重点名词解释

钱包（Wallet)

钱包(Wallet)是一个管理私钥的工具，数字货币钱包形式多样，但它通常包含一个软件客户端，允许使用者通过钱包检查、存储、交易其持有的数字货币。它是进入区块链世界的基础设施和重要入口。

冷钱包(Cold Wallet)

冷钱包(Cold Wallet)是一种脱离网络连接的离线钱包，将数字货币进行离线储存的钱包。使用者在一台离线的钱包上面生成数字货币地址和私钥，再将其保存起来。冷钱包是在不需要任何网络的情况下进行数字货币的储存，因此黑客是很难进入钱包获得私钥的，但它也不是绝对安全的，随机数不安全也会导致这个冷钱包不安全，此外硬件损坏、丢失也有可能造成数字货币的损失，因此需要做好密钥的备份。

热钱包 （Hot Wallet)

热钱包(Hot Wallet)是一种需要网络连接的在线钱包，在使用上更加方便。但由于热钱包一般需要在线使用，个人的电子设备有可能因误点钓鱼网站被黑客盗取钱包文件、捕获钱包密码或是破解加密私钥，而部分中心化管理钱包也并非绝对安全。

因此在使用中心化交易所或钱包时，最好在不同平台设置不同密码，且开启二次认证，以确保自己的资产安全。

公钥(Public Key)

公钥(Public Key)是和私钥成对出现的，和私钥一起组成一个密钥对，保存在钱包中。公钥由私钥生成，但是无法通过公钥倒推得到私钥。公钥能够通过一系列算法运算得到钱包的地址，因此可以作为拥有这个钱包地址的凭证。

私钥(Private Key)

私钥(Private Key)是一串由随机算法生成的数据，它可以通过非对称加密算法算出公并通过私钥对区块链的资产拥有绝对控制权，因此，区块链资产安全的核心问题在于私钥的存储钥，公钥可以再算出币的地址。私钥是非常重要的，作为密码，除了地址的所有者之外，都被隐藏。区块链资产实际在区块链上，所有者实际只拥有私钥，，拥有者需做好安全保管。

和传统的用户名、密码形式相比，使用公钥和私钥交易最大的优点在于提高了数据传递的安全性和完整性，因为两者——对应的关系，用户基本不用担心数据在传递过程中被

黑客中途截取或修改的可能性。同时，也因为私钥加密必须由它生成的公钥解密，发送者也不用担心数据被他人伪造。

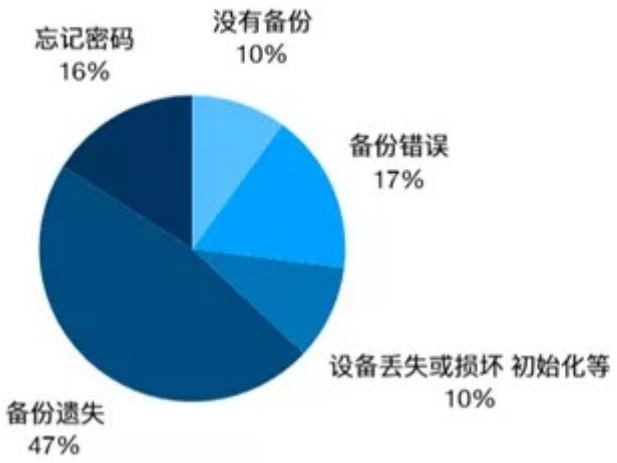
助记词(Mnemonic)

由于私钥是一长串毫无意义的字符，比较难以记忆，因此出现了助记词(Mnemonic)。助记词是利用固定算法，将私钥转换成十多个常见的英文单词。助记词和私钥是互通的，可以相互转换，它只是作为区块链数字钱包私钥的友好格式。所以在此强调：助记词即私钥！由于它的明文性，不建议它以电子方式保存，而是抄写在物理介质上保管好，它和Keystore作为双重备份互为补充。

Keystore

Keystore主要在以太坊钱包 App 中比较常见(比特币类似以太坊Keystore机制的是：BIP38)，是把私钥通过钱包密码再加密得来的，与助记词不同，一般可保存为文本或 JSON 格式存储。换句话说，Keystore需要用钱包密码解密后才等同于私钥。因此，Keystore需要配合钱包密码来使用，才能导入钱包。当黑客盗取Keystore后，在没有密码情况下, 有可能通过暴力破解Keystore密码解开Keystore，所以建议使用者在设置密码时稍微复杂些，比如带上特殊字符，至少 8 位以上，并安全存储。

由于区块链技术的加持使得区块链数字钱包安全系数高于其他的数字钱包，其中最为关键的就是两点：防盗和防丢。相比于盗币事件原因的多样化，造成丢币事件发生的原因主要有五个类型：没有备份、备份遗失、忘记密码、备份错误以及设备丢失或损坏。因此，我们在备份一个区块链数字钱包的时候，对私钥、助记词、Keystore一定要进行多重、多次备份，把丢币的风险扼杀在摇篮之中。



为大家提供一份来自imToken总结的钱包安全“十不原则”：

1. 不使用未备份的钱包；
2. 不使用邮件传输或存储私钥；

3.不使用微信收藏或云备份存储私钥；
4.不要截屏或拍照保存私钥；
5.不使用微信，QQ 传输私钥；
6.不要将私钥告诉身边的人；
7.不要将私钥发送到群里；
8.不使用第三方提供的未知来源钱包应用；
9.不使用他人提供的 Apple ID；
10. 不要将私钥导入未知的第三方网站。

公有链(Public Blockchain)

