# 共建安全防线 共治安全环境 共享安全生态

2018 网络安全生态峰会

PeckShield

# 螳螂捕蝉，黄雀在后
# 以太坊智能合约的漏洞和陷阱

吴磊

PeckShield

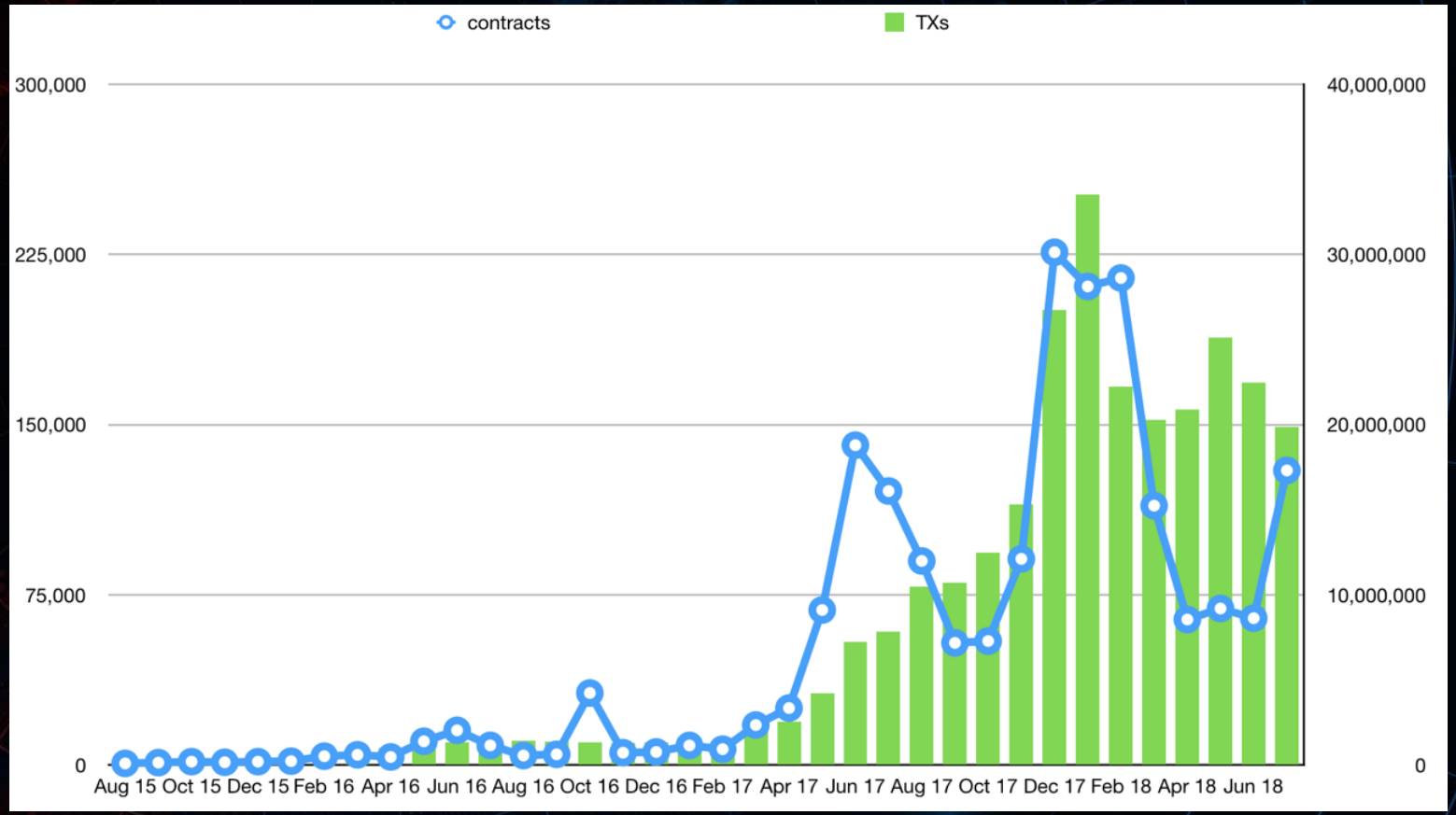
# Who am I ?

