HawkHigh Audit Report

Version 1.0

Manar Maher

May 8, 2025

Hawkhigh Audit Report

manarmaher1

May 8, 2025

Prepared by: Lead Auditors: - Manar Maher

Table of Contents

Protocol Summary

Hawk High School is an upgradeable smart contract system utilizing OpenZeppelin’s UUPSUpgradeable pattern. The protocol simulates a school session where:

Students enroll by paying fees, receive weekly reviews, and must meet a cutoff score to graduate.

Teachers provide reviews and share fees as wages.

Principal manages teachers, starts/ends sessions, upgrades the system, and expels rule-breaking students.

At the end of each 4-week session, the system is upgraded by the principal, students are graduated, and wages are distributed.

Disclaimer

Manarmaher1 team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Roles

Principal:
Teacher:
Student:

Scope

```
├── src
│   ├── LevelOne.sol
│   └── LevelTwo.sol
```

Severity Criteria

Summary of Findings

High

[H-1]

Missing access control in LevelOne.Sol :: Initialize() making it possible for anyone to frontrun and initialize the contract

Description: Only the Principal should be able to initialize the contract and call the Initialize function, However due to the missing access control any malicious actor could call the functions and get ownership over the contract. Which is a common vulnerability that shows in UUPS upgradable contract utilizing Openzeppelin's upgradable pattern contracts.

```
```
function initialize(address _principal, uint256 _schoolFees, address
_usdcAddress) public initializer {
        if (_principal == address(0)) {
            revert HH__ZeroAddress();
        }
```
```

Impact: The intialize function of the malicious contract is left unprotected and anyone could call the function and initialize contract, setting themselves as the owner of the contract/Principal, which in turn allows them to completely compromise the contract's logic and functionality.

Recommended Mitigation: Add a modifier, and put proper access control to ensure the function is only initialized by the owner of the contract which in this case is the principal. and since there's no checks that ensure the contract has been initialized before and can only be initialized once, ensure it's only called once by the initializer modifier.

[H-2]

The `GraduateandUpgrade()` function is lacking checks to ensure that all students received 4 reviews and The session time ended before graduating and updating the contract.

Description: There is no require statements to check if all the students have received 4 reviews, one per week each, or that the time duration of the session have passed. the principal or a malicious owner of the contract in this case could manipulate that to upgrade the contract before the sessionEnd and before all number of students get reviewed and receive 4 reviews each.

```
```

```
   function graduateAndUpgrade(address _levelTwo, bytes memory) public
onlyPrincipal {
        if (_levelTwo == address(0)) {
            revert HH__ZeroAddress();
        }

        uint256 totalTeachers = listOfTeachers.length;

        uint256 payPerTeacher = (bursary * TEACHER_WAGE) / PRECISION;
        uint256 principalPay = (bursary * PRINCIPAL_WAGE) / PRECISION;

        _authorizeUpgrade(_levelTwo);

        for (uint256 n = 0; n < totalTeachers; n++) {
            usdc.safeTransfer(listOfTeachers[n], payPerTeacher);
        }

        usdc.safeTransfer(principal, principalPay);
    }
```

```

**Impact:**

The owner of the contract could graduate the students before :

1) All students received 4 reviews (1 per week × 4 weeks).
2) Current time ≥ sessionEnd (4 weeks after startSession).

This in turn could lead to

- completely missing with the rules of the contract
- student who didn't receive full reviews ( reviews < 4) could graduate
- the principal could upgrade the contract before the sessionEnd
- Teachers and principal could get the wages before the required time
- they could do that repeatedly which would lead to financial loss and manipulation of the funds at the bursary.

**Recommendation:**

Add checks to block upgrading and graduating before the sessionEnd and All the students have received 4 reviews each within the time duration, and for the function to revert if attempted otherwise.

## [H-3]

### In the 'upgradeandGraduate()' function. the teachers wages are miscalculated.

**Description:**

In the function, the teachers wages are calculated through this line of code:

```
```
uint256 payPerTeacher = (bursary * TEACHER_WAGE) / PRECISION;
```
```

The payment for each single teacher should be divided by the total number of teachers, but as we see here it's not.
in the contract all the teachers share the 35% of the bursary that's allocated to them and supposed to be shared among them all.

**Impact:** But due to this arithmetic error it will allows each teacher to get 35% of the whole bursary to themselves. Which will eventually lead to financial loss.

**Proof of concept:**
The contract logic says all the teachers get paid = (Bursary * 35%)
in the code it's each teacher gets paid = (bursary * techer_wage)/ precision

if we assume that the whole Bursary fund is = 1000 USD
```

each teacher wage = 35 USD
and the precision is = 100

and according to the equation in the quote
payperteacher = (1000*35)/100 = 350

taking into consideration that the bursary balance doesn't get updated after the payment.
second teacher payment will be calculated with the same equation = (1000*35)/100 = 350

this will lead to two things: - Wrong wage distribution as teachers will get the wrong payment, which is way above the amount specified to them.

- and fund drainage with more teachers checking in for their wages.
- the first few teachers might drain up the whole bursary.

Recommendation:

make sure to correct the logic used to distribute and calculate the payments for the teachers.
ex:'{

```
    uint256 totalTeacherpayment = (bursary * 35) / 100;
    uint256 payPerTeacher = totalTeacherpayemnt / listOfTeachers.length;

}
```

and ensure that the balance is updated and any remaining balance is handled appropriately

Medium

[M-1]

The giveReview() function incorrectly handles and counts the student reviews.

Description: in the giveReview function ::

```
 require(reviewCount[_student] < 5, "Student review count exceeded!!!");
        require(block.timestamp >= lastReviewTime[_student] + reviewTime,
"Reviews can only be given once per week");
```

According to the docs each student shall receive 1 review each per week, and since the

school session lasts 4 weeks, then each student receive 4 reviews.
at first look we see a problem, which is in the require statement here, the final count should be (reviewcount [_student] < 4).
assuming that a student have already received 4 reviews. and we start counting from zero (0,1,2,3,4,5) the students will be capable of receiving extra reviews.

with a closer look we see that the reviewcount is never incremented. so if the reviewcount stays with zero, no matter how many reviews the students get, it will not get incremented nor updated within the function, which could lead to unlimited number of reviews.

Recommendation:
increment reviewcount within the function to keep track of how many reviews the students received and make sure it doesn't exceed 4 reviews per session.

Low

[L-1]
Empty function in _authorizeUpgrade() function.

Description:

```
    function _authorizeUpgrade(address newImplementation) internal
override onlyPrincipal {}
```

Gas

[G-1]

`HH__HawkHighFeesNotPaid` is never used inside the LevelOne contract.

Informational

[I-1]

No events emitted in some functions like `initialize()` and `graduateAndUpgrade()`