公有链(Public Blockchain)简称公链，是指全世界任何人都可随时进入读取、任何人都能发送交易且能获得有效确认的共识区块链。公链通常被认为是完全去中心化的，链上数据都是公开透明的，不可更改，任何人都可以通过交易或挖矿读取和写入数据。一般会通过代币机制(Token)来鼓励参与者竞争记账，来确保数据的安全性。

交易所(Exchange)

与买卖股票的证券交易所类似，区块链交易所即数字货币买卖交易的平台。数字货币交易所又分为中心化交易所和去中心化交易所。

去中心化交易所：交易行为直接发生在区块链上，数字货币会直接发回使用者的钱包，或是保存在区块链上的智能合约。这样直接在链上交易的好处在于交易所不会持有用户大量的数字货币，所有的数字货币会储存在用户的钱包或平台的智能合约上。去中心化交易通过技术手段在信任层面去中心化，也可以说是无需信任，每笔交易都通过区块链进行公开透明，不负责保管用户的资产和私钥等信息，用户资金的所有权完全在自己手上，具有非常好的个人数据安全和隐私性。目前市面上的去中心化交易所有WhaleEx、Bancor、dYdX等。

中心化交易所：目前热门的交易所大多都是采用中心化技术的交易所，使用者通常是到平台上注册，并经过一连串的身份认证程序(KYC)后，就可以开始在上面交易数字货币。用户在使用中心化交易所时，其货币交换不见得会发生在区块链上，取而代之的可能仅是修改交易所数据库内的资产数字，用户看到的只是账面上数字的变化，交易所只要在用户提款时准备充足的数字货币可供汇出即可。当前的主流交易大部分是在中心化交易所内完成的，目前市面上的中心化交易所有币安，火币，OKEx等。

节点(Node)

在传统互联网领域，企业所有的数据运行都集中在一个中心化的服务器中，那么这个服务器就是一个节点。由于区块链是去中心化的分布式数据库，是由千千万万个"小服务器"组成。区块链网络中的每一个节点，就相当于存储所有区块数据的每一台电脑或者服务器。所有新区块的生产，以及交易的验证与记帐，并将其广播给全网同步，都由节点来完成。节点分为"全节点"和"轻节点"，全节点就是拥有全网所有的交易数据的节点，那么轻节点就是只拥有和自己相关的交易数据节点。由于每一个全节点都保留着全网数据，这意味着，其中一个节点出现问题，整个区块链网络世界也依旧能够安全运行，这也是去中心化的魅力所在。

RPC (Remote ProcedureCall)

远程过程调用(Remote ProcedureCall，缩写为 RPC)是一个计算机通信协议。以太坊 RPC 接口是以太坊节点与其他系统交互的窗口，以太坊提供了各种 RPC 调用：HTTP、IPC、WebSocket 等等。在以太坊源码中，server.go是核心逻辑，负责 API 服务的注入，以及请求处理、返回。http.go实现 HTTP 的调用，websocket.go实现 WebSocket 的调用，ipc.go实现 IPC 的调用。以太坊节点默认在 8545 端口提供了 JSON RPC 接口，数据传输采用 JSON 格式，可以执行 Web3 库的各种命令，可以向前端（例如imToken、Mist 等钱包客户端）提供区块链上的信息。

以太坊黑色情人节漏洞(ETH Black Valentine's Day)

2018 年 3 月 20 日，慢雾安全团队观测到一起自动化盗币的攻击行为，攻击者利用以太坊节点Geth/Parity RPC API 鉴权缺陷，恶意调用eth_sendTransaction盗取代币，持续时间长达两年，单被盗的且还未转出的以太币价值就高达现价2千万美金(以当时 ETH 市值计算)，还有代币种类164 种，价值难以估计（很多代币还未上交易所正式发行）。

共识(Consensus)

共识算法主要是解决分布式系统中，多个节点之间对某个状态达成一致性结果的问题。分布式系统由多个服务节点共同完成对事务的处理，分布式系统中多个副本对外呈现的数据状态需要保持一致性。由于节点的不可靠性和节点间通讯的不稳定性，甚至节点作恶，伪造信息，使得节点之间出现数据状态不一致性的问题。通过共识算法，可以将多个不可靠的单独节点组建成一个可靠的分布式系统，实现数据状态的一致性，提高系统的可靠性。

区块链系统本身作为一个超大规模的分布式系统，但又与传统的分布式系统存在明显区别。由于它不依赖于任何一个中央权威，系统建立在去中心化的点对点网络基础之上，因此分散的节点需要就交易的有效与否达成一致，这就是共识算法发挥作用的地方，即确保所有节点都遵守协议规则并保证所有交易都以可靠的方式进行。由共识算法实现在分散的节点间对交易的处理顺序达成一致，这是共识算法在区块链系统中起到的最主要作用。
区块链系统中的共识算法还承担着区块链系统中激励模型和治理模型中的部分功能，为了解决在对等网络中(P2P),相互独立的节点如何达成一项决议问题的过程。简而言之，共识算法是在解决分布式系统中如何保持一致性的问题。

工作量证明PoW(Proof of Work)

PoW(Proof of Work)是历史上第一个成功的去中心化区块链共识算法。工作量证明是大多数人所熟悉的，被比特币、以太坊，莱特币等主流公链广泛使用。
工作量证明要求节点参与者执行计算密集型的任务，但是对于其他网络参与者来说易于验证。在比特币的例子中，矿工竞相向由整个网络维护的区块链账本中添加所收集到的交易，即区块。为了做到这一点，矿工必须第一个准确计算出“nonce”，这是一个添加在字符串末尾的数字，用来创建一个满足开头特定个数为零的哈希值。不过存在采矿的大量电力消耗和低交易吞吐量等缺点。

权益证明PoS(Proof of Stake)