- 吴磊
- PeckShield联合创始人兼工程副总裁
- 美国北卡州立大学计算机博士
- 移动安全专家，对安卓平台的安全问题有深入研究
- 目前专注于区块链智能合约安全以及合约安全审计

PeckShield

# Blockchain Status Quo

- Ethereum in 2017
  - 1,090 Dapps & 700+ Tokens
  - 100,000 New Users Per Day
  - Daily Trading > 1,000,000
- Market Cap in 2018
  - 1,640 Cryptocurrencies
  - Market Cap > 330 Billion
  - Global GDP Rank: 28th



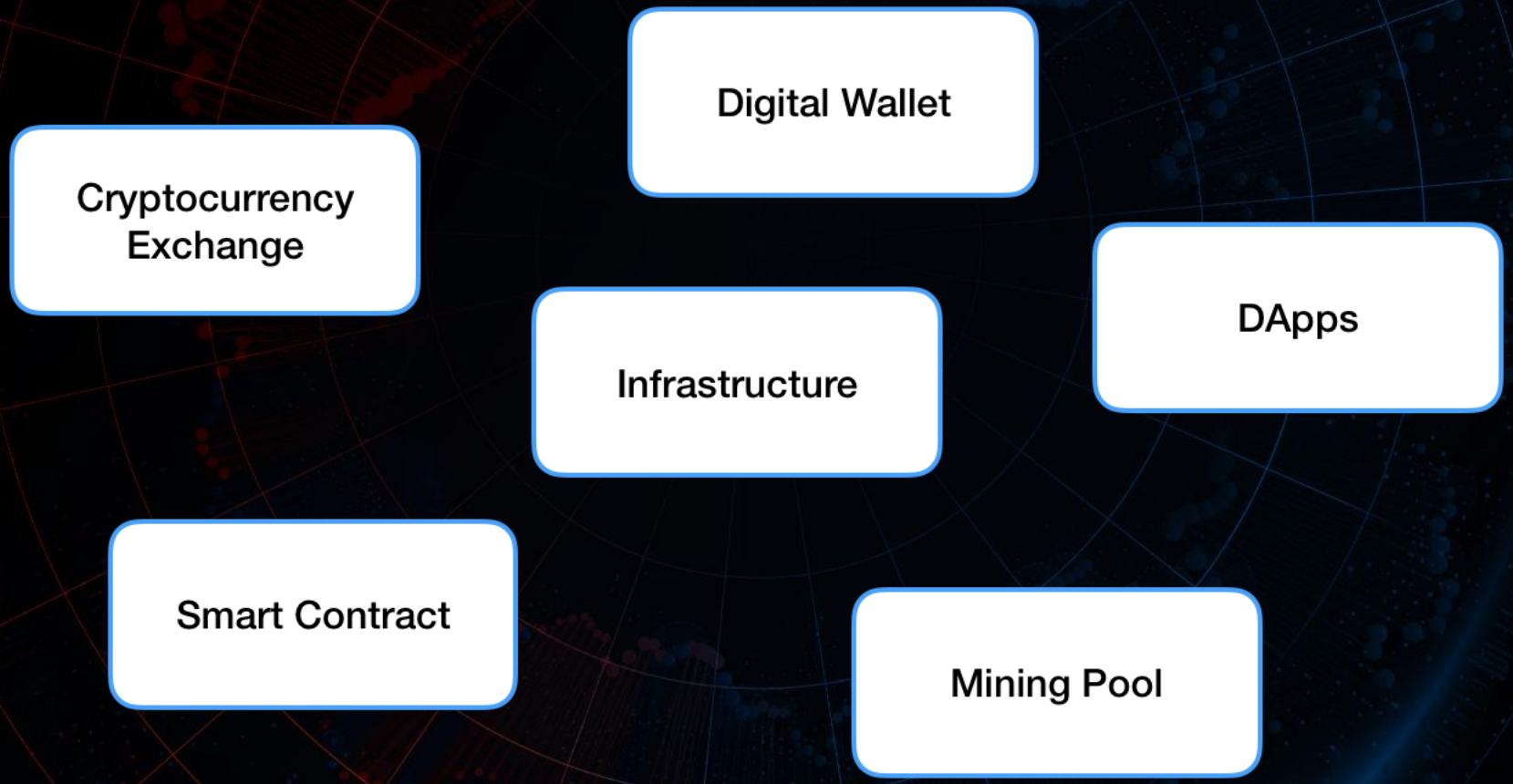
PeckShield

# Blockchain Security Incidents

| | |
|---|---|
| 2018/06 | Bithumb Hacks with $31 Million Dollars Stolen |
| 2018/05 | EDU, BAIC Smart Contracts Bugs |
| 2018/04 | BEC, SMT Smart Contracts Bugs |
| 2018/04 | Myetherwallet Suffer from DNS Hijacking |
| 2018/02 | BitGrail Hacks with Stolen Nano Tokens of 170 Million Dollars |
| 2018/01 | Coincheck Hacks with 530 Million Dollars Stolen |
| 2017/12 | Nicehash Hacks with 4700 BTC Missing with 62 Million Dollars |
| 2017/06 | Bithumb Hacks with 1 Billion Korean Yuan Loss and 30 Thousand User Info. Leaked |
| 2016/08 | Bitfinex Hacks with 120,000 BTC Stolen of 75Million Dollars |
| 2016/01 | Cryptsy Hacks with 13,000 BTC and 300,000 LTC Stolen |
| 2015/01 | Bitstamp Hacks with 19,000 BTC Stolen |
| 2014/03 | Poloniex Hacks with 12.3% BTC Lost |
| 2014/02 | Mt.Gox Hacks with Followed Bankruptcy |

