Neon EVM

Ethereum Smart Contracts Scaled by Solana
(version 1.2., 16 September 2021)

Neon EVM is an Ethereum virtual machine on Solana that enables dApp developers to use Ethereum tooling to scale and get access to liquidity on Solana.

Abstract

Ethereum remains the dominant blockchain protocol linked to smart contract trading and settlement. Its infrastructure for dApp developers and end-users is the most advanced.

This paper introduces Neon EVM: a tool that allows for Ethereum-like transactions to be processed on Solana, taking full advantage of the functionality native to Solana, including parallel execution of transactions. As such, Neon EVM allows dApps to operate with the low gas fees, high transaction speed, and high throughput of Solana, as well as offering access to the growing Solana market.

The Ethereum state is represented by Merkle-Patricia Trie that stores key-value data for all smart contracts, and smart contracts written in Solidity do not have separate references to shared data and contracts' code. Therefore, these smart contracts have to be executed in sequence to ensure deterministic behavior. This limits a throughput: highly optimized blockchains with EVM are capable of processing up to a maximum of 1,500 TPS.

However, Solana is designed to support massive scaling of decentralized applications, with a maximum throughput of more than 50,000 TPS. To take full advantage of Solana's functionality, Neon EVM is built as a smart contract of Solana. This also ensures flexibility in terms of updates: Neon EVM can be updated easily when new Ethereum functionality appears.

In the case of Neon EVM, an intermediary proxy server that can be run by anybody wraps up Ethereum-like transactions into Solana transactions and sends them to Solana for parallel execution of Neon EVM contract. To ensure the parallel execution of smart contracts, Neon EVM implements several strategies. In particular, each contract keeps its data in its own Solana storage and account balances used to pay for Neon transactions are also separated.

The solution allows any Ethereum application to be run on Solana without any changes to its codebase, including UniSwap, SushiSwap, 0x, and MakerDAO. All key tools for Ethereum dApps can work on Solana, including Solidity, Metamask, Remix, Truffle, and others.

Neon EVM is best suited to developers that want to enjoy a first-mover advantage and reach new customers on Solana, or those who want to scale with the low gas fees and high throughput that Solana provides. It is also good for those developers who are looking to tap into liquidity on Solana.

Introduction

Ethereum is set to remain among the booming blockchain ecosystems. The number of active dApps on Ethereum is hovering above 300 and the number of active users of these dApps is close to 6 million, with the number of transactions on the rise. Ethereum's popularity is not only down to its processing of smart contracts, but also thanks to its sophisticated infrastructure for application development.

Solana is one of the most technically advanced blockchains, offering low gas fees and high throughput of transactions due to its technological innovations. Among these innovations is its Proof-of-Stake consensus system that is reinforced via a Proof-of-History protocol, a transaction parallelization technology that optimizes resources and ensures that Solana can scale horizontally across GPUs and SSDs, along with an optimized mempool system that speeds up throughput.

