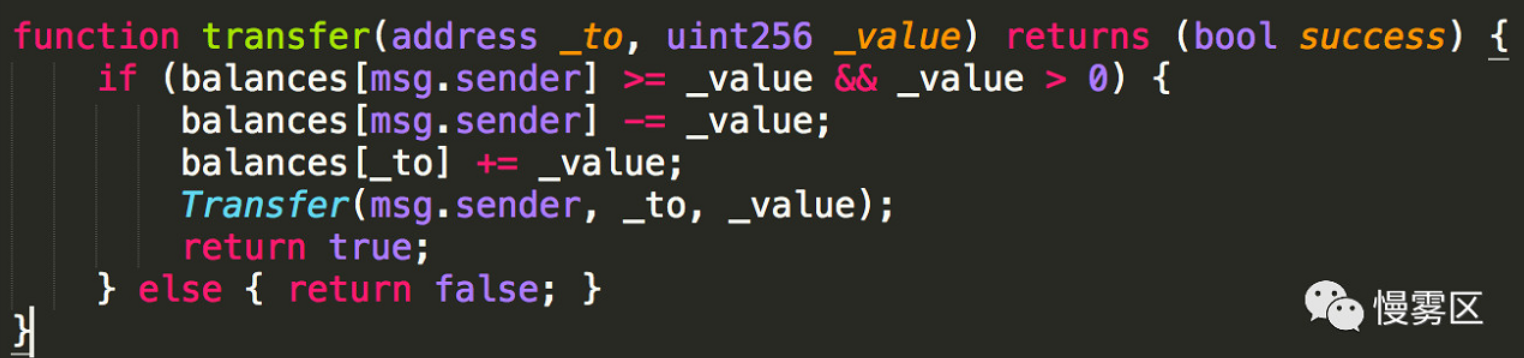
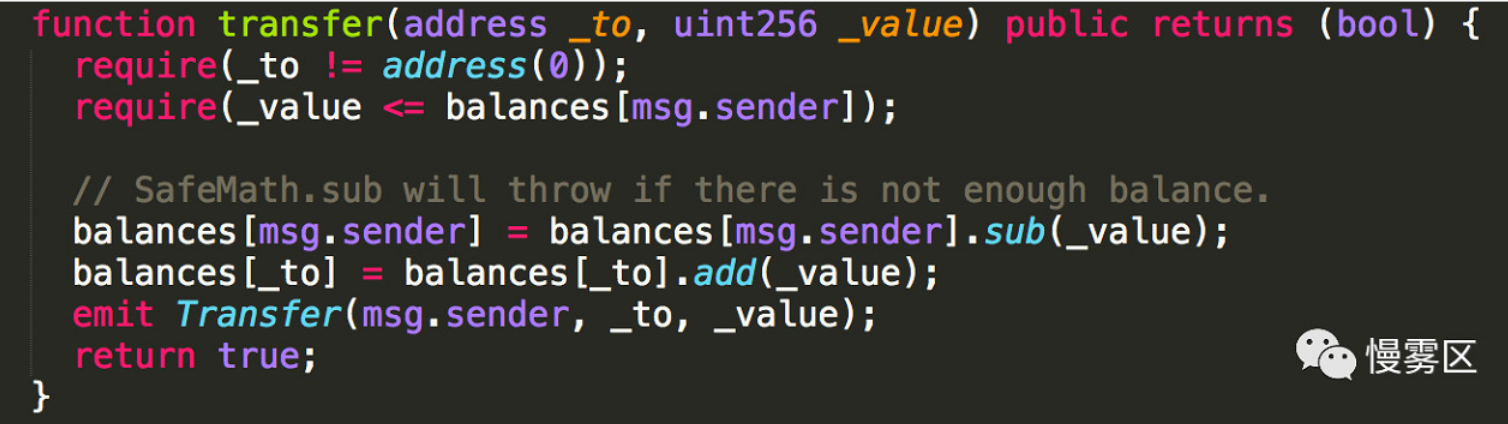
# 漏洞细节