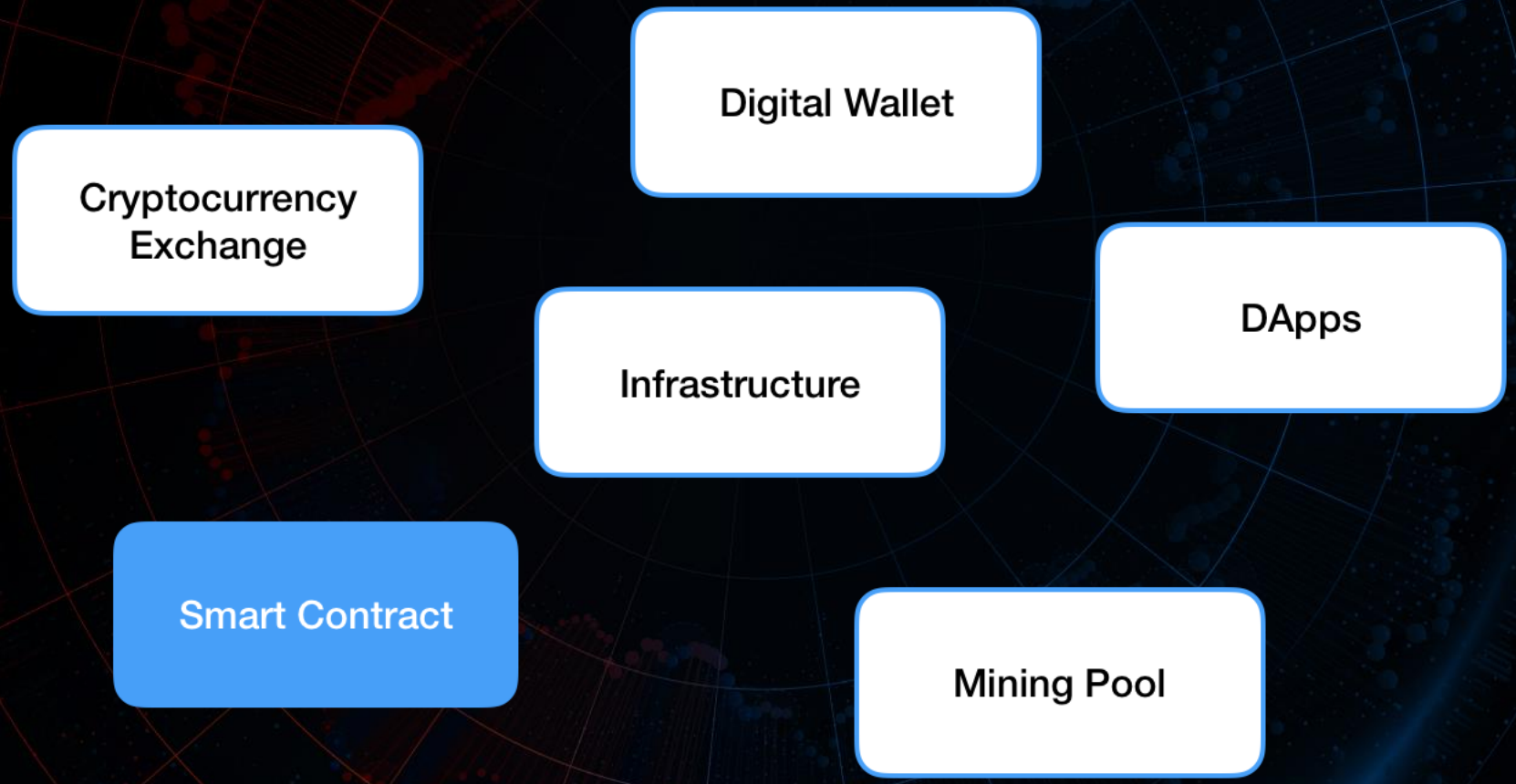
PeckShield

# Blockchain Ecosystem



PeckShield

# Blockchain Ecosystem – Smart Contract



PeckShield

# Security of Smart Contract

| Disclosed by PeckShield | |
|---|---|
| CodeName | CVE-ID |
| batchOverflow | CVE-2018-10299 |
| proxyOverflow | CVE-2018-10376 |
| transferFlaw | CVE-2018–10468 |
| ownerAnyone | CVE-2018-10705 |
| multiOverflow | CVE-2018-10706 |
| burnOverflow | CVE-2018-11239 |
| ceoAnyone | CVE-2018-11329 |
| allowAnyone1 | CVE-2018-11397 |
| allowAnyone2 | CVE-2018-11398 |
| tradeTrap1 | CVE-2018-12017 |
| tradeTrap2 | CVE-2018-12062 |
| tradeTrap3 | CVE-2018-12079 |
| | … |

PeckShield

# Security of Smart Contract

| Disclosed by PeckShield | |
|---|---|
| CodeName | CVE-ID |
| batchOverflow | CVE-2018-10299 |
| proxyOverflow | CVE-2018-10376 |
| transferFlaw | CVE-2018-10468 |
| ownerAnyone | CVE-2018-10705 |
| multiOverflow | CVE-2018-10706 |
| burnOverflow | CVE-2018-11239 |
| ceoAnyone | CVE-2018-11329 |