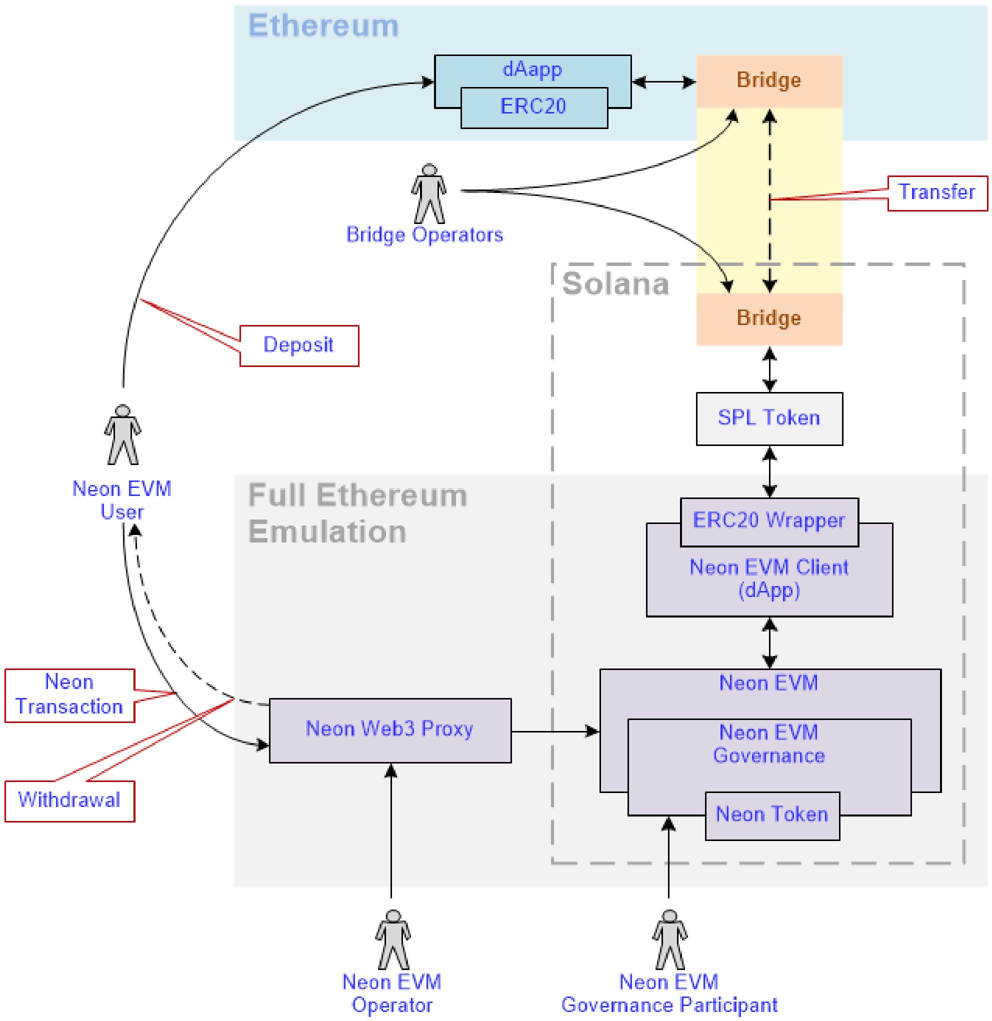
Neon EVM is a cross-chain solution that allows dApp developers to access the advantages of Solana and thus expand their services: to offer new products like arbitrage or high-frequency trading, grow their user base, and decrease costs where possible, including gas fees. It enables full compatibility with Ethereum on Solana.

Neon EVM Fundamentals

Solana blockchain has its own Berkeley Packet Filter virtual machine (BPF virtual machine). This virtual machine is used in the Linux kernel and has already been tested through time. BPF bytecode was originally designed for fast execution. Solana supports just-in-time compilation for BPF bytecode, which significantly increases the speed of execution of BPF contracts. Neon EVM smart contract is written in Rust and compiled to BPF bytecode. This allows us to take full advantage of Solana functionality, including parallel execution of transactions. It also makes it easy to update Neon EVM regardless of Solana's hard forks.

Let's take a deeper look at the technical solution that Neon EVM offers.

Neon EVM Architecture



Technical definitions:

- **Neon EVM** is an Ethereum Virtual Machine which represents a smart contract written in Rust and compiled into Berkeley Packet Filter bytecode of a virtual machine running on Solana.
- **Neon EVM user** is any user who has an account in Neon EVM with a balance in NEON, ERC20, and ERC721 tokens.
- **Neon EVM client** is any application that has an EVM (Solidity/Vyper/etc.) bytecode contract loaded into the Neon EVM on Solana.
- **Neon EVM operator** is any Solana account that pays for the execution of a Neon transaction in SOL tokens and receives payment for this work from the Neon EVM user in an arbitrary token specified by the user.

- **Neon EVM governance** is a decentralized Neon EVM governance that manages Neon EVM work by setting up Neon EVM parameters and updating Neon EVM software; it gets fees for its services.
- **Neon Web3 Proxy** is a tool that can be used by a Neon EVM operator to package a Neon transaction into a Solana transaction.
- **Neon Transaction** is a transaction formed according to Ethereum rules with a signature produced by Ethereum rules.
- **Bridge** is an EVM third-party (independent from Neon) solution with its own operators.