PoS(Proof of Stake)——权益证明机制，一种主流的区块链共识算法，目的是为了让区块链里的分布式节点达成共识，它往往和工作量证明机制(Proof of Work)一起出现，两种都被认为是区块链共识算法里面的主流算法之一。作为一种算法，它通过持币人的同意来达成共识，目的是确定出新区块，这过程相对于PoW，不需要硬件和电力，且效率更高。
PoS共识中引入了 Stake 的概念，持币人将代币进行 Staking，要求所有的参与者抵押一部分他们所拥有的 Token 来验证交易，然后获得出块的机会，PoS共识中会通过选举算法，按照持币量比例以及 Token 抵押时长，或者是一些其他的方式，选出打包区块的矿工。矿工在指定高度完成打包交易，生成新区块，并广播区块，广播的区块经过PoS共识中另外一道"门槛"，验证人验证交易，通过验证后，区块得到确认。这样一轮PoS的共识过程就进行完成了。权益证明通过长期绑定验证者的利益和整个网络的利益来阻止不良行为。锁定代币后，如果验证者存在欺诈性交易，那么他们所抵押的 Token 也会被削减。

委托权益证明DPoS(Delegate Proof of Stake)

委托权益证明，其雏形诞生在 2013 年 12 月 8 日，Daniel Larimer 在bitsharetalk首次谈及用投票选择出块人的方式，代替PoS中可能出现的选举随机数被操纵的问题。在DPoS中，让每一个持币者都可以进行投票，由此产生一定数量的代表 ,或者理解为一定数量的节点或矿池，他们彼此之间的权利是完全相等的。持币者可以随时通过投票更换这些代表，以维系链上系统的“长久纯洁性”。在某种程度上，这很像是国家治理里面的代议制，或者说是人大代表制度。这种制度最大的好处就是解决了验证人过多导致的效率低下问题，当然，这种制度也有很明显的缺点，由于 “代表”制度，导致其一直饱受中心化诟病。

多签(Multi-sig)

多签(Multi-sig)指的是需要多个签名才能执行的操作(这些签名是不同私钥生成的)。这可用于提供更高的安全性，即使丢失单个私钥的话也不会让攻击者取得帐户的权限，多个值得信赖的各方必须同时批准更新，否则无效。
我们都知道，一般来说一个比特币地址对应一个私钥，动用这个地址中的资金需要私钥的持有者发起签名才行。而多重签名技术，简单来说，就是动用一笔资金时需要多个私钥签名才有效。多签的一个优势就是可以多方对一笔付款一起达成共识，才能支付成功。

软分叉(Soft-fork)

软分叉(Soft-fork)更多情况下是一种协议升级，当新共识规则发布后，没有升级的旧节点并不会意识到代码已经发生改变，而继续生产不合法的区块，就会产生临时性分叉，但新节点可以兼容旧节点，即新旧节点始终在同一条链上工作。

硬分叉(Hard-fork)

硬分叉(Hard-fork)是区块链发生永久性分歧，在新共识规则发布后，已经升级的节点无法验证未升级节点产生的区块，未升级节点也无法验证已经升级的节点产生的区块，即新旧节点互不兼容，通常硬分叉就会发生，原有正常的一条链被分成了两条链（已升级的一条链和未升级的一条链，且这两条链互不兼容）。

智能合约(Smart Contract)

智能合约并不是一个新的概念，早在 1995 年就由跨领域法律学者 Nick Szabo 提出：智能合约是一套以数字形式定义的承诺(Promises)，包括合约参与方可以在上面执行这些承诺的协议。在区块链领域中，智能合约本质可以说是一段运行

在区块链网络中的代码，它以计算机指令的方式实现了传统合约的自动化处理，完成用户所赋予的业务逻辑。

攻击方法介绍

恶意挖矿攻击(Cryptojacking)

恶意挖矿攻击(Cryptojacking)是一种恶意行为，指未经授权的情况下劫持用户设备挖掘加密货币。通常，攻击者会劫持受害者设备(个人PC 或服务器)的处理能力和带宽，由于加密货币挖掘需要大量算力，攻击者会尝试同时感染多个设备，这样他们能够收集到足够的算力来执行这种低风险和低成本的挖矿活动。

一般恶意挖矿软件会诱导用户在计算机上加载挖矿代码，或通过使用类似网络钓鱼的方法，如恶意链接、电子邮件或是在网站里植入挖矿脚本等方式，使系统无意中被隐藏的加密挖矿程序感染进而完成攻击行为。近年来，随着加密货币价格的上涨，更加复杂的恶意软件被开发出来，使恶意挖矿攻击事件层出不穷。

在此我们为大家提供几条建议防范恶意挖矿攻击：

1、注意设备性能和 CPU 利用率；
2、在 Web 浏览器上安装挖矿脚本隔离插件，例如MinerBlock，NoCoin和 Adblocker；
3、小心电子邮件附件和链接；
4、安装一个值得信赖的杀毒软件，让软件应用程序和操作系统保持最新状态。

无利益攻击(Nothing at Stake Attack)

无利益攻击(Nothing at Stake Attack)，是在PoS共识机制下一个有待解决的问题，其问题的本质可以简单概括为“作恶无成本，好处无限多”。

当PoS共识系统出现分叉(Fork)时，出块节点可以在“不受任何损失”的前提下，同时在两个分叉上出块；无论哪一个分叉后面被公认为主链，该节点都可以获得“所有收益”且不会有任何成本损失。这就很容易给某些节点一种动力去产生新的分叉，支持或发起不合法交易，其他逐利的出块节点会同时在多条链(窗口)上排队出块支持新的分叉。随着时间的推移，分叉越来越多，非法交易，作恶猖狂。区块链将不再是唯一链，所有出块节点没有办法达成共识。

