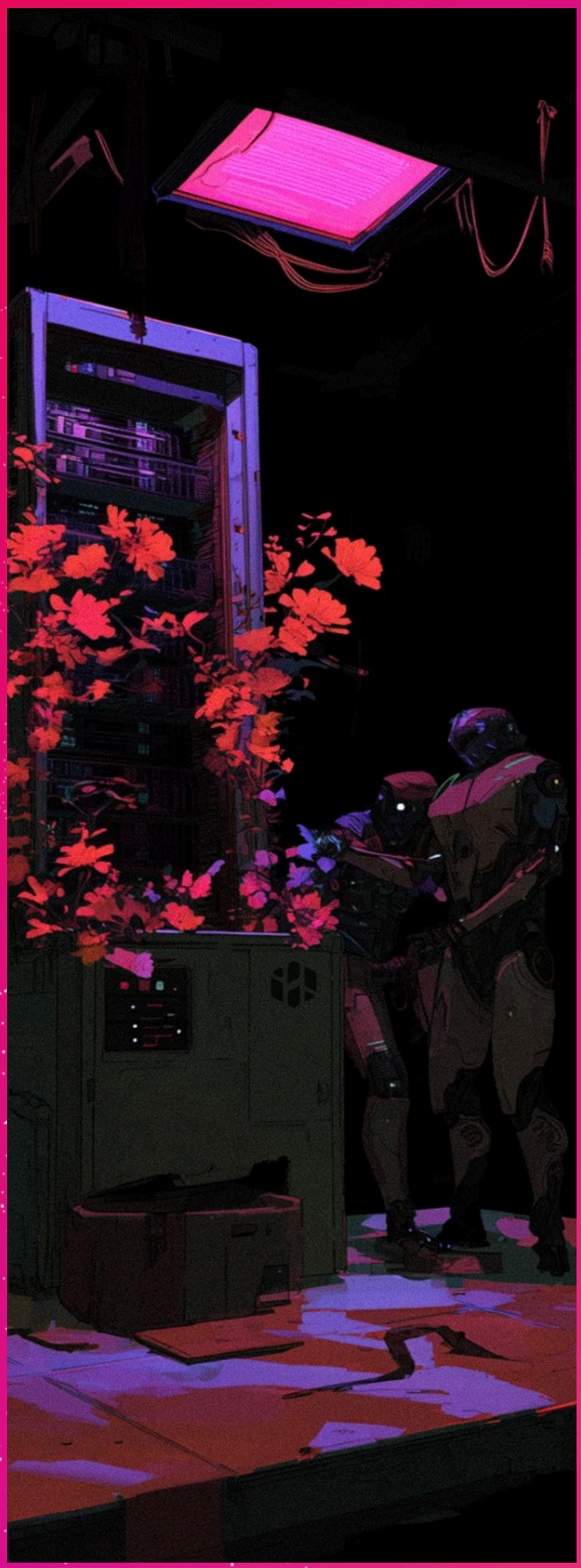
IMMUNEFI AUDIT

Immunefi / Halogen



DATE June 17, 2025

AUDITOR DrasticWaterMelon, Security Researcher

REPORT BY Immunefi

ABOUT IMMUNEFI

Immunefi is the leading onchain security platform, having directly prevented hacks worth more than $25 billion USD. Immunefi security researchers have earned over $120M USD for responsibly disclosing over 4,000 web2 and web3 vulnerabilities, more than the rest of the industry combined.

Through Magnus, Immunefi delivers a comprehensive suite of best-in-class security services through a single command center to more than 300 projects — including Sky (formerly MakerDAO), Optimism, Polygon, GMX, Reserve, Chainlink, TheGraph, Gnosis Chain, Lido, LayerZero, Arbitrum, StarkNet, EigenLayer, AAVE, ZKsync, Morpho, Ethena, USDT0, Stacks, Babylon, Fuel, Sei, Scroll, XION, Wormhole, Firedancer, Jito, Pyth, Eclipse, PancakeSwap and many more.

Magnus unifies SecOps across the entire onchain lifecycle, combining Immunefi's market leading products and community of elite security researchers with a curated set of the very best security products and technologies provided by top security firms — including Runtime Verification, Dedaub, Fuzzland, Nexus Mutual, Failsafe, OtterSec and others.

Magnus is powered by Immunefi's proprietary vulnerabilities dataset — the largest and most comprehensive in web3, ensuring that security leaders and teams have the best possible tools for identifying and mitigating life threats before they cause catastrophic harm, all while reducing operational overhead and complexity.

Learn how you can benefit too at immunefi.com.

