北京大学肖臻老师《区块链技术与应用》公开课笔记

> 美链，对应肖老师视频：click here
> 全系列笔记请见：click here
> **About Me:**点击进入我的Personal Page

2018年发生问题的美链(Beauty Chain)漏洞

关于美链(Beauty Chain)

- 美链(Beauty Chain)是一个部署在以太坊上的智能合约，有自己的代币BEC。
  - 没有自己的区块链，代币的发行、转账都是通过调用智能合约中的函数来完成的
  - 可以自己定义发行规则，每个账户有多少代币也是保存在智能合约的状态变量里
  - ERC 20是以太坊上发行代币的一个标准，规范了所有发行代币的合约应该实现的功能和遵循的接口
  - 美链中有一个叫batchTransfer的函数，它的功能是向多个接收者发送代币，然后把这些代币从调用者的帐户上扣除

> ERC指的是Ethereum Request for Comments.

函数实现

下为batchTransfer()的实现代码：

```
function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
  uint cnt = _receivers.length;
  uint256 amount = uint256(cnt) * _value;
  require(cnt > 0 && cnt <= 20);
  require(_value > 0 && balances[msg.sender] >= amount);

  balances[msg.sender] = balances[msg.sender].sub(amount);
  for (uint i = 0; i < cnt; i++) {
      balances[_receivers[i]] = balances[_receivers[i]].add(_value);
      Transfer(msg.sender, _receivers[i], _value);
  }
  return true;
}
}
```

保存代币接收者的数组，存放接收者的地址
转账金额(每个人)
计算需要转账的总金额(人数*value)
接收者数目最多20个
检查发起调用的账户确实有足够的代币
发起账户的代币数减去amount
通过循环，给每个接收者发送价值value的代币

攻击

有什么问题吗?

问题出现在计算总金额上面：`uint256 amount = uint256(cnt) * _value;`。

假如计算时，value的值很大，这个乘法结果可能会产生溢出(计算机底层数据表示的知识，无相关知识者就不要理解为什么了。)。溢出后算得的amount可能是一个很小的数。换言之，在调用者账户做减法时，减掉的是一个很小的数字，但转账时，转走的仍然是一笔很大的代币。从而导致系统中凭空产生大量代币。

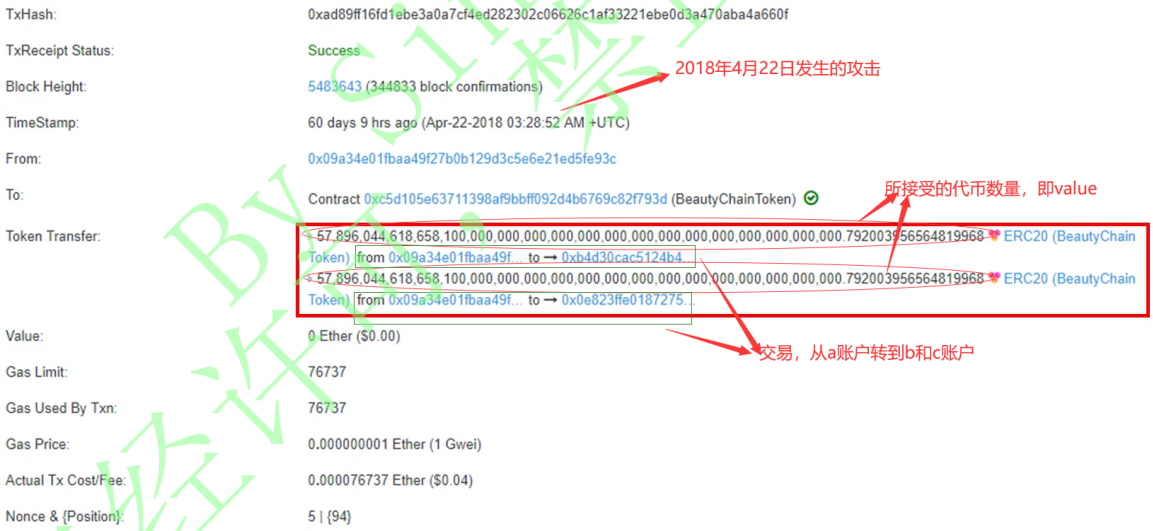
攻击细节

```
Function: batchTransfer(address[] _receivers, uint256 _value)

MethodID: 0x83f12fec
[0]:  0000000000000000000000000000000000000000000000000000000000000040
[1]:  8000000000000000000000000000000000000000000000000000000000000000
[2]:  0000000000000000000000000000000000000000000000000000000000000002
[3]:  000000000000000000000000b4d30cac5124b46c2df0cf3e3e1be05f42119033
[4]:  000000000000000000000000e823ffe018727585eaf5bc769fa80472f76c3d7
```

- 第0号参数是_receivers数组在参数列表中的位置，即从第64个byte开始，也就是第2号参数
  - 第2号参数先指明数组长度为2，然后第3号参数和第4号参数表明两个接受者的地址
- 第1号参数是给每个接受者转账的金额
- 通过这样的参数计算出来的amount恰好溢出为0!

