ERC20 API: An Attack Vector on the Approve/TransferFrom Methods

0. Abstract

In this article we describe a possible attack vector on the standard ERC20 Ethereum Token API. This is an attack on the API itself, not on any particular implementations, so all conformant implementations are potentially vulnerable. The attack uses the methods "approve" and "transferFrom" defined by ERC20. We also give some thoughts about how the described attack could be prevented or at least mitigated using the current version of the ERC20 API. We also suggest changes to the ERC20 API that would make the described attack impossible.

1. Introduction

ERC20[1], defines a standard API for Ethereum Tokens smart contracts. Tokens are defined by Ethereum Foundation[2] as the following:

> *Tokens in the ethereum ecosystem can represent any fungible tradable good: coins, loyalty points, gold certificates, IOUs, in game items, etc. Since all tokens implement some basic features in a standard way, this also means that your token will be instantly compatible with the ethereum wallet and any other client or contract that uses the same standards.*

So, ERC20 is supposed to be the standard way to implement basic features of all tokens making them compatible with common Ethereum software such as Ethereum Wallet.

2. Approve/TransferFrom Methods

Among other things, ERC20 defines the following two methods to be implemented by every Ethereum Token smart contract:

```
function transferFrom(address _from, address _to, uint256 _value)
returns (bool success)
```

> *Send `_value` amount of tokens from address `_from` to address `_to`*
> *The `transferFrom` method is used for a withdraw workflow, allowing contracts to send tokens on your behalf, for example to "deposit" to a contract address and/or to charge fees in sub-currencies; the command should fail unless the `_from` account*

[1] https://github.com/ethereum/EIPs/issues/20
[2] https://www.ethereum.org/token

has deliberately authorized the sender of the message via some mechanism; we propose these standardized APIs for approval:

```
function approve(address _spender, uint256 _value)
returns (bool success)
```

Allow `_spender` to withdraw from your account, multiple times, up to the `_value` amount. If this function is called again it overwrites the current allowance with `_value`.

Apart from updating the allowance, the ERC-20 proposal does not specify the intended semantics of multiple calls to `approve`.

In the following sections we will show how these two methods, as defined in ERC20, could be used in an attack that allows a spender to transfer more tokens than the owner of the tokens ever wanted to allow the spender to transfer.

3. Attack Scenario

Here is a possible attack scenario:

1. Alice allows Bob to transfer N of Alice's tokens ($N > 0$) by calling the `approve` method on a Token smart contract, passing the Bob's address and N as the method arguments
2. After some time, Alice decides to change from N to M ($M > 0$) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve` method again, this time passing the Bob's address and M as the method arguments
3. Bob notices the Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom` method to transfer N Alice's tokens somewhere
4. If the Bob's transaction will be executed before the Alice's transaction, then Bob will successfully transfer N Alice's tokens and will gain an ability to transfer another M tokens
5. Before Alice noticed that something went wrong, Bob calls the `transferFrom` method again, this time to transfer M Alice's tokens.

So, an Alice's attempt to change the Bob's allowance from N to M ($N > 0$ and $M > 0$) made it possible for Bob to transfer $N + M$ of Alice's tokens, while Alice never wanted to allow so many of her tokens to be transferred by Bob.

4. Attack Analysis

The attack described above is possible because the `approve` method overwrites the current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by certain value atomically, unless token owner is a smart contract, rather than an account[3].

[3] Unlike accounts, smart contracts may perform several operations atomically, i.e. check current allowance and then set new one

5. Workaround

Because the described attack allows an attacker to transfer at most $N + M$ tokens when the allowance is being changed from N to M, then, changing the allowance from N to 0 and then from 0 to M seems quite safe. A token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e. that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Unfortunately, such checking does not seem to be possible via standard Web3 API[4], because to do the check one needs to be able to analyze the changes in the storage of a smart contract made by particular transactions, including internal transactions. Though, such checking is still possible using advanced blockchain explorers such as EtherCamp[5].
Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

6. Suggested ERC20 API Changes

This section suggests changes to ERC20 API that are supposed to make the attack described above impossible.

6.1. Atomic "Compare And Set" Approve Method

We suggest the following method to be added to ERC20 API:

```
function approve(
  address _spender,
  uint256 _currentValue,
  uint256 _value)
returns (bool success)
```

> *If current allowance for `_spender` is equal to `_currentValue`, then overwrite it with `_value` and return `true`, otherwise return `false`.*

This change alone is enough to address the attack vector described above. Suggestions given below are not required, but are supposed to make usage of `approve` and `transferFrom` methods more convenient and less error-prone.

6.2. Separate Log Message for "TransferFrom" Transfers

We suggest the following event to be added to ERC20 API:

---

[4] https://github.com/ethereum/wiki/wiki/JavaScript-API
[5] https://live.ether.camp/

```
event Transfer(
  address indexed _spender,
  address indexed _from,
  address indexed _to,
  uint256 _value)
```

Triggered when tokens are transferred via `transferFrom` method.

Note, that for backward compatibility reasons, token contracts will probably have to log both, old three-args and new four-args `Transfer` events in `transferFrom` method.

6.3. Four-Args Approval Event

We suggest the following event to be added to ERC20 API:

```
event Approval(
  address indexed _owner,
  address indexed _spender,
  uint256 _oldValue,
  uint256 _value)
```

Triggered whenever either `approve` method is called.

Note, that for backward compatibility reasons, token contract will probably have to log both, old three-args and new four-args `Approval` events in `approve` method.

7. Authors

- Mikhail Vladimirov <mikhail.vladimirov@gmail.com>
- Dmitry Khovratovich <khovratovich@gmail.com>