TERMINOLOGY

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- **Likelihood** represents the likelihood of a finding to be triggered or exploited in practice
- **Impact** specifies the technical and business-related consequences of a finding
- **Severity** is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

LIKELIHOOD	IMPACT		
	HIGH	MEDIUM	LOW
CRITICAL	Critical	Critical	High
HIGH	High	High	Medium
MEDIUM	Medium	Medium	Low
LOW	Low		
NONE	None		

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

EXECUTIVE SUMMARY

Over the course of 1 day in total, Halogen Capital engaged with Immunefi to review the halogencapital/sc-digid repository. In this period of time a total of 5 issues were identified.

SUMMARY

Name	Halogen Capital
Repository	https://github.com/halogencapital/sc-digid
Type of Project	Asset management
Audit Timeline	June 9th - June 10th

ISSUES FOUND

Severity	Count	Fixed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	1	0	1
Insights	4	0	4

CATEGORY BREAKDOWN

Bug	1
Gas Optimization	3
Informational	1

FINDINGS

IMM-LOW-01

`FundGenerator` uses non-upgradeable dependencies #2

Id	IMM-LOW-01
Severity	LOW
Category	Bug
Status	Acknowledged

Description

`FundGenerator` (link) is intended to be used for the implementation of `TransparentUpgradeableProxy` instances.

Because of this, it should utilize the upgradeable versions of OpenZeppelin's library contracts, which use custom storage locations in order to ensure that future implementation upgrades do not change the implementation's storage layout, which would lead to corrupting the proxy's storage.

Furthermore, it is a good practice for implementation contracts to define a storage gap of 50 storage words, in order to make space for any additional storage variables that will be used in future implementations and to ensure such additional variables do not clash with any pre-existing storage.

Recommendation

Use `OwnableUpgradeable` (link) instead of Ownable.

IMM-INSIGHT-01

`FundGenerator.deployFund` can be optimized #5

Id	IMM-INSIGHT-01
Severity	INSIGHT
Category	Gas Optimization
Status	Acknowledged

Description

`FundGenerator` (link) defines a `FundGenerator.tokenCount` (link) storage variable which tracks the amount of `Fund` (link) instances deployed using `FundGenerator.deployFund()` (link).

`FundGenerator.deployFund()` gas consumption can be reduced by removing the `tokenCount` storage variable and defining a custom view function.

Recommendation

The existing `tokenCount` storage variable should be removed from the mentioned contract and a custom function should be implemented in its place:

```
TypeScript
contract FundGenerator is Ownable {
    address[] public tokens;
-   uint256 public tokenCount;

    mapping(address => mapping(address => bool)) public trustee;

    ...

    function deployFund(string calldata _name, string calldata _ticker, uint256 _supply)
onlyOwner public returns (address) {
        Fund token = new Fund(_name, _ticker, _supply, address(this));
        token.transfer(msg.sender, _supply);

        tokens.push(address(token));
```

```
-       tokenCount += 1;

        emit FundDeployed(address(token));
        return address(token);
    }

    ...

+   function tokenCount() external view returns(uint256) {
+       return tokens.length;
+   }
}
```

IMM-INSIGHT-02

Avoid logging `block.timestamp` #6

Id	IMM-INSIGHT-02
Severity	INSIGHT
Category	Gas Optimization
Status	Acknowledged

Description

`FundGenerator.addTrustee` (link) and `FundGenerator.removeTrustee` (link) respectively log `NewTrustee` (link) and `TrusteeRemoved` (link) events.

Both events contain `block.timestamp` as their log data, which is redundant given that such data is contained in the block's header.

Recommendation

To reduce the highlighted methods' gas consumption, avoid logging unnecessary information.

IMM-INSIGHT-03

`FundGenerator.deployFund()` does not comply with Checks-Effects-Interactions pattern #7

Id	IMM-INSIGHT-03
Severity	INSIGHT
Category	Informational
Status	Acknowledged

Description

`FundGenerator.deployFund()` (link) fails to comply with the CEI pattern as it first deploys a new `Fund` (link) instance and transfers all of the received tokens to the caller and then updates its internal storage.

Recommendation

Execute storage updates before deploying a new `Fund` instance and transferring the tokens.

IMM-INSIGHT-04

`Fund.generator` should be immutable #8

Id	IMM-INSIGHT-04
Severity	INSIGHT
Category	Informational
Status	Fixed in

Description

`Fund.generator` (link) is the sole account allowed to call `Fund.mint()` (link) and `Fund.burn()` (link) successfully.

Given that the contract doesn't implement a permissioned setter method to modify such a storage variable's value, it is effectively a constant value. As such, it may be declared as `immutable` to avoid loading its value from the contract's storage, reducing the contract's gas consumption.

Recommendation

Declare `Fund.generator` as `immutable`.