

Audit Findings Report

1. Executive Summary

This report documents the security review of the Halborn NEAR CTF contracts. The assessment focused on state correctness, storage persistence, and economic integrity. Findings were validated against the embedded Rust unit tests.

Total Findings: 5 (4 Critical, 1 High)

2. Overview

Folder name: HalbornCTF_Rust_NEAR

Date: 2026

Auditor: Seth Brockob

3. Scope

In-scope contracts:

- halborn-near-ctf/src/lib.rs
- halborn-near-ctf-associated-contract/src/lib.rs
- halborn-near-ctf-staking/src/lib.rs

Out of scope:

- Deployment scripts
- Build configuration files
- Documentation files

4. Methodology

- Manual review of contract logic
- Review of embedded unit tests
- Economic impact analysis for token supply and staking accounting

5. Risk Rating

- **Critical:** Full loss of functionality, funds, or protocol correctness
- **High:** Major 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
N-01	resume() Sets Incorrect Status	Critical
N-02	mint_tokens() Loses Tokens for Unregistered Users	Critical
N-03	Metadata Functions Consume Metadata	High
N-04	make_event_offline() Has No Effect	Critical
N-05	unstake() Logic Error and Accounting Inconsistency	Critical

7. Detailed Findings

N-01 - resume() Sets Incorrect Status

Risk: Critical

Description:

The resume() function sets the contract status to Paused instead of Working. The tests test_resume_bug_contract_stays_paused() and test_resume_bug_CANNOT_use_contract() confirm the contract remains unusable after calling resume().

Code Section:

- halborn-near-ctf/src/lib.rs: resume()
- Tests: test_resume_bug_contract_stays_paused(), test_resume_bug_CANNOT_use_contract()

Impact:

Permanent denial of service. Once paused, the contract cannot be resumed, and all guarded functions remain unusable.

Recommendation on Improvement:

Set the contract status to Working:

```
self.status = Status::Working;
```

N-02 - mint_tokens() Loses Tokens for Unregistered Users

Risk: Critical

Description:

`mint_tokens()` increases `total_supply` even when the recipient is not registered. The tests `test_mint_tokens_bug_unregistered_user()` and `test_mint_tokens_bug_token_loss()` confirm supply inflation without a corresponding balance.

Code Section:

- halborn-near-ctf/src/lib.rs: `mint_tokens()`
- Tests: `test_mint_tokens_bug_unregistered_user()`,
`test_mint_tokens_bug_token_loss()`

Impact:

Permanent token loss and supply inflation without ownership. This creates economic inconsistencies and undermines token integrity.

Recommendation on Improvement:

Register users before crediting balances:

```
if !self.accounts.contains_key(&account_id) {  
    self.accounts.insert(&account_id, &0u128);  
}
```

N-03 - Metadata Functions Consume Metadata

Risk: High

Description:

`get_symbol()`, `get_name()`, and `get_decimals()` use `token_metadata.take()` which removes metadata from storage. The tests `test_metadata_consumption_bug()`, `test_metadata_consumption_bug_multiple_functions()`, and `test_metadata_consumption_bug_ft_metadata()` show metadata becomes unavailable after one call.

Code Section:

- halborn-near-ctf/src/lib.rs: `get_symbol()`, `get_name()`, `get_decimals()`
- Tests: `test_metadata_consumption_bug()`,
`test_metadata_consumption_bug_multiple_functions()`,
`test_metadata_consumption_bug_ft_metadata()`

Impact:

Token metadata becomes permanently inaccessible, breaking standard NEAR token metadata interfaces and downstream integrations.

Recommendation on Improvement:

Use read-only access with `get()` instead of `take()`.

N-04 - `make_event_offline()` Has No Effect

Risk: Critical

Description:

`make_event_offline()` modifies a local copy of the event and never persists it. The test `test_make_event_offline_bug_no_effect()` confirms the event remains live and registrations still succeed.

Code Section:

- halborn-near-ctf-associated-contract/src/lib.rs: `make_event_offline()`
- Test: `test_make_event_offline_bug_no_effect()`

Impact:

Events cannot be taken offline. This bypasses intended access control and allows registrations during cancellations or shutdowns.

Recommendation on Improvement:

Persist the updated event back into storage:

```
let mut event = self.events.get(&event_id).unwrap();
event.is_live = false;
self.events.insert(&event_id, &event);
```

N-05 - unstake() Logic Error and Accounting Inconsistency

Risk: Critical

Description:

unstake() uses saturating_sub and refunds inconsistently. The tests test_unstake_bug_when_balance_reaches_zero(), test_unstake_bug_logic_inconsistency(), and test_unstake_edge_case_saturating_sub() show incorrect total_staked accounting.

Code Section:

- halborn-near-ctf-staking/src/lib.rs: unstake()
- Tests: test_unstake_bug_when_balance_reaches_zero(), test_unstake_bug_logic_inconsistency(), test_unstake_edge_case_saturating_sub()

Impact:

Incorrect total_staked accounting and inconsistent refunds, undermining staking economics and protocol balance correctness.

Recommendation on Improvement:

Validate amount <= balance and update total_staked based on the actual unstaked amount.

8. Conclusion

The NEAR contracts contain critical state management flaws that lead to permanent pause states, token loss, and incorrect accounting. These issues compromise both functionality and economic correctness. Remediation should prioritize state correctness, storage persistence, and balance validation.