为了预防这样的情况发生，许多类PoS共识机制对此的解决方法是引入惩罚机制，对作恶的节点进行经济惩罚(Slashing)，以建立更加稳定的网络。DPoS实际上也是无利益攻击的解决方案之一，由上文我们可知DPoS这个机制由持币人选出出块节点来运营网络，出块节点会将一部分奖励分给投票者。

双花攻击(Double Spend Attack)

双花攻击(Double Spend Attack)即一笔钱花了两次，双重支付，利用货币的数字特性两次或多次使用“同一笔钱”完成支付。双花不会产生新的 Token，但能把自己花出去的钱重新拿回来。简单说就是，攻击者将一笔 Token 转到另外一个地址，通常是转到交易所进行套现，然后再利用一些攻击手法对转账交易进行回滚。目前有常见的几种手法能够引发双花攻击：

Race Attack

这种攻击主要通过控制矿工费来实现双花。攻击者同时向网络中发送两笔交易，一笔交易发给自己(为了提高攻击成功的概率，他给这笔交易增加了足够的矿工费)，一笔交易发给商家。由于发送给自己的交易中含有较高的手续费，会被矿工优先打包进区块的概率比较高。这时候这笔交易就会先于发给商家的那笔交易，那么发给商家的交易就会被回滚。对于攻击者来说，通过控制矿工费，就实现了同一笔 Token 的“双花”。

Finney Attack

攻击者主要通过控制区块的广播时间来实现双花，攻击对象针对的是接受 0 确认的商家。假设攻击者挖到区块，该区块中包含着一个交易，即 A 向 B 转了一定数量的 Token，其中 A 和 B 都是攻击者的地址。但是攻击者并不广播这个区块，而是立即找到一个愿意接受 0 确认交易的商家向他购买一个物品，向商家发一笔交易，用 A 向商家的地址 C 支付，发给商家的交易广播出去后，攻击者再把自己之前挖到的区块广播出去，由于发给自己的交易先于发给商家的交易，对于攻击者来说，通过控制区块的广播时间，就实现了同一笔 Token 的“双花”。

Vector76attack

Vector76 Attack 又称“一次确认攻击”，也就是交易确认一次后仍然可以回滚，是 Finney Attack 和 Race Attack 的组合。

攻击者创建两个节点，节点 A 连接到商家节点，节点 B 连接到区块链网络中的其他节点。接着，攻击者用同一笔 Token 发起两笔交易，一笔交易发送给商家地址，我们称为交易 1；一笔交易发送给自己的钱包地址，我们称为交易 2。与上面说的 Race Attack 一样，攻击者对交易 2 添加了较高的矿工费从而提高了矿工的打包概率，此时，攻击者并没有把这两笔交易广播到网络中去。

接着，攻击者开始在交易1所在的分支上进行挖矿，这条分支我们命名为分支1。攻击者挖到区块后，并没有广播出去，而是同时做了两件事：在节点 A 上发送交易 1，在节点 B 上发送交易 2。

由于节点 A 只连接了商家节点，所以当商家节点想把交易 1 传给其它对等节点时，连接了更多节点的节点 B，已经把交易 2 广播给了网络中的大部分节点。于是，从概率上来讲，交易 2 就更有可能被网络认定为是有效的，交易 1 被认定为无效。

交易 2 被认为有效后，攻击者立即把自己之前在分支1上挖到的区块，广播到网络中。这时候，这个接受一次确认就支付的商家，会确认交易成功，然后攻击者就可以立即变现并转移资产。

同时，由于分支2连接的更多节点，所以矿工在这个分支上挖出了另一个区块，也就是分支 2 的链长大于分支 1 的链长。于是，分支 1 上的交易就会回滚，商家之前支付给攻击者的交易信息就会被清除，但是攻击者早已经取款，实现了双花。

51% attack

攻击者占有超过全网 50% 的算力，在攻击者控制算力的这段时间，他可以创造一条高度大于原来链的新链。那么旧链中的交易会被回滚，攻击者可以使用同一笔 Token 发送一笔新的交易到新链上。

异形攻击(Alien Attack)

异形攻击(Alien Attack)实际上是一个所有公链都可能面临的问题，又称地址池污染，是指诱使同类链的节点互相侵入和污染的一种攻击手法，漏洞的主要原因是同类链系统在通信协议上没有对不同链的节点做识别。

这种攻击在一些参考以太坊通信协议实现的公链上得到了复现：以太坊同类链，由于使用了兼容的握手协议，无法区分节点是否属于同个链，利用这一点，攻击者先对以太坊节点地址进行收集并进行恶意握手操作，通过跟节点握手达成污染地址池的目的，使得不同链的节点互相握手并把各自地址池里已知的节点推送给了对方，导致更多的节点互相污染，最终扩散致整个网络。遭受异形攻击的节点通常会通信性能下降，最终造成节点阻塞、主网异常等现象。相关公链需要注意持续保持主网健康状态监测，以免出现影响主网稳定的攻击事件出现。

钓鱼攻击(Phishing)

所谓“钓鱼攻击(Phishing)”，指的是攻击者伪装成可以信任的人或机构，通过电子邮件、通讯软件、社交媒体等方式，以获取收件人的用户名、密码、私钥等私密信息。随着技术的发展，网络钓鱼攻击不仅可以托管各种恶意软件和勒索软件攻击，而且更糟糕的是这些攻击正在呈现不断上升的趋势。

建议用户保持警惕，通过即时通讯 App、短信或电子邮件获取到的每条信息都需要谨慎对待，不要在通过点击链接到达的网站上输入凭据或私钥，在交易时尽可能的使用硬件钱包和双因素认证(2FA)，生态中的项目方在攻击者没有确切告知漏洞细节之前，不要给攻击者转账，若项目方无法准确判断和独自处理，可以联系安全公司协助处理。

木马攻击(Trojan Horse Attack)