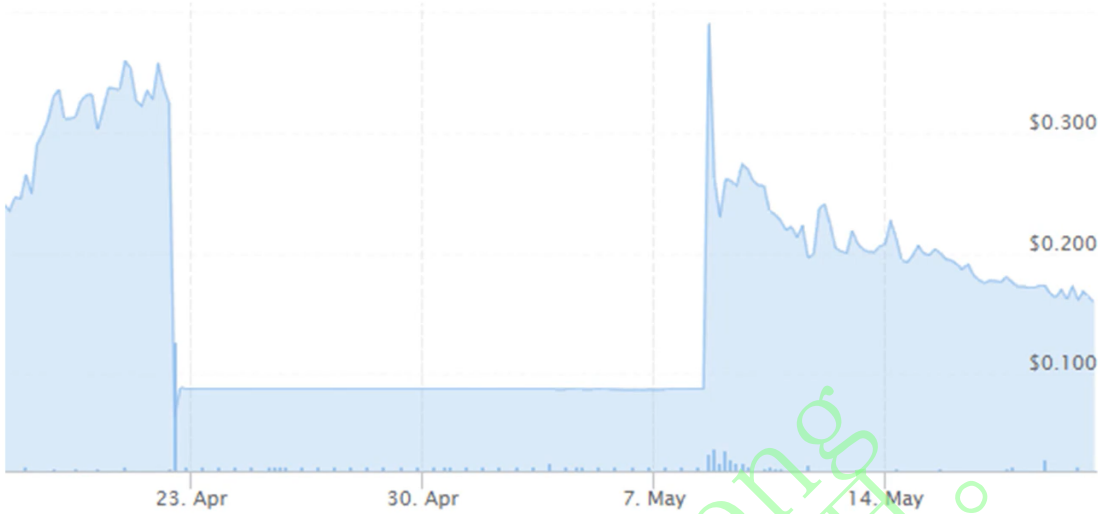
攻击细节



攻击结果

攻击发生后，对该代币价格产生了致命打击，4.23该代币价格产生断崖式跳水

攻击结果



- 攻击在2018年4月22日发生，攻击发生后币值暴跌

4.24日，OKEx交易所紧急公告暂停提币交易，防止黑客获利逃跑。两天后，进行交易回滚，从而及时弥补了损失。

关于OKEx暂停BEC交易和提现的公告【更新】

OKEx
22 天前 · 更新于

尊敬的OKEx用户，

2018年4月22日13时左右，BEC出现异常交易，应BeautyChain (BEC)项目方的要求，暂时关闭BEC/USDT、BEC/BTC、BEC/ETH的交易和BEC的提现，具体开放时间另行通知。OKEx会随时与项目方保持联络，待项目方有最新进展我们会第一时间公布，给您带来的不便深表歉意，感谢您对OKEx的支持和理解。

OKEx

2018-4-22

----------2018-4-24 17:00 更新----------

经讨论决定OKEx将BEC的BEC/USDT、BEC/BTC、BEC/ETH三个交易区的交易数据回滚至香港时间2018-04-22 13:18:00，在此时间以后所有参与BEC交易的账户会根据BEC的交易账单记录进行回滚，其他币种的交易记录不受影响，回滚以后所有参与BEC交易的账户均不会有任何资金损失。在2018年4月22日13:18:00以后没有参与BEC交易的账户不受此次数据回滚的影响。BEC的交易和提现开放时间将另行通知。

OKEx

2018-4-24

不过，该代币相对较为小众，引发影响远远没有The DAO事件大。

反思

在进行数学运算，需要考虑【溢出】的可能性。solidity中实际上提供了一个专用的库SafeMath，如果采用其提供的乘法计算，就可以检测到溢出。如下为SafeMath中乘法实现的代码：

```
library SafeMath {

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
    if (a == 0) {
      return 0;
    }
    c = a * b;
    assert(c / a == b);
    return c;
  }
```

用a*b得到c，再去判断用c除以a能否再得到b，如果产生溢出，assert()就会得到错误。

回头看一下bathcTransfer中的代码实现：

因此，有人怀疑这是自导自演，但从结果上看并非监守自盗。可能是由于开发者粗心大意，在乘法这里忘记使用安全函数了吧。

后记

我记得之前知乎上曾经有一个问题，为什么程序员要写bug，不写bug不可以吗？

如果你是非计算机专业，看到这里，看到了The DAO和美链中的实现漏洞，应该就可以得到问题的答案了。

人生不也是这样吗？我们不可能预先考虑到所有的场景，并做出最优的安排，编程也是如此，总有未知的漏洞存在。

我所亲身经历的一个事件

我曾经参与国产某型先进飞机研制过程信息化管理工作，当时遇到一个令人啼笑皆非的bug：

各个生产部门的生产单在线上产生("去纸化")，存在许多类单子，我们将其分为两类。一类单号可以由系统自动生成，另一类则由于某些原因需要用户手动输入某个编号。

对于前者，在编号栏我们便不允许用户填写，给其显示"自动生成"字样；对于后者，则仅为一个可以填写的文本框。获得编号后，要进行唯一性校验，如果该编号已经存在就不允许产生这个编号的单据。

为了实现函数复用，这两类单据的唯一性校验采用了同一个函数。只是在前者，后台校验时传入的编号为"自动生成"，后者则是生产部门用户手动输入的编号。

然而，某个现场工人在创建某个第二类单据时，由于业务不熟练，在应该自己手动填入编号的地方写入了”自动生成“四个字，希望系统帮其自动产生文件编号。

此时，唯一性校验判断通过，因为并不存在一个名为"自动生成"的文件存在，所以系统中就成功创建了一个名为"自动生成"的文件。该专业厂现场工人便认为自己的工作完成了。

然而，使用系统的其他人。在想要创建第一类文件时，唯一性校验发现存在一个”自动生成“文件，导致唯一性校验不通过，整个生产受阻，无法继续进行，对该型先进飞机生产过程造成了困扰(由于可能涉密，这里进行模糊化处理，希望理解)。

在我和我的同事收到问题后，历经数个小时排查，最终才定位到这一问题，发现原因后啼笑皆非。这种bug在发生之前，没有任何一个人可以提前料想到。