Nonce Management

Normally, a wallet can calleth_getTransactionCount to get the next nonce to use for a transaction. However, since transactions sent to Flashbots Protect are potentially sensitive, even exposing the incremented nonce can leak information about the user's activity.

As such, transactions sent to Flashbots Protect are only included in theeth_getTransactionCount results when querying the"pending" nonce, and only if the request is signed by the user's private key.

This is done by sending a JSON-RPC request to the Flashbots Protect RPC endpoint with the following parameters:

{ "jsonrpc": "2.0", "method": "eth_getTransactionCount", "params": ["0xYOUR_ADDRESS", "pending"], "id": 1 } The request is then signed and the signature is included in theX-Flashbots-Signature header. Without such a signature, the returned nonce will only include transactions sent to the public mempool.

Authentication

To authenticate your request, sign the payload and include the signed payload in theX-Flashbots-Signature header of your request.

curl -X POST -H "Content-Type: application/json" -H "X-Flashbots-Signature: :" --data '{"jsonrpc":"2.0","method":"eth_getTransactionCount","params":["0xYOUR_ADDRESS","pending"],"id":1}' https://rpc.flashbots.net The private key of the address your want to query must be used to sign the payload.

The signature is calculated by taking theEIP-191 hash of the json body encoded as UTF-8 bytes. Here's an example using ethers.js:

- ethers.js
- web3.py
- go

import

{ Wallet , utils }

from

'ethers' ;

const privateKey =

'0x1234' ; const wallet =

new

Wallet (privateKey) ; const body = '{"jsonrpc":"2.0","method":"eth_getTransactionCount","params":["0xYOUR_ADDRESS","pending"],"id":1}' ; const signature = wallet . address +

':'

+ wallet . signMessage (utils . id (body)) ; from web3 import Web3 from eth_account import Account , messages

body

'{"jsonrpc":"2.0","method":"eth_getTransactionCount","params":["0xYOUR_ADDRESS","pending"],"id":1}' message = messages . encode_defunct (text = Web3 . keccak (text = body) . hex ( )) signature = Account . from_key (private_key) . address +

':'

+ Account . sign_message (message , private_key) . signature . hex ( ) body :=

{"jsonrpc":"2.0","method":"eth_getTransactionCount","params":["0xYOUR_ADDRESS","pending"],"id":1} hashedBody := crypto . Keccak256Hash ([ ] byte (body)) . Hex ( ) sig , err := crypto . Sign (accounts . TextHash ([ ] byte (hashedBody)) , privKey) signature := crypto . PubkeyToAddress (privKey . PublicKey) . Hex ( )

+

":"

+ hexutil . Encode (sig)