Company Simulator Competitive Audit

github.com/lucasfhope

October 31, 2025

Contents

Competitive Audit Details

This competitive audit is a CodeHawks First Flight.

The findings described in this report correspond to the following commit hash:

```
8bfead8b8a48bf7d20f1ee78a0566ab2b0b76e2d
```

Protocol Summary

Company Simulator is a decentralized smart contract system built in Vyper that simulates the operations of a virtual company. It includes modules for production, inventory management, customer demand, shareholding, reputation, and financial health.

Scope

```
src/
 --- Cyfrin_Hub.vy          # Core contract managing company operations
 --- CustomerEngine.vy      # Simulates customer demand and triggers
    sales
```

Roles

Owner:

- Deploys and controls core company functions such as production, share cap increases, and debt repayment

Investor:

- Public user who funds the company and receives proportional shares

Customer:

- Simulates demand by purchasing items from the company

Findings

Severity	Number of valid findings
High	1
Medium	0
Low	0
Total	**1**

[High-1] Investors can steal funds supplied by the owner (UNIQUE)

Description

The protocol intends for investors to invest ETH for company shares which can be redeemed for their supplied ETH and any company profits. The contract owner can also supply funding to the contract, but the owner does not receive any shares.

Because the owner does not receive shares, and the shares can be redeemed based on the percentage of shares and the net worth of the company, any funding the owner supplies can be redeemed by an investor.

```
@view
@internal
def get_share_price() -> uint256:
    """
    @notice Calculates the current share price based on net worth.
    @dev Net worth = company_balance - holding_debt (capped at 0).
    @dev Share price = net_worth / issued_shares.
    @dev If no shares issued, returns INITIAL_SHARE_PRICE.
    @return Price per share in wei.
    """
    if self.issued_shares == 0:
        return INITIAL_SHARE_PRICE
@>  net_worth: uint256 = max(self.company_balance - self.holding_debt,
    0)
@>  return net_worth // self.issued_shares
```

Risk

Likelihood:

This will occur whenever the owner funds the company through `fund_owner`, which requires calling `fund_cyfrin` with 0 as the parameter. An investor would also have funded the contract by calling `fund_cyfrin` with 1 as the parameter, which would allow them to receive shares. An investor could redeem their shares and collect some of the owner's funds, since the share price is based off of the net worth of the contract and the amount of shares. The owner does not receive any shares, but their supplied funding is directly tied to the share price.

Impact:

Funds supplied by the owner directly impact the share price and will be used to payout shareholders. While the MAX_PAYOUT_PER_SHARE is capped, repeated investing and withdrawing will allow malicious investors to steal ETH from the contract.

Proof of Concept

Add this test to `tests/unit/test_Industry.py`.

```
def test_Attacker_Can_Steal_Owner_Provided_Funds(industry_contract, OWNER, PATRICK):
    amount_to_invest = boa.env.get_balance(PATRICK)
    boa.env.set_balance(OWNER, SET_OWNER_BALANCE)
    with boa.env.prank(OWNER):
        industry_contract.fund_cyfrin(0, value=SET_OWNER_BALANCE)

    initial_balance = industry_contract.get_balance()
    print(f"Initial Industry Contract Balance: {initial_balance}")

    user_balance_before = boa.env.get_balance(PATRICK)
    print(f"Initial Attacker Balance:          {user_balance_before}")
    print(f"Attacker invest amount:            {amount_to_invest}")

    with boa.env.prank(PATRICK):
        industry_contract.fund_cyfrin(1, value=amount_to_invest)
        print(f"Attacker shares: {industry_contract.get_my_shares(caller=PATRICK)}")
        industry_contract.withdraw_shares()
    user_balance_after = boa.env.get_balance(PATRICK)
    print(f"End Attacker Balance:              {user_balance_after}")

    assert user_balance_after > user_balance_before, "Attacker should have increased their balance"
```

This will show that because the owner has funded the contract, an investor can invest and withdraw in one transaction, stealing 80% of the owners funds provided. Note that it is only 80% percent because of the 10% early withdrawal fee over the entire company net worth.

Recommended Mitigation

Consider giving the owner shares for the funds they provide. Another option would be to calculate share price based on the precentage of share funding vs owner funding, giving the owner the correct percentage of the company based on their supplied funds.