Audit Findings Report

1. Executive Summary

This report documents the security review of the Halborn Ethereum CTF contracts: `HalbornToken`, `HalbornLoans`, and `HalbornNFT`. The assessment focused on upgradeability, access control, token economics, and reentrancy risks. All findings were validated against the provided Foundry unit tests.

Total Findings: 14 (13 Critical, 1 Low)

2. Overview

Folder name: HalbornCTF_Solidity_Ethereum

Date: 2026

Auditor: Seth Brockob

3. Scope

In-scope contracts:

- src/HalbornToken.sol
- src/HalbornLoans.sol
- src/HalbornNFT.sol

In-scope tests:

- test/HalbornToken.t.sol
- test/HalbornLoans.t.sol
- test/HalbornNFT.t.sol

Out of scope:

- Library dependencies (OpenZeppelin)
- Build artifacts
- Test utilities

4. Methodology

- Manual review of in-scope contracts
- Review of Foundry test suites
- Analysis of upgradeability, access control, and economic risk exposure

5. Risk Rating

- **Critical:** Full compromise, irreversible fund loss, or protocol takeover
- **High:** Major security or economic impact with realistic exploitation paths
- **Medium:** Material impact with limited practical exploitation
- **Low:** Minor impact or highly constrained exploitation
- **Informational:** Best practice issues without direct impact

6. Findings Summary

ID	Title	Risk
H-01	HalbornToken: UUPS Upgrade Bypass	Critical
H-02	HalbornToken: Loans Address Manipulation	Critical
H-03	HalbornToken: Unrestricted Token Minting	Critical
H-04	HalbornToken: Unrestricted Token Burning	Critical
H-05	HalbornLoans: UUPS Upgrade Bypass	Critical
H-06	HalbornLoans: Infinite Token Minting via Malicious Loan Contract	Critical
H-07	HalbornLoans: Arbitrary Token Burning via Loan Contract	Critical

H-08	HalbornLoans: Reentrancy in Collateral Withdrawal	Critical
H-09	HalbornNFT: Merkle Root Manipulation	Critical
H-10	HalbornNFT: Unlimited Airdrop Minting	Critical
H-11	HalbornNFT: UUPS Upgrade Bypass	Critical
H-12	HalbornNFT: Price Manipulation After Upgrade	Critical
H-13	HalbornNFT: ETH Drainage via Malicious Upgrade	Critical
H-14	HalbornNFT: Unchecked idCounter Overflow	Low

7. Detailed Findings

H-01 – HalbornToken: UUPS Upgrade Bypass

Risk: Critical

Description:

The token contract inherits from `UUPSUpgradeable`, but `_authorizeUpgrade()` is empty. Any address can upgrade the implementation and reinitialize ownership. The test `test_vulnerableUUPSupgrade()` shows an unauthorized user upgrading to a malicious implementation.

Code Section:

- src/HalbornToken.sol: `_authorizeUpgrade(address)`
- Test: `test_vulnerableUUPSupgrade()` in test/HalbornToken.t.sol

Impact:

Complete contract takeover. Attackers gain control of minting, burning, and ownership logic, which directly destroys token integrity and value.

Recommendation on Improvement:

Implement access control on upgrades:

```
function _authorizeUpgrade(address) internal override onlyOwner {}
```

H-02 – HalbornToken: Loans Address Manipulation

Risk: Critical

Description:

After a malicious upgrade, an attacker can call `setLoans()` and permanently assign themselves as the loans authority. The test `test_setLoansAddress()` shows that the original owner can no longer change the loans address.

Code Section:

- src/HalbornToken.sol: `setLoans(address)`
- Test: `test_setLoansAddress()` in test/HalbornToken.t.sol

Impact:

Permanent mint/burn backdoor. The attacker can mint or burn arbitrarily, permanently compromising token economics.

Recommendation on Improvement:

Add governance controls (timelock or multi-sig) around `setLoans()` and prevent reinitialization after upgrade.

H-03 – HalbornToken: Unrestricted Token Minting

Risk: Critical

Description:

`mintToken()` trusts the loans address without further validation. In `test_unlimitedMint()`, the attacker upgrades the contract, sets themselves as loans, and mints `type(uint256).max`.

Code Section:

- src/HalbornToken.sol: `mintToken(address,uint256)`
- Test: `test_unlimitedMint()` in test/HalbornToken.t.sol

Impact:

Infinite token inflation and total loss of economic integrity.

Recommendation on Improvement:

Enforce hard supply caps and validate minting logic beyond a single trusted address.

H-04 – HalbornToken: Unrestricted Token Burning

Risk: Critical

Description:

`burnToken()` allows burning from any address by the loans authority. In `test_unlimitedBurn()`, the attacker upgrades and burns a victim's balance.

Code Section:

- src/HalbornToken.sol: `burnToken(address,uint256)`
- Test: `test_unlimitedBurn()` in test/HalbornToken.t.sol

Impact:

Direct, irreversible user fund loss.

Recommendation on Improvement:

Restrict burns to validated loan-default scenarios with explicit authorization and logging.

H-05 – HalbornLoans: UUPS Upgrade Bypass

Risk: Critical

Description:

The loans contract also leaves `_authorizeUpgrade()` empty. In `test_vulnerableUUPSupgrade()`, an unauthorized user upgrades and reinitializes the loans contract.

Code Section:

- src/HalbornLoans.sol: `_authorizeUpgrade(address)`
- Test: `test_vulnerableUUPSupgrade()` in test/HalbornLoans.t.sol

Impact:

Full takeover of the lending system, enabling arbitrary changes to collateral and mint/burn logic.

Recommendation on Improvement:

Add access control to upgrade logic and enforce upgrade governance.