Neon EVM has the following functions:
- Uploading EVM contracts (built by Solidity/Vyper compilers) to individual Solana accounts.
- Checking signatures according to Ethereum rules on Solana.
- Executing Neon transactions, including, if necessary, in an iterative manner taking into account Solana resource constraints with the financial guarantees for completion of transactions.
- Calculating gas consumption according to Ethereum rules.
- Receiving a payment from the user to Neon EVM operator for the gas consumed and fees in any NEON or any ERC-20 token specified by the user.
- Calculating and withdrawing fees in SOL tokens to the governance pool of Neon EVM from the Neon EVM operator account for execution of Neon transactions.
- Storing EVM data of contracts in the form of a hash table using the Hash Array Mapped Trie algorithm (HAMT).

Neon Web3 Proxy

Neon Web3 Proxy is a service that provides a Web3 API to access the Solana blockchain. It is an intermediary for communication between Neon EVM clients and Neon EVM and it can be run by Neon EVM operators. Neon Web3 Proxy is optional for any Neon EVM client. Its main functionality is to help Neon EVM clients start using Neon EVM without any changes to their codebase.

ERC20 SPL-Wrapper

For each Solana token, an ERC20 SPL-Wrapper contract can be deployed. The task of the ERC20 SPL-Wrapper is to ensure the interaction of the Solana applications with EVM (Solidity/Vyper/etc.) bytecode contracts. ERC20 SPL-Wrapper can be also used to transfer funds in Solana tokens using Ethereum wallets such as Metamask.

ERC20 SPL-Bridge

This contract is for ERC20 contracts. When it’s called, it generates a Solana token which represents the corresponding ERC20 token in the SPL-token contract. The Solana tokens registered in the SPL-token contract can be transferred to Solana contracts.

Independence of Operations Within Neon EVM

Neon EVM ensures the independence of its operations by providing open access to its infrastructure to anybody who is willing and capable of running Neon Web3 Proxy. Moreover, Neon Web 3 Proxy can be replaced with a client library by any Neon EVM client. The transactions received by Neon EVM cannot be discriminated against because they do not have any attributes that determine their priority. The unchangeable nonce and user signature fields verified by Neon EVM guarantee the consistency of execution of Neon transactions and protect from re-execution.

Preconditions for the Execution of Neon Transactions

As with other Solana-native dApps, in order to enable the execution of Neon transactions, the user must grant Neon EVM access to their accounts. Neon EVM then acts on behalf of the user for user transactions such as transfers, swaps, or other operations. It verifies the information received (the nonce field and the signature), and enables the operation that is asked for.

To prevent forgery and unlawful operations, the following fields are checked and verified:

- The nonce field: it has a unique transaction index, which is verified by the Neon EVM smart contract. This makes double-spend attacks impossible.
- The signature field: it is formed according to the Ethereum rules. The Neon transaction signature is verified by the Neon EVM smart contract. Note: the procedures for validating the Solana signature and the Neon signature are different and are implemented using different algorithms.

Neon EVM access to user accounts is secured in the following ways:

- Neon EVM smart contract code is freely available for anyone to review. It is being audited in September, 2021.
- Any user can delegate access to their Neon EVM smart contract account to any independent Solana account. This can prevent any malicious actor from withdrawing user funds in case of an attack on Neon EVM or its governance. In case of attack, users who enable this feature will be able to withdraw their tokens to their Solana accounts.
- Neon transactions are validated by independent Solana validators.
- Decentralised Neon EVM governance is responsible for the updates of the Neon EVM contract.

Parallel Execution of Neon Transactions on Solana

Most blockchains process transactions in a single thread, meaning that the blockchain state is modified by one contract at a time. In contrast, Solana can process tens of thousands of contracts in parallel, using as many cores as are available on a Solana node. This functionality is known as Sealevel, and it greatly increases the throughput.

Parallel processing is possible because Solana transactions describe all the states a transaction will read or write while executing. This prevents transactions from overlapping, which allows independent transactions and those that are reading the same state to be executed concurrently.

Neon transactions are executed by Solana as native transactions in parallel while restricting access to shared data from the Solana state. However, in some cases a Neon transaction requires more resources than Solana allocates for one transaction. In this case, the Neon EVM executes the transaction iteratively, and the extended mode of restricting access to shared data in the Solana state is used. (For details, see the section on iterative execution of Neon transactions.)

To ensure the parallel execution of Solana transactions, Solana requires a list of all Solana accounts involved in a transaction. If there is a call to a Solana account that is not specified in the header of the Solana transaction, the algorithm aborts the execution with an error.