| allowAnyone1 | CVE-2018-11397 |
| allowAnyone2 | CVE-2018-11398 |
| tradeTrap1 | CVE-2018-12017 |
| tradeTrap2 | CVE-2018-12062 |
| tradeTrap3 | CVE-2018-12079 |
| | ... |

PeckShield

# Security of Smart Contract

| Disclosed by PeckShield | |
|---|---|
| CodeName | CVE-ID |
| batchOverflow | CVE-2018-10299 |
| proxyOverflow | CVE-2018-10376 |
| transferFlaw | CVE-2018–10468 |
| ownerAnyone | CVE-2018-10705 |
| multiOverflow | CVE-2018-10706 |
| burnOverflow | CVE-2018-11239 |
| ceoAnyone | CVE-2018-11329 |
| allowAnyone1 | CVE-2018-11397 |
| allowAnyone2 | CVE-2018-11398 |
| tradeTrap1 | CVE-2018-12017 |
| tradeTrap2 | CVE-2018-12062 |
| tradeTrap3 | CVE-2018-12079 |
| | … |

| Not Disclosed Yet |
|---|
| CVE-ID |
| CVE-2018-11310 |
| CVE-2018-11441 |
| CVE-2018-11446 |
| CVE-2018-11520 |
| CVE-2018-11521 |
| CVE-2018-11561 |
| CVE-2018-11582 |
| CVE-2018-11585 |
| CVE-2018-12105 |
| CVE-2018-12340 |
| CVE-2018-12341 |
| CVE-2018-12566 |
| … |

