As detailed in the forum post related to the forthcomingdYdX Protocol Upgrade, the chain is about to introduce a new module from the core Cosmos SDK called “AuthZ”.

Source: x/authz | Explore the SDK

x/authz

is an implementation of a Cosmos SDK module, perADR 30, that allows granting arbitrary privileges from one account (the granter) to another account (the grantee). Authorizations must be granted for a particular Msg service method one by one using an implementation of the Authorization

interface.

To give a bit of context for regular Cosmos users, this module makes it possible to use services likeREStake | Auto-compound your staking rewards. Its utility covers a much wider range of potential use cases, including advanced account management, with many more expected to come in the future.

What users are less aware of is the dangerous capabilities this also carries. To explain what those are, let’s have a look at the types of transactions the authZ module can execute:

- MsgGrant: This is the initiation transaction to allow a certain type of action as well as specifying the conditions associated with that (more details later).
- MsgRevoke: To remove an existing Grant.

Therefore, users should pay extra attention to any transaction they sign on their wallet that is related to the MsgGrant

function.

Here is an example of the signature required by Restake to perform their actions on your behalf, in which you can find the aforementioned MsgGrant

as well as all the details associated with that request:

[

restake-example

692×1308 93.1 KB

](https://europe1.discourse-cdn.com/standard21/uploads/dydx/original/2X/7/7eef2a913b1fa6d772e3a11460f08539cb1040c9.jpeg)

At the current time, there is no specific warning sign in the Keplr wallet related to an authz transaction. But it is imperative that users pay an extreme amount of attention to these custom transactions and understand what they are signing. It is unfortunate that protections are not yet in place to prevent users from signing without reading carefully. It will likely happen in the future, but in the meantime, the risk is real, and careless users could face serious harm if they sign malicious authZ transactions without reading.

To understand what a malicious authZ could look like, let’s break down the types of permissions that a MsgGrant

can ask for:

- GenericAuthorization: implements the Authorization

interface that gives unrestricted permission to execute the provided Msg on behalf of granter’s account. YOU SHOULD NEVER SIGN SUCH TRANSACTION

unless being strictly in control of what you’re about to do.

- SendAuthorization: implements the MsgSend

authorization, allowing the grantee to send coins from the granter’s balance. Arguments take a (positive) SpendLimit

that specifies the maximum amount of tokens the grantee can spend and an optional AllowList

that specifies to which addresses a grantee can send token.

- StakeAuthorization: it takes an AuthorizationType

to specify whether you want to authorise delegating, undelegating or redelegating (i.e. these have to be authorised seperately). It also takes a required MaxTokens

that keeps track of a limit to the amount of tokens that can be delegated/undelegated/redelegated. If left empty, the amount is unlimited. Additionally, this Msg takes an AllowList

or a DenyList

, which allows you to select which validators you allow or deny grantees to stake with.

Now you have all the necessary information to understand what an authZ transaction can do. It can be used for great purposes, but if left unchecked and signed carelessly, any malicious actor could be empowered with enormous capabilities over your wallet. It could spend your coins or do even more than that if you ever grant an unwanted GenericAuthorization

to a scammer.

As we wait until wallets implement necessary warnings regarding authZ transaction signatures, it is up to the users to be extra careful themselves.

We hope this quick tutorial will help users make a good use of this powerful module and stay clear of the risks that come along with it.

Thanks for reading,

Govmos (the governance arm of the PRO Delegators' Validator)

[

](https://dydx.forum/t/validator-introduction-pro-delegators/1487)