Here is a draft proposal for monkeypatching EIP-2612 support on ERC-20. This is by no means a complete proposal.

Summary

ApprovalTarget is an immutable contract that accepts token approvals from users across various ERC-20s, then enables anyone with a valid user signature to transfer those tokens. You can think of it like monkeypatching EIP-2612 support on all ERC-20 compliant tokens.

Abstract

To use ERC-20 tokens in dApps, approving is needed, which is expensive.

Design a ApprovalTarget contract to create a single signed approval flow for ERC-20 tokens. This allows users to send approval transaction only once.

Motivation

EIP-2612 allows abstraction in the ERC-20 approve

method. But some of ERC-20 tokens do not have a permit

function. Introducing ApprovalTarget indirectly allows approval by signature for all ERC-20 compliant tokens.

In addition ApprovalTarget replaces approve

transaction with a signed approve and will improves UX.

Specification

ApprovalTarget has three functions.

function nonces(address owner) external view returns (uint)

function PERMIT_AND_TRANSFER_FROM_TYPEHASH() external view returns (bytes32)

function permitAndTransferFrom(address erc20, address owner, address recipient, uint256 value, uint256 deadline, uint8 v, bytes32 r, bytes32 s) external

The semantics of which are as follows:

For all addresses erc20

, owner

, recipient

, uint256s value

, deadline

and nonce

, uint8 v

, bytes32 r

and s

, a call to permitAndTransferFrom(erc20, owner, recipient, value, deadline, v, r, s)

will call transferFrom(owner, recipient, value)

, increment nonces[owner]

by 1. If and only if the following conditions are met:

- The current blocktime is less than or equal to deadline

.

- erc20

is not the zero address. this address must be ERC-20 compliant token address.

- owner

is not the zero address.

- nonces[owner]

(before the state update) is equal to nonce

.

- r

, s

and v

is a valid secp256k1

signature from owner

of the message:

NOTE: spender

in the provided signature must be the same as msg.sender

. recipient

isn’t attested by a signature from owner

, and is instead chosen by msg.sender

, attested in the signature as spender

.

If any of these conditions are not met, the permitAndTransferFrom

call must revert.

keccak256(abi.encodePacked(hex"1901", DOMAIN_SEPARATOR, keccak256(abi.encode(keccak256("PermitAndTransferFrom(address erc20,address owner,address spender,value,uint256 nonce,uint256 deadline)"), erc20, owner, msg.sender, // NOTE: spender value, nonce, deadline))))

where DOMAIN_SEPARATOR

is defined according to EIP-712.

In summary the caller of the permitAndTransferFrom

function must be spender

. spender

can choose recipient

.

Rationale

The spender

is not provided in permitAndTransferFrom

paramters. If spender

is not msg.sender

, the transaction will revert.

The recipient

is provided by not the owner

but the spender

. So, the recipient

isn’t attested by a signature.

More details here GitHub - massun-onibakuchi/approval-target