# 1 概念

以太坊改进提案EIP20定义了ERC20代币标准，为代币定义了标准接口。

标准接口提供了转移代币的基本功能，并允许代币获得批准，以便另一个链上第三方可以使用他们。

# 2 ERC20标准API和规范

下面列出的接口和规范使用的是Solidity 0.4.17（或更高版本）的语法，接口使用者必须处理返回的false，不能假设永远不会返回false。

## 2.1 name

返回代币的名称，如“MyToken”。

为可选项，可以用来提高可用性，但是接口和其它合约不能假定它必须存在。

```
function name() public view returns (string)
```

## 2.2 symbol

返回代币的符号，如“HIX”。

为可选项，可以用来提高可用性，但是接口和其它合约不能假定它必须存在。

```
function symbol() public view returns (string)
```

## 2.3 decimals

返回代币的精确度，如“8”代表代币精确到小数点后8位。

为可选项，可以用来提供可用性，但是接口和其它合约不能假定它必须存在。

```
function decimals() public view returns (uint8)
```

## 2.4 totalSupply

返回代币的总供应量。

```
function totalSupply() public view returns (uint256)
```

## 2.5 balanceOf

返回给定的owner地址的代币余额。

```
function balanceOf(address _owner) public view returns (uint256 balance)
```

## 2.6 transfer

将value数量的代币转移到地址to，并且必须触发Transfer事件。如果消息调用者的账户余额没有足够的代币，则该函数需要抛出异常。

注意，即使转移代币的数量为0也必须被视为正常调用并触发Transfer事件。

```
function transfer(address _to, uint256 _value) public returns (bool success)
```

## 2.7 transferForm

将value数量的代币从地址from转移到地址to，并且必须触发Transfer事件。transferFrom方法主要用于代币提现工作流，允许合约代表用户转移代币。例如这可以用于允许合约代表用户转移代币。除非from账户通过某种机制授权消息发送者，否则函数应该抛出异常。

注意，即使转移代币的数量为0也必须被视为正常调用并触发Transfer事件。

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

## 2.8 approve

允许spender多次从用户的账户中提款，最多为value数额。如果再次调用此函数，它会用新的value覆盖当前的允许值allowance。

为了防止该函数受到抢跑攻击(在后面会说明)，用户在为某个spender授权金额时，应当先将他的allowance设为0，然后再设置成你的目标值。

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

## 2.9 allowance

返回spender还允许从owner账户中提款的数量。

```
function allowance(address _owner, address _spender) public view returns (uint256
remaining)
```

# 3 ERC20事件(Events)

## 3.1 Transfer

必须在代币发生转移时触发，包括0 value转移。创建新代币的代币合约应该在创建代币时触发from地址为0x0的Transfer事件。

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

## 3.2 Approval

任何approve方法的成功调用都应该触发该事件。

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

# 4 approve可能存在的抢跑攻击

假定这样一个场景：A通过approval方法给B授权了N个代币的使用权，则此时B可以用transferForm方法将N个代币从A地址转出来使用，若过了一段时间A改变主意，决定变更B可以支配的代币额度为M，则A需要再次调用approval方法给B授权M个代币。

假设在这个过程中B一直在监听A的行为，当他发现A第二次调用approval方法改变额度时，他立刻以更高的手续费调用transferForm函数提前将N个代币转走，则当A的第二次approval方法生效后B还能继续提取M个代币。

在上述场景中，A只想让B使用N或M个代币，但是最终B通过交易抢跑可以提款N+M个代币。抢跑攻击成功的关键是每次approval方法生效时value都会直接覆盖掉allowance而不去判断allowance有无被使用。

# 5 资料来源


EIP20以太坊改进提案

approve接口可能存在的抢跑攻击