ERC20 Token Allowances

Justus Perlwitz

2018-04-13

Contents

Have you ever wondered how your users can spend ERC20 platform tokens to buy assets and other tokens from you? Or — in more technical terms — how to make sure a smart contract executes a particular function as soon as it receives tokens? This post is all about using ERC20 `approve()` and `transferFrom()` to allow your users to do much more than just keep tokens in a wallet.

But before jumping into the nitty-gritty, I'd like to present a common use case. Together, we will find a good way to implement it with smart contracts.

> Alice wants to pay 20 Eve Tokens (ET) to Bob's Lemon Juice Stand and receive 20 Lemon Juice Tokens (LJT) in return.
>
> Bob can only give Alice her LJT if he can verify that he has received the correct amount of ET from her in a single transaction. For that, the smart contract needs proof of payment.

Let's assume that Alice has a token wallet at `alice_address`, and Bob's Lemon Juice Stand has a smart contract called `bob` at `bob_address`. There's also an ET contract contract called `eve` and a LJT contract called `ljt`.

If Alice uses `eve.transfer(bob_address, 20 ether)`, the amount will appear at Bob's Lemon Juice Stand address and it can be verified with `eve.balanceOf(bob_address)`. His balance will have increased by 20 ET. But just checking `eve.balanceOf(bob_address)` won't allow Bob to check where the tokens came from. So how can Alice prove that she transferred the tokens?

To put it another way, Bob's Lemon Juice Stand's ET balance could also increase by 20 ET if two people send each him 10 ET. There is simply no functionality in an ERC20 compatible token that allows someone to trace the origin of the funds. We have to find a different way to achieve this while still staying in the realm of smart contracts — which is also called *on-chain*. Let's take a closer look at what that means.

The term *on-chain* describes all the functionality that can be implemented using smart contracts on the Ethereum Virtual Machine. *Off-chain* describes anything that is not run on the Ethereum Virtual Machine but still uses the Ethereum block chain as a data source and verifiable transaction log.

If we want to verify that Alice's payment took place using a program — implemented off-chain — there are a few options. We can easily listen for ET ERC20 events using, for example, a web3.js filter. The two possible events are `Transfer` and `Approval`, as is documented here. Assuming that our contract emits an event called `Transfer(from, to, amount)`, we can listen for any event matching `Transfer(alice_address, bob_address, 20 ether)` and we can be immediately notified when Alice transfers her tokens.

Inside a smart contract, we can't listen for events like this. We need to look for an alternative solution that allows us to send tokens and executes some piece of smart contract code in Bob's Lemon Juice Contract all in one transaction on-chain.

For that we need to use `eve.approve(receiver, amount)` and `eve.transferFrom(from, amount)`. With `eve.approve()`, we can give someone permission to withdraw a certain amount of tokens from our address. This is called an allowance. Bob must then have a function in his smart contract called `bob.giveToken()` that does the following:

1. Check the token allowance `eve.allowance(msg.sender)` (`msg.sender` is `alice_address` in this example).
2. Calculate the correct amount of LJT to be paid out.
3. Ensure that Bob's Lemon Juice Stand has enough tokens by checking `ljt.balanceOf(bob_address)`.
4. Call `eve.transferFrom(alice_address, 20 ether)` and Assert that the call was successful.
5. Call `ljt.transfer(alice_address, 20 ether)` and Assert that the call was successful.

If Bob's Lemon Juice Stand contract provides the `bob.giveToken()` function, the process is easy. If Alice now wants to receive LJT in return for her ET, the following two transactions are necessary:

1. Alice calls `eve.approve(bob_address, 20 ether)`. No token balances will change.
2. Alice calls `bob.giveToken()`. If the transaction is validated, Alice now has 20 more LJT and 20 fewer ET.

So we can't solve this problem in less than two transactions. With ERC20, there is no way to transfer tokens and call smart contract functionality at the same time. When it comes to token transactions, simple and secure is better. Newer standards such as ERC223 or EIP777 provide this functionality.