木马攻击(Trojan Horse Attack)是指攻击者通过隐藏在正常程序中的一段具有特殊功能的恶意代码，如具备破坏和删除文件、发送密码、记录键盘和 DDoS 攻击等特殊功能的后

门程序，将控制程序寄生于被控制的计算机系统中，里应外合，对被感染木马病毒的计算机实施操作。可用来窃取用户个人信息，甚至是远程控制对方的计算机而加壳制作，然后通过各种手段传播或者骗取目标用户执行该程序，以达到盗取密码等各种数据资料等目的。

在区块链领域，诸如勒索木马、恶意挖矿木马一直是行业内令人头疼的安全顽疾，据币世界报道，随着比特币的飙升，推动整个数字加密货币价格回升，与币市密切相关的挖矿木马开始新一轮活跃，仅 2019 年上半年挖矿木马日均新增 6 万个样本，通过分析发现某些新的挖矿木马家族出现了快速、持续更新版本的现象，其功能设计越来越复杂，在隐藏手法、攻击手法方面不断创新，与杀软厂商的技术对抗正在不断增强。

供应链攻击(Supply Chain Attack)

供应链攻击(Supply Chain Attack)是一种非常可怕的攻击方式，防御上很难做到完美规避，由于现在的软件工程，各种包/模块的依赖十分频繁、常见，而开发者们很难做到一一检查，默认都过于信任市面上流通的包管理器，这就导致了供应链攻击几乎已经成为必选攻击之一。把这种攻击称成为供应链攻击，是为了形象说明这种攻击是一种依赖关系，一个链条，任意环节被感染都会导致链条之后的所有环节出问题。

供应链攻击防不胜防且不计代价，建议所有数字加密货币相关项目(如交易所、钱包、DApp等)都应该强制至少一名核心技术完整审查一遍所有第三方模块，看看是否存在可疑代码，也可以通过抓包查看是否存在可疑请求。

交易回滚攻击(Roll Back Attack)

交易回滚攻击(Roll Back Attack)，故名思义，指的是能对交易的状态进行回滚。回滚具体是什么意思呢？回滚具体指的是将已经发生的状态恢复成它未发生时候的样子。那么，交易回滚的意思就是将已经发生的交易变成未发生的状态。即攻击者本来已经发生了支付动作，但是通过某些手段，让转账流程发生错误，从而回滚整个交易流程，达到交易回滚的目的，这种攻击手法多发于区块链上的的智能合约游戏当中，当用户的下注动作和合约的开奖动作在一个交易内的时候，即内联交易。攻击者就可以通过交易发生时检测智能合约的某些状态，获知开奖信息，根据开奖信息选择是否对下注交易进行回滚。

建议开发者们不要将用户的下注与开奖放在同一个交易内，防止攻击者通过检测智能合约中的开奖状态实现交易回滚攻击。

交易排挤攻击(Transaction Congestion Attack)

交易排挤攻击(Transaction Congestion Attack)是针对 EOS 上的使用 defer 进行开奖的游戏合约的一种攻击手法，攻击者可以通过某些手段，在游戏合约的 defer 开奖交易前发送大量的 defer 交易，恶意侵占区块内的 CPU 资源，使得智能合约内本应在指定区块内执行的 defer 开奖交易因资源不足无法执行，只能去到下一个区块才执行。由于很多 EOS

上的游戏智能合约使用区块信息作为智能合约本身的随机数，同一个 defer 开奖交易在不同区块内的执行结果是不一样的。通过这样的方式，攻击者在获知无法中奖的时候，就通过发送大量的 defer 交易，强行让智能合约重新开奖，从而达到攻击目的。

建议智能合约开发者对在不同区块内执行结果不同的关键的操作不要采用 defer 交易的方式，降低合约被攻击的风险。

随机数攻击(Random Number Attack)

随机数攻击(Random Number Attack)，就是针对智能合约的随机数生成算法进行攻击，预测智能合约的随机数。目前区块链上很多游戏都是采用的链上信息（如区块时间，未来区块哈希等）作为游戏合约的随机数源，也称随机数种子。使用这种随机数种子生成的随机数被称为伪随机数。伪随机数不是真的随机数，存在被预测的可能。当使用可被预测的随机数种子生成随机数的时候，一旦随机数生成的算法被攻击者猜测到或通过逆向等其他方式拿到，攻击者就可以根据随机数的生成算法预测游戏即将出现的随机数，实现随机数预测，达到攻击目的。

建议智能合约开发者不要使用不安全的随机数种子生成随机数，降低合约被攻击的风险。

hard_fail状态攻击hard_fail Attack

hard_fail是什么呢？简单来说就是出现错误但是没有使用错误处理器(error handler)处理错误，比方说使用onerror捕获处理，如果说没有onerror捕获，就会hard_fail。EOS 上的交易状态记录分为 executed,soft_fail, hard_fail, delayed 和 expired 这 5 种状态，通常在链上大部分人观察到的交易，都是 executed 的，或者 delayed 的，而没有失败的交易，这就导致大部分开发者误以为 EOS 链上没有失败的交易记录，从而忽略了对交易状态的检查。攻击者利用这个细节，针对链上游戏或交易所进行攻击，构造执行状态为hard_fail的交易，欺骗链上游戏或交易所进行假充值攻击，从而获利。 在此提醒交易所和 EOS DApp游戏开发者在处理转账交易的时候需要严格校验交易状态，确保交易执行状态为executed。

重放攻击(Replay Attack)

