Audit Findings Report

1. Executive Summary

This report documents the security review of the Halborn Substrate CTF pallets. The assessment focused on pause-state integrity and emergency control enforcement. Findings were validated against the pallet unit tests.

Total Findings: 3 (2 Critical, 1 High)

2. Overview

Folder name: HalbornCTF_Rust_Substrate

Date: 2026

Auditor: Seth Brockob

3. Scope

In-scope pallets:

- pallets/pause/src/lib.rs
- pallets/allocations/src/lib.rs

In-scope tests:

- pallets/pause/src/tests.rs
- pallets/allocations/src/tests.rs

Out of scope:

- Runtime configuration files
- Benchmarking code
- Build scripts

4. Methodology

- Manual review of pallet logic
- Review of pallet unit tests
- State integrity and access control analysis

5. Risk Rating

- **Critical:** System-wide denial of service or loss of control
- **High:** Major operational impact or bypass of emergency controls
- **Medium:** Material impact with limited exploitation
- **Low:** Minor impact

6. Findings Summary

ID	Title	Risk
S-01	toggle() Does Not Toggle State	Critical
S-02	unpause() Does Not Unpause System	Critical
S-03	Allocations Bypass Pause Check	High

7. Detailed Findings

S-01 – toggle() Does Not Toggle State

Risk: Critical

Description:

`toggle()` writes the current pause state back into storage instead of negating it. The test `toggle_bug_does_not_toggle()` shows that calling toggle() does not change the paused state.

Code Section:

- pallets/pause/src/lib.rs: `toggle()`
- Test: `toggle_bug_does_not_toggle()` in pallets/pause/src/tests.rs

Impact:

Administrators cannot reliably toggle the pause state, leading to potential permanent downtime or inability to enforce emergency shutdowns.

Recommendation on Improvement:

Store the negated value:

```
<Paused<T>>::put(!Self::paused());
```

S-02 – unpause() Does Not Unpause System

Risk: Critical

Description:

`unpause()` writes the current pause state back into storage and emits a misleading event. The test `unpause_bug_does_not_unpause()` confirms the system remains paused after calling unpause().

Code Section:

- pallets/pause/src/lib.rs: `unpause()`
- Test: `unpause_bug_does_not_unpause()` in pallets/pause/src/tests.rs

Impact:

Once paused, the system cannot be resumed through the intended function. Event logs indicate false state transitions, increasing operational risk during incidents.

Recommendation on Improvement:

Set the pause state to false directly:

```
<Paused<T>>::put(false);
```

S-03 – Allocations Bypass Pause Check

Risk: High

Description:

`allocate_coins()` does not verify whether the system is paused, even though the pallet depends on `pallet_pause::Config`. The test `allocations_bypass_pause_check()` confirms allocations continue while paused.

Code Section:

- pallets/allocations/src/lib.rs: `allocate_coins()`
- Test: `allocations_bypass_pause_check()` in pallets/allocations/src/tests.rs

Impact:

Emergency pause is ineffective for allocations. This allows protocol activity during shutdown, exposing the system to operational and economic risk during incidents.

Recommendation on Improvement:

Add a pause check before allocation logic:

```
ensure!(!pallet_pause::Pallet::<T>::paused(), Error::<T>::SystemPa
```

8. Conclusion

The Substrate pallets contain critical pause-state logic errors that break emergency controls. The pause pallet cannot reliably toggle or resume, and the allocations pallet ignores the pause state entirely. These issues undermine shutdown safety and should be corrected before production use.