SBP Automated Security Assessment for peaq

2023-06-01

[Medium] Insecure randomness algorithm usage

Summary

The source of randomness in the runtime for the `pallet_contracts` is configured to use the `pallet_randomness_collective_flip` implemented in Substrate.
The output of collective flip is highly predictable as it is based on the last 81 blocks and should not be used as a true source of randomness.

Issue details
In the definition of the configuration for the runtime in runtime/krest/lib.rs:

```
// Create the runtime by composing the FRAME pallets that were previously configured.
construct_runtime!(
   pub enum Runtime where
      Block = Block,
      NodeBlock = opaque::Block,
      UncheckedExtrinsic = UncheckedExtrinsic
   {
      System: frame_system::{Pallet, Call, Config, Storage, Event<T>} = 0,
      RandomnessCollectiveFlip: pallet_randomness_collective_flip::{Pallet, Storage} = 1,
      Timestamp: pallet_timestamp::{Pallet, Call, Storage, Inherent} = 2,
      // Add other FRAME pallets here
   }
);
```

as well as in the configuration of the `pallet_contracts` , the is `pallet_randomness_collective_flip` used.

```
impl pallet_contracts::Config for Runtime {
            type Time = Timestamp;
            type Randomness = RandomnessCollectiveFlip;
            ...
}
```

Mitigation

Using a secure randomness for the contracts pallet, either with the usage of an oracle of a project like `drand` or a secure library.

[High] ExistentialDeposits is configured to be 0
Summary

In the runtime configuration, the `ExistentialDeposits` is set to be 0. This could result in cheaply filling up the blockchain storage with a lot of accounts that have low or zero balance. We recommend to set the value of the existential deposit to a sensibly low, but non-zero value.
Issue description
In Substrate, the reason for requiring an existential deposit for accounts is to optimise storage. Having an account going below the existential deposit will result in the account being reaped (the account data will be deleted along with the remaining funds in that address) to conserve space on the blockchain. In the current runtime configuration of Peaq (runtime/krest/src/lib.rs), the existential deposit is set to be 0:

```
pub ExistentialDeposits: |_currency_id: CurrencyId| -> Balance {
   0
};
```

This means that once an account had any balance stored, the `AccountData` associated with it will never be cleared from the storage.
Having transaction fees lowers the risk but does not mitigate this issue: the cost of permanent storage is not accounted for in the weight calculation for extrinsics in substrate in general. This could allow an attacker to fill up the blockchain storage.

Risk
An Attacker might fill up the blockchain storage by distributing small amounts of balances between large numbers of accounts.
Mitigation
We recommend to set the value of the existential deposit to a sensibly low, but non-zero value. As an example, setting the existential deposit to be the worth of $1 in Peaq token would already make such an attack expensive enough to lower the risk of storage clutter.