Integration with SafeAuth

This guide demonstrates creating an externally-owned account using your email or social media account. Once authenticated, you can sign transactions and interact with your Safe accounts.

TheSafeAuthPack is an authentication system that utilizes theWeb3Auth(opens in a new tab) MPC technology. It was developed in collaboration with Web3Auth to create a smooth onboarding experience for web2 users across different dapps.

Prerequisites

- Node.js and npm(opens in a new tab)

Install dependencies

yarn

add

@safe-global/auth-kit

@web3auth/safeauth-embed

Create a SafeAuthPack instance

We will use theSafeAuthPack exported from the@safe-global/auth-kit package.

Create an instance of theSafeAuthPack(opens in a new tab) using the requiredSafeAuthConfig configuration object.

Supported networks:

- Production: Ethereum, Polygon, BSC, Avalanche, Optimism, Celo, Arbitrum, Gnosis chain
- Test: Sepolia, Polygon Mumbai, BSC Testnet, Avalanche Testnet, Arbitrum Testnet, Optimism Testnet

import { SafeAuthPack , SafeAuthConfig , SafeAuthInitOptions , } from

'@safe-global/auth-kit'

const

safeAuthConfig :

SafeAuthConfig

= { txServiceUrl :

'https://safe-transaction-mainnet.safe.global' , } const

safeAuthInitOptions :

SafeAuthInitOptions

= { enableLogging :

true , showWidgetButton :

false , chainConfig : { chainId :

'0x1' , rpcTarget :

{ rpcUrl } } , }

// You can also pass the SafeAuthConfig as a parameter to the SafeAuthPack constructor if you are using a custom txServiceUrl domain // e.g. const safeAuthConfig: SafeAuthConfig = { // txServiceUrl: 'https://safe-transaction-mainnet.safe.global' // } const

safeAuthPack

=

new

SafeAuthPack (safeAuthConfig) await

safeAuthPack .init (safeAuthInitOptions)

Sign in to an Ethereum account

After creating yourSafeAuthPack instance, initiate the authentication process by calling thesignIn() method. Typically, this method is called when the user clicks a "Sign In" button on the web page.

After successfully signing in, you will create a new Ethereum Wallet. This wallet will be used for all future logins and can beshared across different applications .

// The signIn() method returns the user's Ethereum address and the associated Safe addresses // Theawait will last until the user is authenticated. Therefore, it will be active while the authentication popup is being displayed. const

authKitSignData

=

await

safeAuthPack .signIn ()

The returnedauthKitSignData data contains the following properties:

AuthKitSignInData { eoa : string // The safe signer safes ?: string[] // The list of associated Safe addresses in the chain }

ThesignOut() method removes the current session.

await

safeAuthPack .signOut ()

After the user is authenticated, callgetProvider() to get the Ethereum provider instance. This is a EIP-1193(opens in a new tab) compatible provider you can wrap using your favorite library (web3, ethers).

safeAuthPack .getProvider ()

We offer two methods for listening to events,subscribe() andunsubscribe() .

const

accountChangedHandler

= (accounts :

string []) => { console .log ('Signer accounts:' , accounts) }

safeAuthPack .subscribe ('accountsChanged' , accountChangedHandler) safeAuthPack .unsubscribe ('accountsChanged' , accountChangedHandler)

TheSafeAuthPack instantiation will return the list of associated Safe addresses as part of the response from thesignIn() method when thetxServiceUrl is provided.

const

safeAuthPack

=

new

SafeAuthPack ()

Signing and executing transactions using the SafeAuthPack and Protocol Kit

TheSafeAuthPack can be used with theProtocol Kit to establish a connection to a Safe. This connection is made using theprovider andsigner associated with the authenticated account.

After connecting, you can use any of the methods provided in theProtocol Kit(opens in a new tab) .

import { ethers } from

'ethers' import { EthersAdapter } from

'@safe-global/protocol-kit'

// Wrap EIP-1193 provider with ethers const

provider

=

new

ethers .BrowserProvider (safeAuthPack .getProvider ()) const

signer

=

provider .getSigner ()

// Create the Safe EthersAdapter const

ethAdapter

=

new

EthersAdapter ({ ethers , signerOrProvider : signer || provider , })

// Instantiate the protocolKit const

protocolKit

=

await

Safe .create ({ ethAdapter , safeAddress , })

// Create a Safe transaction with the provided parameters const

safeTransactionData :

MetaTransactionData

= { to :

{ ethAddress } , data :

'0x' , value :

ethers .parseUnits ('0.0001' ,

'ether') .toString () , }

const

safeTransaction

=

await

protocolKit .createTransaction ({ transactions : [safeTransactionData] , })

// Sign the transaction if the Safe have several owners // safeTransaction = await
protocolKit1.signTransaction(safeTransaction) // safeTransaction = await protocolKit2.signTransaction(safeTransaction)

// Execute the transaction await

protocolKit .executeTransaction (safeTransaction)

Sign messages using theSafeAuthPack

You can also sign any arbitrary message or transaction as a regular Signing Account with your favorite web3 library:

// Using web3 const

web3

=

new

Web3 (safeAuthPack .getProvider ())

await

web3 . eth .sendTransaction (tx) await

web3 . eth .signTransaction (tx) const

message

=

'hello world' const

address

=

'0x...' await

web3 . eth . personal .sign (message , address)

// Using ethers const

provider

=

new

ethers .BrowserProvider (safeAuthPack .getProvider ()) const

signer

=

provider .getSigner ()

await

signer .sendTransaction (tx) await

signer .signTransaction (tx) await

signer .signMessage (message)

Examples

- React(opens in a new tab)
- Vanilla Typescript(opens in a new tab)