The procedure for parallel execution of Neon transactions consists of the following steps:

1) Neon Web3 Proxy, which has a built-in EVM similar to Neon EVM, receives a Neon transaction from the user.
2) Neon Web3 Proxy performs a test launch of the Neon transaction, calling the Solana node for the current state via a public Solana cluster of RPC endpoints or its own Solana node.
3) As a result of the test performed, Neon Web3 Proxy receives a complete list of contracts and accounts involved in the Neon transaction.
4) Neon Web3 Proxy forms a Solana transaction using a list of Neon contracts and Neon accounts, inside which the Neon transaction is wrapped.
5) Neon Web3 Proxy sends the Solana transaction to on-chain execution to the Solana cluster.
6) The Solana cluster gets the Solana transaction and sends it for execution to the leading Solana node.().
7) The concurrent transaction processor of the Solana node) executes Solana transactions -in parallel, checking the independence by verifying the Solana accounts in the header of the Solana transaction.
8) Solana transactions with the packed Neon transactions are executed in parallel, calling Neon EVM smart contract in the following manner:
    a. Neon EVM smart contract is loaded.
    b. EVM (Solidity, Vyper, etc.) bytecode smart contract is loaded.
    c. Each EVM (Solidity, Vyper, etc.) bytecode smart contract executed on Solana has its own independent state.
    d. The Neon EVM smart contract executes any Neon transaction by calling EVM smart contract method.

e. When the Neon transaction is executed on-chain, data from the Solana state is used and changed.
f. At the end of execution of the Neon transaction, the Neon EVM updates the Solana state.

Acceleration of Neon Transaction Execution

As noted above, Neon Web3 Proxy performs a test run to obtain a complete list of Neon accounts that are used for the execution of Neon transactions. A test run takes time, and this time could be critical when a transaction needs to be executed fast.
Any Neon transaction can be executed without a test run in the following manner:

- The Solana transaction is built on the client side (web/mobile) with a Neon transaction packaged within it. The Solana transaction is sent directly to a Solana node without Neon Web3 Proxy. It's important to understand that, using this method, it's up to the sender to make sure that:
    - In cases when the Neon transaction is too big, it has to be executed iteratively (see more in the section on iterative execution of Neon transactions below).
    - A list of all Neon accounts and contracts corresponding to the Neon transaction has to be determined on the client-side.
- Additional methods in the Neon Web3 Proxy with a transfer of the list of Neon accounts involved can be used. It's important to understand that — using this method — it's up to the sender to make sure that a list of all Neon accounts and contracts corresponding to the Neon transaction has to be determined on the client side.

Iterative Execution of Neon Transactions

Solana blockchain limits the resources allocated to the execution of a single transaction to ensure optimal usage of hardware. To perform the best service possible (taking into account the existing restrictions of Solana), Neon EVM introduces iterative execution of Neon transactions.
The main steps of iterative execution are the following:

- Neon EVM transfers the deposit in SOL tokens from the operator's account to a separate account.
- The size of the deposit is determined by the Neon EVM settings set by Neon EVM governance. The deposit consists of:
    - A reward to Solana validators for executing Solana transactions.
    - A fee to the Neon EVM governance that goes to the governance pool.
    - A fee for iterative execution of the Neon transaction that is blocked.
- Neon EVM blocks Solana accounts used in Neon transactions.
- If any Solana accounts are already blocked by another Neon transaction, then the new transaction is queued for execution by Neon Web3 Proxy.

Neon EVM settings set by Neon EVM governance:

- *The maximum number of iterations per Neon transaction*. Solana currently charges a fee to verify the signatures specified in a Solana transaction. Thus, in a Solana transaction, only the Solana signature of the Neon EVM operator in charge of the transaction is specified. All Neon signatures are verified by the Neon EVM during the execution of Neon transactions. The number of iterations per Neon transaction is unknown in advance. It is necessary to limit the execution time of a transaction because all accounts and contracts involved in this transaction will be blocked for use in other Neon transactions. Therefore, Neon EVM governance sets the maximum number of iterations and the maximum number of waiting blocks (Mn). The size of the deposit directly depends on the maximum number of iterations.
- *A fee to the Neon EVM governance pool* (see the section on Neon EVM economy and governance).
- *A deposit for the iterative execution of a Neon transaction*:
    - Depends on the number of accounts involved in the transaction, as this affects the parallel execution of transactions in Solana.
    - Is paid to the operator who performs the last step of the Neon transaction and finalizes it.
- *The maximum number of waiting blocks (Mn) is determined by Neon EVM governance.* The operator is given a maximum of blocks (Mn) between two iterations when it can perform the next iteration. After Mn blocks, any other operator can continue the execution and receive the deposit.

