# 转账

要让合约接收以太币的转账，需要使用 **payable** 修饰符

转账有三种方式：**transfer**、**send**、**call**

1. **transfer**

**<address payable>.transfer(uint amount)**

将数量为amount的以太币转给 payable类型的地址

代码示例：

function test(address recipient) external payable{

uint256 amount = msg.value;

**payable(recipinet).transfer(amount);** //向recipinet转账，数量为amount

}

* 如果异常会转账失败，抛出异常（等价于 require(send)）
* 有gas限制，最大2300

**一般比较推荐transfer**，因为相比send它有报告提示，而且可以回滚操作。

1. **send**

**<address payable>.send(uint256 amount) returns(bool)**

将数量为amount的以太币发给 payable类型的地址,返回转账结果（true/false）

代码示例：

function test(address payable rec) payable{

uint256 amount = msg.value;

**payable(rec).send(amount);** //向rec转账

}

* 如果异常会转帐失败，只返回 false，**不会终止执行，不会回滚**
* 有gas限制，最大2300

send存在一个漏洞：

function sendTest() public{

require(!payedOut);

**winner.send(amount);** //不论成功、失败，都会执行下面的语句，可能导致出错

payedOut = true;

}

建议措施：

bool success = **winner.send(1 eth)**;

if(success){

…

}else{

…

}

1. **call**

**<address>.call(bytes memory) returns (bool, bytes memory)**

目标合约地址.call(**二进制编码**)，返回 call是否成功、目标函数的返回值

**<payable address>.call{value:amount1, gas:ammount2}(bytes memory) returns (bool, bytes memory)**

payable目标合约地址.call{value:发送数额, gas:gas数量}(**二进制编码**)，返回call是否成功、目标函数的返回值

* 如果异常会转帐失败，仅会返回false，不会终止执行（调用合约的方法并转账）
* 没有gas限制

**二进制编码** 利用结构化编码函数获得：

abi.encodeWithSignature(“目标函数”，函数参数1, 函数参数2, …)

代码示例：

//call要调用的目标合约

contract OtherContract{

uint256 private \_x = 0;

event Log(uint amount, uint gas); //收到eth事件，记录amount和gas

fallback() external payable{}

//返回合约ETH余额

function **getBalance()** view public **returns(uint)**{

return address(this).balance;

}

//设置 \_x 的函数，并可以往合约转ETH（payable）

function **setX(uint256 x)** external payable{

\_x = x;

if(msg.value > 0){

emit Log(msg.value, gasleft());

}

}

//读取x

function **getX()** external view **returns(uint x)**{

x = \_x;

}

}

利用call调用目标合约：

contract CallContract{

event Response(bool success, bytes data); //输出call返回的结果success和data

//调用setX(uint256)函数

function callSetX(address payable addr, uint256 x) public payable{

(bool suc, bytes memory data) =

addr.call{value:msg.value}(abi.encodeWithSignature(“setX(uint256)”,x));

emit Response(suc, data); //释放事件

}

//调用getX()函数

function callGetX(address addr) external returns(uint256){

(bool suc, bytes memory data) =

addr.call(abi.encodeWithSignature(“getX()”));

emit Response(suc, data); //释放事件

return abi.decode(data,(uint256));

}

//调用不存在的函数：如果输入的函数不存在，call仍能执行成功，并返回suc，

//但其实调用的是目标合约的fallback函数

function callNoExist(address addr) external{

(bool suc, bytes memory data) =

abi.call(abi.encodeWithSignature(“noMethod(uint256)”));

emit Response(suc, data);

}

}

call可能会发生重入攻击