Security Research Labs

Origin Trail Feature Security Assurance: ForeignAssets

Hacking assessment report

V1.2, 13 August 2025, FINAL

Audited By: Aarnav Bos aarnav@srlabs.de

Marc Heuse marc@srlabs.de

Abstract. This work describes the result of the thorough and independent security assurance audit performed by Security Research Labs for Origin Trail's implementation of support for Foreign Assets, wrapped TRAC, and XCM within the Neuroweb blockchain. Security Research Labs is a consulting firm that has been providing specialized audit services in the Polkadot ecosystem since 2019, including for the Substrate and Polkadot projects.

During this audit, the Origin Trail team provided access to relevant code and supported the research team effectively. The code in scope was verified to assure that the business logic of the product is resilient to hacking and abuse.

The research team identified several issues ranging from high to low severity and recommends mitigating these before upgrading the Neuroweb runtime. Every audit finding has been verified to be mitigated, only benchmarks have to be run again.

Content

1 Disclaimer

This report describes the findings and core conclusions derived from the audit carried out by Security Research Labs within the agreed-on timeframe and scope as detailed in Table 1. Please note that this report does not guarantee that all existing security vulnerabilities were discovered in the codebase exhaustively.

2 Motivation and scope

Many blockchains represent foreign tokens natively, in the manner of wrapped tokens. Typically, wrapped tokens are minted with equivalence to the amount of native tokens locked on the corresponding foreign chain. Origin Trail's feature introduces a representation of TRAC, a foreign token available from Snowbridge through AssetHub on the Neuroweb blockchain. Additionally, general support for foreign assets and XCM was introduced.

The current audit was a spotlight review covering the introduction of XCM and three new pallets to the Neuroweb runtime. The new pallets consist of two instances of pallet-assets to represent foreign assets, and pallet-wrapper, a pallet to mint wrapped TRAC and lock foreign TRAC.

The in-scope components are reflected in Table 1. The audit was limited to the core logic components added and modified in a single pull-request [1]. Thus, the audit did not cover the full runtime, but only the specific part of the runtime affected by the pull request.

Repository	Component(s)
Neuroweb	/runtime [2]
	/pallets/wrapper [3]

Table 1. In-scope Neuroweb components

3 Findings summary

We identified five issues - summarized in Table 2 - during our analysis of the in-scope components of the Neuroweb codebase. In summary, one high severity, three medium severity, and one low severity issues were found.

Issue	Severity	Status
Incorrect enforcement of ProxyType::NonTransfer may lead to theft of funds and assets [4]	High	Mitigated
Missing pausing functionality for TRAC wrapper [5]	Medium	Mitigated, needs new benchmark run
Incorrect configuration of runtime weights [6]	Medium	Accepted
Lack of approval deposit enables storage bloating [7]	Medium	Mitigated
Runtime configuration allows anyone to create both local and foreign assets [8]	Low	Mitigated

Table 2 Issue summary

4 Detailed findings

4.1 Incorrect enforcement of ProxyType::NonTransfer may lead to theft

Attack scenario	**An attacker can steal a user's funds**
Location	Runtime
Attack impact	A user may lose their assets or funds stolen or lost.
Severity	High
Status	Mitigated

The proxy extrinsic allows a Delegate to call extrinsics on behalf of their Delegator, restricted to calls set by their `ProxyType`. The `ProxyType::NonTransfer` prevents the delegate from dispatching any extrinsics that may allow the movement or manipulation of funds or assets.

```
impl InstanceFilter<RuntimeCall> for ProxyType {
  fn filter(&self, c: &RuntimeCall) -> bool {
    match self {
       ...
      ProxyType::NonTransfer => !matches!(
        c,
        RuntimeCall::Balances(..) |
          RuntimeCall::Assets(..) |
          RuntimeCall::Vesting(
              pallet_vesting::Call::vested_transfer { .. })
          ),
      ...
    }
  }
}
```

However, the filter does not consider the introduction of new pallets, `ForeignAssets` and `Wrapper`, both of which allow transfers and manipulation of funds and issuance.

This opens up a large avenue for attacks. Notably:

1. An attacker may transfer the assets to their account.
2. An attacker could create a lot of assets, draining a user's funds.
3. An attack they could mint or burn an arbitrary amount of assets if their Proxy parent owns the asset.
4. An attacker could exhaust a user's foreign TRAC by transferring exchanging it for wrapped TRAC.

To mitigate this issue, we recommend adding `Wrapper` and `ForeignAssets` to the `ProxyType::NonTransfer`.