重放攻击(Replay Attack)，是针对区块链上的交易信息进行重放，一般来说，区块链为了保证不可篡改和防止双花攻击的发生，会对交易进行各种验证，包括交易的时间戳，nonce，交易 id 等，但是随着各种去中心化交易所的兴起，在智能合约中验证用户交易的场景越来越多。这种场景一般是需要用户对某一条消息进行签名后上传给智能合约，然后在合约内部进行验签。但由于用户的签名信息是会上链的，也就是说每个人都能拿到用户的签名信息，当在合约中校验用户签名的时候，如果被签名的消息不存在随着交易次数变化的变量，如时间戳，nonce 等，攻击者就可以拿着用户的签名，伪造用户发起交易，从而获利。

这是一种最早出现于DApp生态初期的攻击形态，由于开发者设计的开奖随机算法存在严重缺陷，使得攻击者可利用合约漏洞重复开奖，属于开发者较为容易忽略的错误。因此，开发者们在链上进行验签操作的时候，需要对被签名消息加上各种可变因子，防止攻击者对链上签名进行重放，造成资产损失。

重入攻击(Reentrancy Attack)

重入攻击(Reentrancy Attack)首次出现于以太坊，对应的真实攻击为 The DAO 攻击，此次攻击还导致了原来的以太坊分叉成以太经典(ETC)和现在的以太坊(ETH)。由于项目方采用的转账模型为先给用户发送转账然后才对用户的余额状态进行修改，导致恶意用户可以构造恶意合约，在接受转账的同时再次调用项目方的转账函数。利用这样的方法，导致用户的余额状态一直没有被改变，却能一直提取项目方资金，最终导致项目方资金被耗光。 提醒智能合约开发者在进行智能合约开发时，在处理转账等关键操作的时候，如果智能合约中存储了用户的资金状态，要先对资金状态进行修改，然后再进行实际的资金转账，避免重入攻击。

假充值攻击 False Top-up

假充值攻击(False Top-up)，分为针对智能合约的假充值攻击和对交易所的假充值攻击。在假充值攻击中，无论是智能合约还是交易所本身，都没有收到真实的 Token，但是用户又确实得到了真实的充值记录，在这种情况下，用户就可以在没有真正充值的情况下从智能合约或交易所中用假资产或不存在的资产窃取真实资产。

智能合约假充值攻击

针对智能合约的假充值主要是假币的假充值，这种攻击手法多发于 EOS 和波场上，由于 EOS 上代币都是采用合约的方式进行发行的，EOS 链的系统代币同样也是使用这种方式发行，同时，任何人也可以发行名为 EOS 的代币。只是发行的合约帐号不一样，系统代币的发行合约为"eosio.token"，而其他人发行的代币来源于其他合约帐号。当合约内没有校验 EOS 代币的来源合约的时候，攻击者就能通过充值攻击者自己发布的 EOS 代币，对合约进行假充值攻击。而波场上的假充值攻击主要是 TRC10 代币的假充值攻击，由于每一个 TRC10 都有一个特定的tokenid进行识别，当合约内没有对tokenid进行校验的时候，任何人都可以以 1024 个 TRX 发行一个 TRC10 代币对合约进行假充值。

交易所假充值攻击

针对交易所的假充值攻击分为假币攻击和交易状态失败的假充值攻击。以 EOS 和以太坊为例。针对 EOS 可以使用名为 EOS 的假币的方式对交易所进行假充值攻击，如果交易所没有严格校验 EOS 的来源合约为"eosio.token"，攻击就会发生。同时，区别于 EOS，由于以太坊上会保留交易失败的记录，针对 ERC20 Token，如果交易所没有校验交易的状态，就能通过失败的交易对交易所进行 ERC20 假充值。除此之外，hard_fail状态攻击也是属于假充值攻击的一种。

建议交易所和智能合约开发者在处理转账的时候要充分校验交易的状态，如果是 EOS 或波场上的交易，在处理充值时还要同时校验来源合约是否是"eosio.token" 或tokenid是否为指定的tokenid。

短地址攻击 Short Address Attack

短地址攻击(Short Address Attack)是针对以太坊上 ERC20 智能合约的一种攻击形式，利用的是 EVM 中的对于输入字节码的自动补全机制进行攻击。

一般而言，针对 ERC20 合约中的 transfer 函数的调用，输入的字节码位数都是 136 字节的。当调用 ERC20 中的 transfer 函数进行 ERC20 Token 转账时，如果攻击者提供的地址后有一个或多个 0，那么攻击者就可以把地址后的零省去，提供一个缺位的地址。当对这个地址转账的时候，比方说转账 100 的 A Token，然后输入的地址是攻击者提供的缺位地址，这时候，经过编码输入的数据是 134 字节，比正常的数据少了 2 字节，在这种情况下，EVM 就会对缺失的字节位在编码数据的末尾进行补 0 凑成 136 字节，这样本来地址段缺失的 0 被数据段的 0 补齐了，而由于给地址段补 0，数据段会少 0，而数据段缺失的 0 由 EVM 自动补齐，这就像数据段向地址段移动补齐地址段缺失字节位，然后数据段缺失的字节位由 EVM 用 0 补齐。这种情况下，转账金额就会由 100 变成 100 * 16 的 n 次方，n 是地址缺失的 0 的个数。通过这种方式，攻击者就能对交易所或钱包进行攻击，盗窃交易所和钱包的资产。

建议交易所和钱包在处理转账的时候，要对转账地址进行严格的校验，防止短地址攻击的发生。

假币攻击 Fake Token Attack

假币攻击(Fake Token Attack)，是针对那些在创建官方 Token 时采用通用创建模版创建出来的代币，每个 Token 的识别仅根据特定的标记进行识别，如 EOS 官方 Token 的识别标记是"eosio.token"合约，波场的 TRC10 的识别标记是tokenid，以太坊的 ERC20 是用合约地址作为识别标记。那么这样就会出现一个问题，如果收款方在对这些 Token 进行收款的时候没有严格校验这些 Token 特有的标记，攻击就会发生，以 EOS 为例子，由于 EOS 官方 Token 采用的是合约来发行一个名为 EOS 的 Token，标记 EOS 本身的标识是"eosio.token" 这个发行帐号，如果在接受转账的时候没有校验这个标识，攻击者就能用其他的帐号同样发行一个名为 EOS 的 Token，对交易所或钱包进行假币充值，换取真的代币。