PeckShield

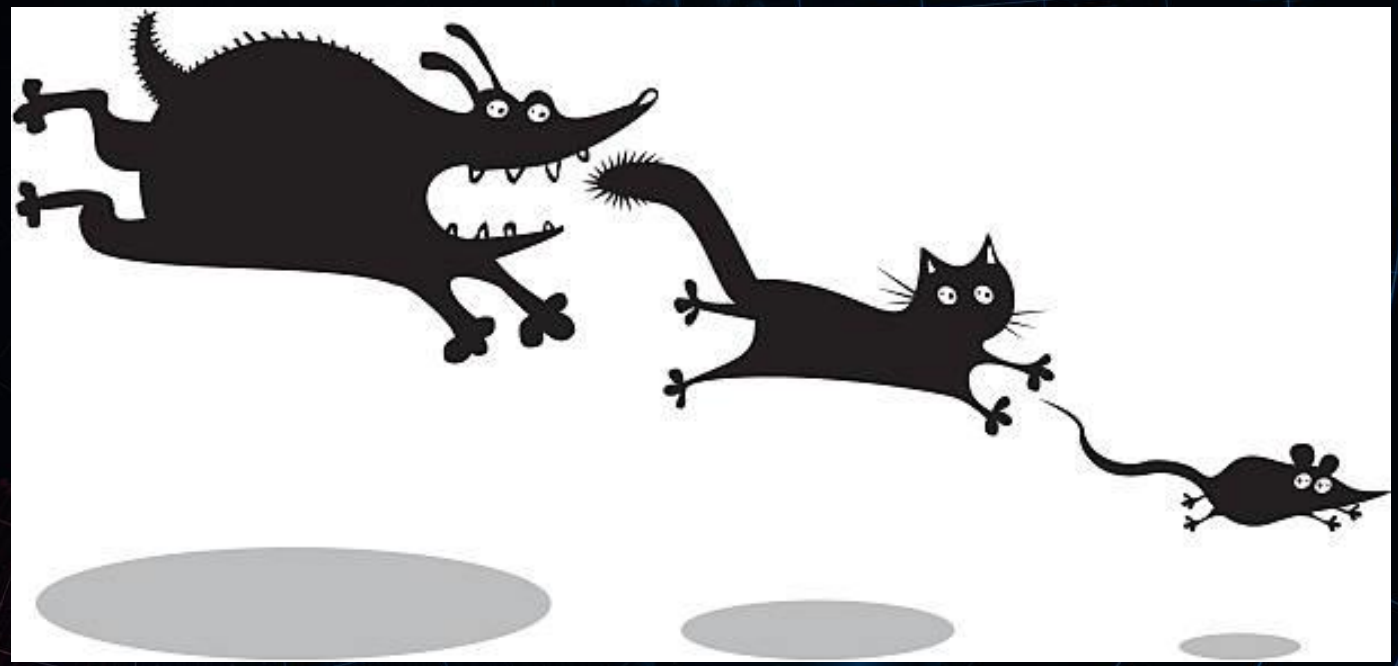
# Vulnerabilities and Traps

- Vulnerabilities in Ponzi-Like Crypto Games
  - Fomo3D
  - EOS Fomo3D
- Smart Contract Traps
  - tradeTrap
  - Other honeypot contracts

PeckShield

# Vulnerabilities and Traps

- Vulnerabilities in Ponzi-Like Crypto Games
  - Fomo3D
  - EOS Fomo3D
- Smart Contract Traps
  - tradeTrap
  - Other honeypot contracts



PeckShield

# Ponzi-Like Crypto Games

## Fomo3D and other ponzi games consume $46 million in ether

23 JUL, 2018 BY DAVID BORMAN

NEWS

**Fomo3D is a decentralized game hosted on the Ethereum network which asks players to purchase keys for a chance to win a cache of ether. The whole game is clearly a scam, but this hasn't stopped people from throwing money at it and other similar games, to a tune of $46 million in ether, bitcoin.com is reporting.**

The game has an exceedingly simple premise: Each time a key is purchased, the ether goes into the grand prize, and the 24 hour timer is reset. If the timer runs out, the user who last purchased a key will win the whole prize. The current total just for Fomo3D is about 21,460 eth or approximately $9,678,460 at the time of this writing. There are other games with similar premises also running on ethereum.