The formula for calculating the deposit:

- **Mn**: maximum number of waiting for blocks between two iterations for one operator, after which the transaction can be continued by another operator.
- **Mi**: maximum number of iterations per Neon transaction.
- **Li**: number of SOL to check one Solana signature.
- **Na**: number of accounts used in Solana transaction.
- **Fa**: deposit in SOL for the iterative execution to be paid to each Solana account that is specified in a transaction.
- **Fg**: fee in SOL to the Neon EVM governance pool.
- **Rs**: resulting deposit in SOL.

$$Rs = Mi * Li + Fa * Na + Fg$$

The main steps for iterative execution of a Neon transaction are the following:

- If it’s not the first iteration, then:
    - If more than Mn blocks have passed, the execution is passed to another operator.
    - Restore the state of the Neon EVM.
- Complete the maximum EVM steps specified in the Solana transaction.
- If it’s not the end of execution:
    - Record the operator which completed a step in iterative execution.

- Increase the number of completed iterations.
- Save the state of the Neon EVM to the state of Solana.

The conditions to end the iterative execution of a Neon transaction:

- The Neon transaction is completed.
- Mi iterations of Neon transactions have been reached.
- The Neon transaction was canceled; in this case, the unspent deposit is burned. Neon transactions can be canceled:
  - By any Neon EVM operator if Mn blocks have not passed from the last iteration.
  - Otherwise, by the Neon EVM operator from the last iterative execution.

At the end of the iterative execution of a Neon transaction:

- The result of the Neon transaction is saved into the Solana receipt.
- The Neon EVM operator is paid the remaining deposit.

The formula for calculating the remaining deposit:

- **Mi**: maximum number of iterations per Neon transaction.
- **Md**: the number of iterations used to execute a Neon transaction.
- **Li**: the number of SOL to check one Solana signature.
- **Na**: the number of accounts in the Solana transaction.
- **Fa**: the amount of the deposit in SOL for iterative execution for one account in the Solana transaction.
- **Rd**: the resulting remaining deposit in SOL.

$$Rd = (Mi - Md) * Li + Fa * Na$$

Payment for Neon Transactions

The cost of transaction execution in Neon EVM is calculated according to Ethereum rules using a gas calculation tool available on Ethereum.

The cost of gas in Neon EVM is significantly lower than on Ethereum, as it is based on Solana's gas price. It is determined by taking into consideration:

- The cost of executing a Solana transaction, which depends on the number of signatures specified in the transaction. As previously mentioned, in Solana transactions, only one signature is specified (that of the Neon EVM operator) because Neon signatures are verified in Neon EVM. (See the above section on iterative execution of transactions.)
- A fee to Neon EVM governance to keep Neon EVM running.
- A fee to the Neon EVM operator that executes the transaction.

A Neon transaction formed according to the Ethereum rules contains two groups of fields that are used in payment for transactions:

- The first group consists of two fields: gas_price and gas_limit. These fields in the Ethereum network are used to pay for the gas spent on the transaction to the miner that produces a block. There are no miners in the case of Neon EVM, but the Neon operators are running proxies and sending Neon transactions to the Solana cluster and are rewarded for the work done. The Neon user can pay for gas in NEON tokens or ERC-20 tokens that the Neon operator agrees to accept as payment.
- The second group consists of a field value. This field is used on Ethereum for two purposes: first, to enable the transfer of ETH tokens. Second, to enable payment to the Ethereum smart contracts according to the functionality that is used by a user. Neon users will be able to transfer only NEON tokens as a value token to prevent issues with the execution of Ethereum smart contracts deployed in Neon EVM.

The payment method is to be determined by the Neon EVM user:

- By default, the Neon EVM user pays the Neon EVM operator in NEON tokens for gas.
- However, Neon EVM gives users an option to choose any other ERC-20 token to pay for Neon transactions.
- The payment preferences are stored in the Solana state.