建议交易所和钱包在处理转账的时候，切记要严格检验各种代币各种标识，防止假币攻击。

整型溢出攻击 Integer Overflow Attack

数据的存储是区块链上重要的一环。但是每个数据类型本身是存在边界的，例如以太坊中 uint8 类型的变量就只能存储 0 ~ 255 大小的数据，超过了就存不下了。那么如果要放

一个超过数据类型大小的数字会怎样呢？例如把 256 存进 uint8 的数据类型中，数据显示出来会变成 1，而不是其他数值，也不会报错，因为 uint8 本身能存一个 8 位二进制数字，最大值为 11111111，如果这个时候加 1，这个二进制数就变成了 100000001，而因为数据边界的关系，只能拿到后 8 位，也就是 00000001，那么数字的大小就变成 1 了，这种情况我们称为上溢。有上就有下，下溢的意思就是一个值为 0 的 uint8 数据，如果这个时候对它进行减 1 操作，结果会变成该数据类型所能存储的最大值加 1 减去被减数，在这个例子中是 255，也就是该数据类型所能存储的最大值。那么如果上述两种情况发生在智能合约当中的话，恶意用户通过下溢的操作，操纵自己的帐号向其他帐号发送超过自己余额数量的代币，如果合约内没有对余额进行检查，恶意用户的余额就会下溢出变成一个超大的值，这个时候攻击者如果大量抛售这些代币，就能瞬间破坏整个代币的价值系统。

建议所有的智能合约开发者在智能合约中对数据进行操作的时候，要严格校验数据边界，防止整形溢出攻击的发生。

条件竞争攻击 Race Condition

条件竞争(Race Condition)攻击的方式很多样，但是核心的本质无非是对某个条件的状态修改的竞争。条件竞争的例子：著名的 Edgeware 锁仓合约的拒绝服务漏洞，这个漏洞问题的本质在于对新建的锁仓合约的余额的这个条件进行竞争。攻击者可以监控所有链上的锁仓请求，提前计算出锁仓合约的地址，然后向合约地址转账，造成锁仓失败。在官方没有修复之前，要防止这种攻击，只能使用比攻击者更高的手续费让自己的锁仓交易先行打包，从而与攻击者形成竞争避免攻击。最后，官方修复方案为不对锁仓合约的余额进行强制性的相等检查，而是采用大于等于的形式，避免了攻击的发生。

建议智能合约的开发者在智能合约中对某些状态进行修改的时候，要根据实际情况充分考虑条件竞争的风险，防止遭受条件竞争攻击。

越权访问攻击 Exceed Authority Access Attack

和传统安全的定义一样，越权指的是访问或执行超出当前账户权限的操作，如本来有些操作只能是合约管理员执行的，但是由于限制做得不严谨，导致关键操作也能被合约管理员以外的人执行，导致不可预测的风险，这种攻击在以太坊和 EOS 上都曾出现过多次。

以 EOS 上著名的BetDice游戏为例，由于在游戏合约内的路由(EOS 内可自定义的事件转发器)中没有对来源账号进行严格的校验，导致普通用户能通过 push action 的方式访问到合约中的关键操作 transfer 函数，直接绕过转账流程进行下注，从而发生了越权攻击，事后虽然BetDice官方紧急修复了代码，并严格限制了来源账号，但这个漏洞已经让攻击者几乎无成本薅走BetDice奖池内将近 5 万 EOS。又如在以太坊使用 solidity 版本为 0.4.x 进行合约开发的时候，很多合约开发者在对关键函数编写的时候不仅没有加上权限

校验，也没有指定函数可见性，在这种情况下，函数的默认可见性为 public，恶意用户可以通过这些没有进行限制的关键函数对合约进行攻击。

建议智能合约开发者们在进行合约开发的时候要注意对关键函数进行权限校验，防止关键函数被非法调用造成合约被攻击。

交易顺序依赖攻击 Transaction-Ordering Attack

在区块链的世界当中，一笔交易内可能含有多个不同的交易，而这些交易执行的顺序会影响最终的交易的执行结果，由于在挖矿机制的区块链中，交易未被打包前都处于一种待打包的 pending 状态，如果能事先知道交易里面执行了哪些其他交易，恶意用户就能通过增加矿工费的形式，发起一笔交易，让交易中的其中一笔交易先行打包，扰乱交易顺序，造成非预期内的执行结果，达成攻击。以以太坊为例，假如存在一个 Token 交易平台，这个平台上的手续费是通过调控合约中的参数实现的，假如某天平台项目方通过一笔交易请求调高交易手续费用，这笔交易被打包后的所有买卖Token的交易手续费都要提升，正确的逻辑应该是从这笔交易开始往后所有的 Token 买卖交易的手续费都要提升，但是由于交易从发出到被打包存在一定的延时，请求修改交易手续费的交易不是立即生效的，那么这时恶意用户就可以以更高的手续费让自己的交易先行打包，避免支付更高的手续费。

建议智能合约开发者在进行合约开发的时候要注意交易顺序对交易结果产生的影响，避免合约因交易顺序的不同遭受攻击。

女巫攻击 Sybil Attack

传闻中女巫是一个会魔法的人，一个人可以幻化出多个自己，令受害人以为有多人，但其实只有一个人。在区块链世界中，女巫攻击(Sybil Attack)是针对服务器节点的攻击。攻击发生时候，通过某种方式，某个恶意节点可以伪装成多个节点，对被攻击节点发出链接请求，达到节点的最大链接请求，导致节点没办法接受其他节点的请求，造成节点拒绝服务攻击。

