Summary

Eth2 validators may lose access to their withdrawal credentials, and as a result lose access to their 32 ETH deposit and inflation rewards. There is also no way to upgrade to new withdrawal credential types. We propose a protocol that allows rotating a validator's withdrawal credentials.

Context & Motivation

In order to stake, a user sends 32 ETH to the Eth2 Deposit Contract, along with:

1. A BLS public key, used by their validator for producing blocks
2. A withdrawal credential (as specified here)

If a validator's state is Unslashed and Exitable, 27 hours later, they can withdraw their 32 ETH principal and any rewards earned to the withdrawal credentials' address. Withdrawal credentials cannot be changed. If a validator loses access to their BLS withdrawal credentials, they are unable to reclaim their funds.

There's a few potential pain points when managing BLS keys, which may result in loss of funds:

- Hardware wallets are not supported by the Deposit CLI, meaning that most validators' credentials so far are on a mnemonic. Secure storage of mnemonics is hard.
- Individuals have reported losing access to their mnemonic, without having backed up their withdrawal credentials (anecdotal).
- Key management best practices suggest periodically rotating keys, which is not possible today.

The benefits from this feature are twofold:

- Security: Tooling around BLS is not mature yet, and key management is one of the most important problems to get right.
- Future Proofing: It'd make introducing new withdrawal credential types easier, since it'd provide an upgrade path for old credentials to migrate to the newer format (e.g. 0x02 keys for triggering from an eth1 contract)

Protocol Description

1. A transaction is submitted specifying new BLS withdrawal credentials for a validator's public key, signed by the validator's private key
2. A timer of T seconds starts.
3. One of following may happen:
4. If the owner of the withdrawal creds is not the person that initiated this transfer, they can veto it
5. If nobody responds after T seconds, the withdrawal credentials are updated.

Implementation

1. A simple way to implement this would be by introducing a new WithdrawalCredentialsRotation

contract on Eth1, which would implement the protocol and emit an event which validators listen for. Once such an event is observed, the Beacon Node updates its validator object's state to the event's newly specified withdrawal credentials. If the pubkey specified does not match a validator, then the Beacon Node ignores that event.

1. Introduce a new OP_CHANGE_WITHDRAWAL_CREDENTIALS

on the BeaconChain which would allow a validator to update their own credentials and a OP_CANCEL_WITHDRAWAL_CREDENTIALS_CHANGE

which would allow the BLS withdrawal creds to cancel the update.

The T parameter can be set very high, e.g. on the order of 6 or 12 months.

Risks & Tradeoffs

Introducing any additional code in the consensus logic is an overhead that should be minimized. The above 2 solutions should be simple to implement, but if the community is supportive, we should investigate the consensus complexity which will be introduced and weigh it against the benefits.

A grieving vector is introduced whereby if a validator's hot signing key is stolen, it can be used to force the withdrawal

credentials to act before T is over. Ideally, if T is long enough, this should never be an issue.

Deployment

This update can be included in the BeaconChain hardfork after The Merge, when withdrawals will be enabled.