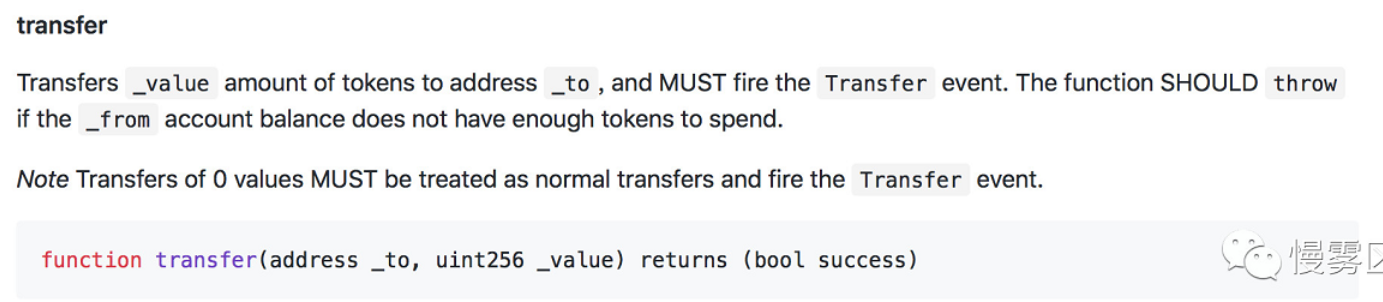
以太坊代币交易回执中 status 字段是 0x1(true) 还是 0x0(false)，取决于交易事务执行过程中是否抛出了异常（比如使用了 require/assert/revert/throw 等机制）。当用户调用代币合约的 transfer 函数进行转账时，如果 transfer 函数正常运行未抛出异常，该交易的 status 即是 0x1(true)。



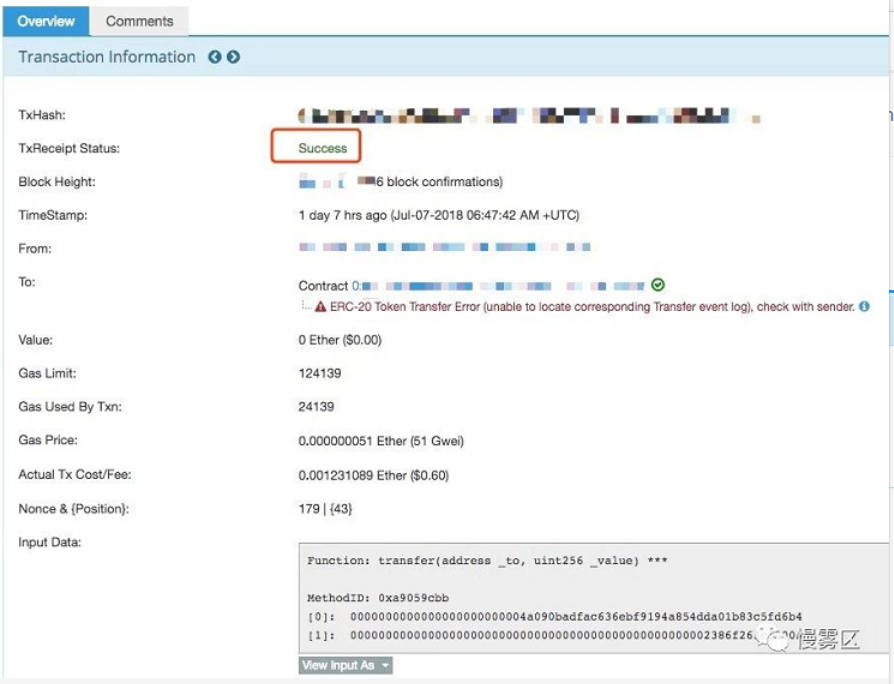
如图代码，某些代币合约的 transfer 函数对转账发起人(msg.sender)的余额检查用的是 if 判断方式，当 balances[msg.sender] < \_value 时进入 else 逻辑部分并 return false，最终没有抛出异常，我们认为仅 if/else 这种温和的判断方式在 transfer 这类敏感函数场景中是一种不严谨的编码方式。而大多数代币合约的 transfer 函数会采用 require/assert 方式，如图：



当不满足条件时会直接抛出异常，中断合约后续指令的执行，或者也可以使用 EIP 20 推荐的 if/else + revert/throw 函数组合机制来显现抛出异常，如图：



我们很难要求所有程序员都能写出最佳安全实践的代码，这种不严谨的编码方式是一种安全缺陷，这种安全缺陷可能会导致特殊场景下的安全问题。攻击者可以利用存在该缺陷的代币合约向中心化交易所、钱包等服务平台发起充值操作，如果交易所仅判断如 TxReceipt Status 是 success（即上文提的 status 为 0x1(true) 的情况） 就以为充币成功，就可能存在“假充值”漏洞。如图：



# 修复方案

除了判断交易事务 success 之外，还应二次判断充值钱包地址的 balance 是否准确的增加。其实这个二次判断可以通过 Event 事件日志来进行，很多中心化交易所、钱包等服务平台会通过 Event 事件日志来获取转账额度，以此判断转账的准确性。但这里就需要特别注意合约作恶情况，因为 Event 是可以任意编写的，不是强制默认不可篡改的选项：

        emit Transfer(from, to, value); // value 等参数可以任意定义

作为平台方，在对接新上线的代币合约之前，应该做好严格的安全审计，这种安全审计必须强制代币合约方执行最佳安全实践。

作为代币合约方，在编码上，应该严格执行最佳安全实践，并请第三方职业安全审计机构完成严谨完备的安全审计。

摘自：https://mp.weixin.qq.com/s/3cMbE6p\_4qCdVLa4FNA5-A