There is an independent market for Neon EVM operators, and a user can choose an operator with reasonable prices for transactions. Neon EVM clients can enforce their payment policies based on their demands. Any Neon EVM user can independently deploy Neon Web3 Proxy and execute a Neon transaction using Neon EVM without help from Neon EVM operators. In this case, the user has to cover the transaction execution on Solana with SOL tokens.

Neon EVM Economy and Governance

The Neon EVM economy is fee-based. The NEON token is an utility token that is needed to pay for the execution of Neon transactions. Later, it will be used for governance purposes. At the launch, the Neon governance will be handled by a multisig account whose keys are held by reputable individuals and entities with a vested interest in the success of Neon EVM, and with a clear and user-friendly process for proposing and voting for protocol improvements. Later, a decentralised governance protocol will be introduced.

Summary

Neon EVM is a viable solution for anyone who is looking to scale Ethereum dApps on Solana in a developer-friendly manner. To take full advantage of Solana's functionality, Neon EVM is built as a smart contract of Solana. This also ensures flexibility in terms of updates. Neon EVM has a high level of decentralization as it is governed with a help of a multisig account to be replace later for the decentralised governance protocol and any user can set up Neon Web3 Proxy and receive payments for executing transactions via Neon EVM.

Neon EVM is a full Ethereum emulation with gas consumption calculated by Ethereum rules, iterative execution of large Ethereum EVM contracts, and full compatibility with all existing Ethereum tools. It enables parallel execution of EVM (Solidity/Vyper/etc.) bytecode contracts on Solana. It also provides access to Solana infrastructure via Ethereum tools and access to native Solana tokens, registered in the SPL-token contract through the ERC20 token interface, native to contracts and tools made for Ethereum. Users can pay for the services of Neon EVM in NEON tokens.

Disclaimer

The information outlined in this White Paper may not be exhaustive and does not imply any elements of a contractual relationship. The content of this White Paper, including that of the website https://neonlabs.org/, is not binding for us and our affiliates and we reserve a right to change, modify, add, or remove portions of this White Paper for any reasons at any time without prior notification. We also reserve a right to annul this White Paper at any time with a prior notification.

This White Paper is designed for general informational purposes only, as a guide to certain of the conceptual considerations associated with the narrow issues it addresses. This White Paper shall not be reproduced, copied, transferred, or otherwise distributed to any third party. While we make efforts to ensure that all information in this White Paper is accurate and up-to-date, any information herein is not professional advice. We also make no representations or warranties regarding the accuracy, reliability, relevance, and completeness of any information herein.

This White Paper is aimed only on the description of Neon EVM without expression of any opinion or promise on its feasibility, investment attractiveness, and (or) relevance. We do not guarantee that the final product will meet your expectations. You should accept all risks related to operations with Neon EVM on your own prior to using the information herein. Information in the White Paper is based on publicly available information and shall not be deemed as a separate investigation. This White Paper does not constitute an investment, legal, tax, regulatory, financial, accounting, or other advice. Nothing in this White Paper shall be deemed to constitute a prospectus of any sort or a solicitation for investment, nor does it in any way pertain to an offering or a solicitation of an offer to buy any securities in any jurisdiction.

Each person who reads this White Paper is reminded that this White Paper has been presented to him/her on the basis that he/she is a person into whose attention the document may be lawfully presented in accordance with the laws of his or her jurisdiction.

Certain statements in this White Paper may constitute forward-looking statements, which are usually identified by the use of words as “is expected”, “believes”, “expects”, “supposedly” “supposes” or similar phrases and statements. Such forward-looking statements or information may be affected by known and unknown risks and uncertainties, which may influence materially on estimates or results implied or expressed in such forward-looking statements. We do not undertake obligations on updating and review of such statements and information. We do not accept any liability for any losses that can be incurred due to use or reliance on such statements. We have not conducted any independent verification in relation to White Paper or any issue reflected herein.
You shall use this White Paper at your own risk. In the absence of express consent, we do not accept any liability for any losses that can be incurred as a result of such usage, including direct and indirect consequences.

This White Paper is not legally binding, does not oblige anyone to execute any agreements or undertake any obligations.
YOU ARE ENTITLED AND ENCOURAGED TO ASK QUESTIONS AND REQUEST NECESSARY INFORMATION. FOR THESE AND ANY OTHER REASONS PLEASE CONTACT US AT info@neonlabs.org.