H-06 – HalbornLoans: Infinite Token Minting via Malicious Loan Contract

Risk: Critical

Description:

After upgrading to a malicious loans implementation, the attacker uses the trusted mint path to create infinite supply.
`test_vulnerableLoanContractReksTokenMint()` shows the attacker minting `type(uint256).max`.

Code Section:

- src/HalbornLoans.sol: `getLoan(uint256)` mint path
- Test: `test_vulnerableLoanContractReksTokenMint()` in test/HalbornLoans.t.sol

Impact:

Unbounded inflation and collapse of token economics.

Recommendation on Improvement:

Cap minting per loan, validate collateralization, and limit trusted mint authority.

H-07 – HalbornLoans: Arbitrary Token Burning via Loan Contract

Risk: Critical

Description:

The malicious loans upgrade burns arbitrary user balances. `test_vulnerableLoanContractReksTokenBurn()` demonstrates burning a victim's balance through the trusted loans relationship.

Code Section:

- src/HalbornLoans.sol: `returnLoan(uint256)` burn path
- Test: `test_vulnerableLoanContractReksTokenBurn()` in test/HalbornLoans.t.sol

Impact:

Targeted or systemic destruction of user balances.

Recommendation on Improvement:

Validate burn operations against loan state and borrower authorization before execution.

H-08 – HalbornLoans: Reentrancy in Collateral Withdrawal

Risk: Critical

Description:

`withdrawCollateral()` transfers NFTs before updating state, enabling reentrancy via `onERC721Received`. In `test_Reentrancy()`, the attacker withdraws multiple NFTs and drains max tokens via `getLoan()`.

Code Section:

- src/HalbornLoans.sol: `withdrawCollateral(uint256)`
- Test: `test_Reentrancy()` in test/HalbornLoans.t.sol

Impact:

Double collateral withdrawal and maximum token drain, causing severe protocol loss.

Recommendation on Improvement:

Apply reentrancy guards and move state updates before external calls (checks-effects-interactions).

H-09 – HalbornNFT: Merkle Root Manipulation

Risk: Critical

Description:

`setMerkleRoot()` has no access control. `test_setMerkelRoot()` shows an unauthorized user replacing the whitelist root.

Code Section:

- src/HalbornNFT.sol: `setMerkleRoot(bytes32)`
- Test: `test_setMerkelRoot()` in test/HalbornNFT.t.sol

Impact:

Whitelist bypass, enabling unauthorized NFT minting and economic dilution.

Recommendation on Improvement:

Restrict `setMerkleRoot()` to authorized admins (onlyOwner or governance).

H-10 – HalbornNFT: Unlimited Airdrop Minting

Risk: Critical

Description:

After root manipulation, crafted proofs allow unlimited mints. `test_setMintUnlimited()` demonstrates minting multiple IDs using attacker-controlled proofs.

Code Section:

- src/HalbornNFT.sol: `mintAirdrops(uint256,bytes32[])`
- Test: `test_setMintUnlimited()` in test/HalbornNFT.t.sol

Impact:

NFT supply inflation and collapse of scarcity.

Recommendation on Improvement:

Track claimed IDs and per-address mint limits to enforce one-time claims.

H-11 – HalbornNFT: UUPS Upgrade Bypass

Risk: Critical

Description:

The NFT contract exposes the same empty `_authorizeUpgrade()`. `test_vulnerableUUPSupgrade()` shows unauthorized upgrades and reinitialization.

Code Section:

- src/HalbornNFT.sol: `_authorizeUpgrade(address)`
- Test: `test_vulnerableUUPSupgrade()` in test/HalbornNFT.t.sol

Impact:

Complete NFT contract takeover, enabling arbitrary minting, pricing changes, and ETH theft.

Recommendation on Improvement:

Restrict upgrades with access control and governance approvals.

H-12 – HalbornNFT: Price Manipulation After Upgrade

Risk: Critical

Description:

After a malicious upgrade, the attacker reinitializes the contract and sets arbitrary pricing. `test_setPrice()` confirms price manipulation.

Code Section:

- src/HalbornNFT.sol: `initialize(bytes32,uint256)`
- Test: `test_setPrice()` in test/HalbornNFT.t.sol

Impact:

Economic manipulation of mint pricing, enabling free mints or denial of service.

Recommendation on Improvement:

Prevent reinitialization after deployment and separate price updates into restricted admin functions.

H-13 – HalbornNFT: ETH Drainage via Malicious Upgrade

Risk: Critical

Description:

In `test_stealETH()`, the attacker upgrades to a malicious implementation and drains all ETH from the contract via a modified `withdrawETH()`.

Code Section:

- src/HalbornNFT.sol: `withdrawETH(uint256)`
- Test: `test_stealETH()` in test/HalbornNFT.t.sol

Impact:

Total loss of ETH held in the contract, causing direct user losses.

Recommendation on Improvement:

Protect upgrades and withdrawals with multi-sig or timelock governance.

H-14 – HalbornNFT: Unchecked idCounter Overflow

Risk: Low

Description:

`idCounter` is incremented in an unchecked block. The test `test_overflowCounter()` highlights this as theoretically overflowable but practically infeasible.

Code Section:

- src/HalbornNFT.sol: `mintBuyWithETH()` unchecked increment
- Test: `test_overflowCounter()` in test/HalbornNFT.t.sol

Impact:

Theoretical token ID reuse if overflow occurs. Practical exploitation is infeasible but it remains a correctness issue.

Recommendation on Improvement:

Use checked arithmetic or enforce a maximum supply cap.

8. Conclusion

All contracts exhibit systemic upgradeability weaknesses that allow complete takeovers, unlimited minting, and ETH theft. These issues introduce severe economic risk to token and NFT holders. Upgrade authorization and access control must be remediated before any production deployment.