob should approve Tom to spend his 5 tokens (241ms)
        ✓ should overwrite the previous allowance (268ms)
        ✓ Tom should transfer 4 tokens of Bob to Charlie (674ms)
    Failure scenarios
        ✓ Charlie should NOT transfer more than his balance token (722ms)
        ✓ Should NOT transfer token to invalid address
        ✓ Tom should NOT transfer more than what is approved to him (117ms)


13 passing (7s)
```

| FILE | % STMTS | % BRANCH | % FUNCS | % LINES | UNCOVERED LINES |
|---|---|---|---|---|---|
| contracts/ | 100.00 | 100.00 | 100.00 | 100.00 | |
| HID.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| **All files** | **100.00** | **100.00** | **100.00** | **100.00** | |

## Tests written by Zokyo Secured team

As part of our work assisting Hypersign in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Hypersign contract requirements for details about issuance amounts and how the system handles these.

```
Contract: HID
   Success scenarios
      ✓ Name of token should be set (131ms)
      ✓ Symbol of token should be set (97ms)
      ✓ has 18 decimals (122ms)
      ✓ totalSupply of token should be 50 million (119ms)
      ✓ balance of admin should be 50 million (122ms)
      ✓ admin should send 100 token to alice (704ms)
      ✓ alice should send 10 token to bob (890ms)
      ✓ alice should send 1.5 token to Dany (700ms)
      ✓ bob should approve Tom to spend his 5 tokens (512ms)
      ✓ should overwrite the previous aloowance (366ms)
      ✓ should increase the previous allowance (378ms)
      ✓ emits an approval event (281ms)
      ✓ Tom should transfer 4 tokens of Bob to Charlie (806ms)
   Failure scenarios
      ✓ Charlie should NOT transfer more than his balance token (286ms)
      ✓ Should NOT transfer token to invalid address
      ✓ Tom should NOT transfer more than what is approved to him (448ms)

16 passing (7s)
```

| FILE | % STMTS | % BRANCH | % FUNCS | % LINES | UNCOVERED LINES |
|---|---|---|---|---|---|
| contracts/ | 100.00 | 100.00 | 100.00 | 100.00 | |
| HID.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| **All files** | **100.00** | **100.00** | **100.00** | **100.00** | |

We are grateful to have been given the opportunity to work with the Hypersign team.

**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.**

Zokyo's Security Team recommends that the Hypersign team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.