4.2 Missing pausing functionality for TRAC wrapper

Attack scenario	**An attacker can mint arbitrary amounts of wrapped TRAC**
Location	pallet-wrapper
Attack impact	An attacker could sway the consensus system or devalue the token
Severity	Medium
Status	Mitigated

The pallet-wrapper pallet allows users to deposit foreign TRAC and in exchange, mint a 1:1 amount of local, "wrapped" TRAC.

```
/// Wrap foreign TRAC tokens into local TRAC tokens
///
/// Transfers foreign TRAC from sender to pallet account and
mints equal amount of local TRAC to sender
#[pallet::call_index(0)]
#[pallet::weight(T::WeightInfo::trac_wrap())]
pub fn trac_wrap(
    origin: OriginFor<T>,
    #[pallet::compact] amount: T::Balance,
) -> DispatchResult {
     ...
}
```

If the source of foreign TRAC is affected by a security issue, an attacker may be able to leverage this to mint an arbitrary amount of wrapped TRAC, leading to compromised network integrity. Due to the use of wrapped TRAC in staking, an attacker may be able to disrupt network integrity by swaying the consensus system in their favour. Additionally, an attacker could devalue the token on exchanges by attempting to swap large amounts.

We recommend implementing a mechanism that enables governance or root-level authority to pause the wrapping and unwrapping of TRAC, ensuring responsive control and risk mitigation when necessary.

4.3 Incorrect configuration of runtime weights

Attack scenario	**An attacker wants to hinder chain operation**
Location	Runtime
Attack impact	An attacker could spam the chain, resulting in degradation of performance
Severity	Medium
Status	Accepted

Every extrinsic available in a Substrate project needs to have a weight assigned which is calculated using a benchmarking system. Depending on the runtime configuration for every pallet, the benchmarking result might be different. The WeightInfo value is incorrectly set for multiple pallets used in the runtime, as it points to the `SubstrateWeight` defined in the pallet itself.

A list of in-scope pallets that are affected by this issue:

- pallet-balances
- pallet-assets
- pallet-assets-instance2

There are several other pallets whose configuration is affected by the same issue; however, they are out of scope and thus unlisted.

As pallet extrinsic benchmarks can be dependent on the actual runtime configuration, this can lead to:

- Overweight extrinsics
- Underweight extrinsics

for all extrinsics that are using the substrate-node template runtime weights.

To mitigate this issue we recommend benchmarking all extrinsics, including Substrate ones, with the actual runtime configuration by including them in the `define_benchmarks!` block, and setting the corresponding WeightInfo value.

4.4 Lack of ApprovalDeposit enables storage bloating

Attack scenario	An attacker wants to hinder node operation
Location	Runtime
Attack impact	An attacker could fill up chain storage, resulting in degradation of performance
Severity	Medium
Status	Mitigated

When configuring pallet-assets for local and foreign assets. assets, the runtime sets the `ApprovalDeposit` to 0.

```
pub const ApprovalDeposit: Balance = 0;
```

This enables an attacker to create approvals, which are stored on-chain, without paying a storage deposit, causing permanent state bloat, degradation in node performance and increasing costs for all network participants.

As mitigation, we recommend enforcing a deposit for approvals by setting `ApprovalDeposit` to a non-zero value. `ExistentialDeposit`, for example, is a commonly utilized value.

4.5 Runtime configuration allows anyone to create assets

Attack scenario	An attacker wants to disrupt network reputation
Location	Runtime
Attack impact	An attacker could defraud users
Severity	Low
Status	Mitigated

In the runtime configuration for `Assets` and `ForeignAssets`, `CreateOrigin` is configured to allow any signed origin to create an asset as long as they're willing to pay a deposit. Allowing any signed origin to create assets may lead to squatting attacks, where attackers, both on chain and through XCM, create assets on IDs that might be reserved for protocol use. Additionally, an attacker may create fake tokens that mimic existing assets to confuse users. This is further amplified by the possibility to create a foreign asset with a user supplied arbitrary `MultiLocation`.

Attackers that squat asset IDs that are potentially reserved for protocol use may hinder the deployment of new assets, or raise questions on the legitimacy of the deployed assets. If the attacker is able to mimic foreign tokens that Neuroweb plans on introducing they may be able to trick users into buying fraudulent assets. Another scenario is when an attacker deploys a representation of a legitimate foreign asset and claims it is supported on Neuroweb, leading to fraud and reputation concerns.

Additionally, due to the implementation of the `MultiCurrencyAdapter`, if functionality is introduced where assets are not strictly verified or hardcoded (like with pallet-wrapper), an attacker may be able to exchange arbitrary assets for legitimate ones.

As mitigation, we recommend restricting the origin for `CreateOrigin` to a governance or root-level authority, ensuring that asset squatting and counterfeit asset creation are prevented.

5 Bibliography

[1] [Online]. Available: https://github.com/OriginTrail/neuroweb/pull/125

[2] [Online]. Available: https://github.com/cl0w5/neuroweb/tree/afcd891edbbf59fc80f71cd256472a9ff91fa32f/runtime

[3] [Online]. Available: https://github.com/cl0w5/neuroweb/tree/afcd891edbbf59fc80f71cd256472a9ff91fa32f/pallets/wrapper

[4] [Online]. Available: https://github.com/srlabs/origin-trail-audit/issues/4

[5] [Online]. Available: https://github.com/srlabs/origin-trail-audit/issues/2

[6] [Online]. Available: https://github.com/srlabs/origin-trail-audit/issues/1

[7] [Online]. Available: https://github.com/srlabs/origin-trail-audit/issues/3

[8] [Online]. Available: https://github.com/srlabs/origin-trail-audit/issues/5