建议在搭建全节点的情况下，服务器需要在系统层面上对网络连接情况进行监控，一旦发现某个IP连接异常就调用脚本配置 iptables 规则屏蔽异常的 IP，同时链开发者在进行公链开发时应该在 P2P 模块中对单 IP 节点连接数量添加控制。

假错误通知攻击 Fake Onerror Notification Attack

EOS 上存在各种各样的通知，只要在 action 中添加require_recipient命令，就能对指定的帐号通知该 action，在 EOS 上某些智能合约中，为了用户体验或其他原因，一般会对onerror通知进行某些处理。如果这个时候没有对onerror通知的来源合约是否是eosio进行检验的话，就能使用和假转账通知同样的手法对合约进行攻击，触发合约中对onerror的处理，从而导致被攻击合约资产遭受损失。

建议智能合约开发者在进行智能合约开发的时候需要对onerror的来源合约进行校验，确保合约帐号为eosio帐号，防止假错误通知攻击。

粉尘攻击 Dusting Attack

粉尘攻击(Dusting Attack)最早发生于比特币网络当中，所谓粉尘，指的是交易中的交易金额相对于正常交易而言十分地小，可以视作微不足道的粉尘。通常这些粉尘在余额中不会被注意到，许多持币者也很容易忽略这些余额。但是由于比特币或基于比特币模型的区块链系统的账本模型是采用 UTXO 模型作为账户资金系统，即用户的每一笔交易金额，都是通过消费之前未消费的资金来产生新的资金。别有用意的用户，就能通过这种机制，给大量的账户发送这些粉尘金额，令交易粉尘化，然后再通过追踪这些粉尘交易，关联出该地址的其他关联地址，通过对这些关联地址进行行为分析，就可以分析一个地址背后的公司或个人，破坏比特币本身的匿名性。除此之外，由于比特币网络区块容量大小的限制，大量的粉尘交易会造成区块的拥堵，从而使得交易手续费提升，进而产生大量待打包交易，降低系统本身的运行效率。

对于如何避免粉尘攻击，可以在构造交易的过程中，根据交易的类型，计算出交易的最低金额，同时对每个输出进行判断，如果低于该金额，则不能继续构造该笔交易。特别的，如果这个输出刚好发生在找零上，且金额对于你来说不太大，则可以通过舍弃该部分的粉尘输出，以充作交易手续费来避免构造出粉尘交易。其次，为了保护隐私性，建议可以在构造交易时把那些金额极小的 UTXO 舍弃掉，使用大额的 UTXO 组成交易。

C2 攻击 C2 Attack

C2 全称 Command and Control，翻译过来就是命令执行与控制，在传统的网络攻击中，在通过各种漏洞进入到目标服务器后，受限于空间，通常通过网络拉取二段 exploit 进行驻留，实现后渗透流程。所以，C2 架构也就可以理解为，恶意软件通过什么样的方式获取资源和命令，以及通过什么样的方式将数据回传给攻击者。在传统的攻击手法中，攻击者一般通过远程服务器拉取命令到本地执行，但是这种方式也有很明显的缺点，就是一旦远程服务器被发现，后续渗透活动就无法正常进行。但是区块链网络提供了一个天然且不可篡改的大型数据库，攻击者通过把攻击荷载(payload)写进交易中，并通过发送交易把该命令永久的刻在区块链数据库中。通过这种方法，即使攻击命令被发现，也无法篡改链上数据，无需担心服务器被发现然后下线的风险。

新技术不断发展，旧有的攻击手法也在随着新技术的变换而不断迭代更新。在区块链的世界中只有在各方面都做好防范，才能避免来自各方面的安全攻击。

洗币 Money Launderin

洗币和洗钱是一样的，只是对象不同，洗钱指的是将一笔非法得到的金钱通过某些操作后变成正当、合法的收入。而洗币也是一样，指的是将非法获取的代币，如通过黑客攻击、携带用户资产跑路或通过诈骗等手段获取的代币，通过某些手段，将其来源变成正

当、合法的来源。如通过交易所进行洗币、智能合约中洗币或通过某些搅拌器进行中转、通过匿名币种如门罗币，Zcash等，令非法所得的资金无法被追踪，最后成功逃过监管达到洗币的目的，然后通过把代币转换成法币离场，完成洗币的流程。

建议各交易所应加强 KYC 策略，增强风控等级，及时监控交易所大资金进出，防范恶意用户通过交易所进行洗币，除此之外，可以通过与第三方安全机构进行合作，及时拦截非法资产，阻断洗钱的可能。

勒索 Ransom

勒索是传统行业中常见的攻击行为，攻击者通过向受害者主机发送勒索病毒对主机文件进行加密来向受害者进行资金勒索。随着区块链技术的发展，近年来，勒索开始呈现新的方式，如使用比特币作为勒索的资金支付手段或使用匿名性更高的门罗币作为资金支付手段。如著名的GandCrab病毒就是比特币勒索病毒，受害者需要向攻击者支付一定量的比特币换取解密私钥。通过这种勒索手段，GandCrab勒索病毒一年就勒索了超过 20 亿美金。值得一提的是，就算向攻击者发送比特币，也不一定能换取解密私钥，造成“人财两空”的局面。

建议，当资产已经因勒索病毒而造成损失时，不要慌张，更不要向攻击者支付比特币或其他加密货币，同时，交易所在收到这些勒索邮件时需额外警惕，千万不能向攻击者支付比特币或其他加密货币，必要时可寻求第三方安全公司的协助。