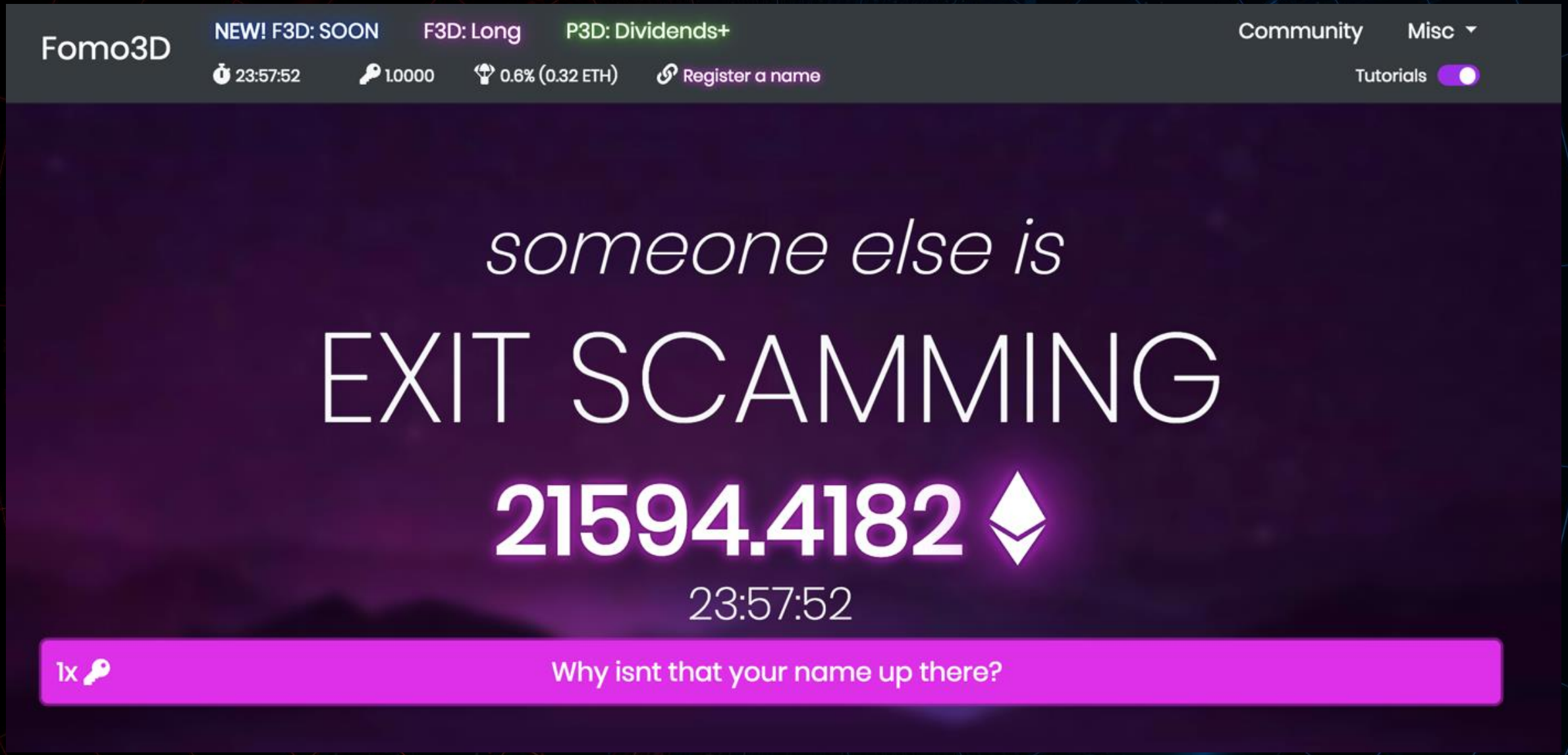
## FOMO3D: The new Ponzi game champion

FOMO3D has now blown away all old records, and its contract currently holds about 75,000 ETH worth around US$35.5 million. It's very much in a league of its own, and the money put into it has also spawned a series of copycats worth about another $15 million, some of which are actual scams as opposed to just scam-themed games.

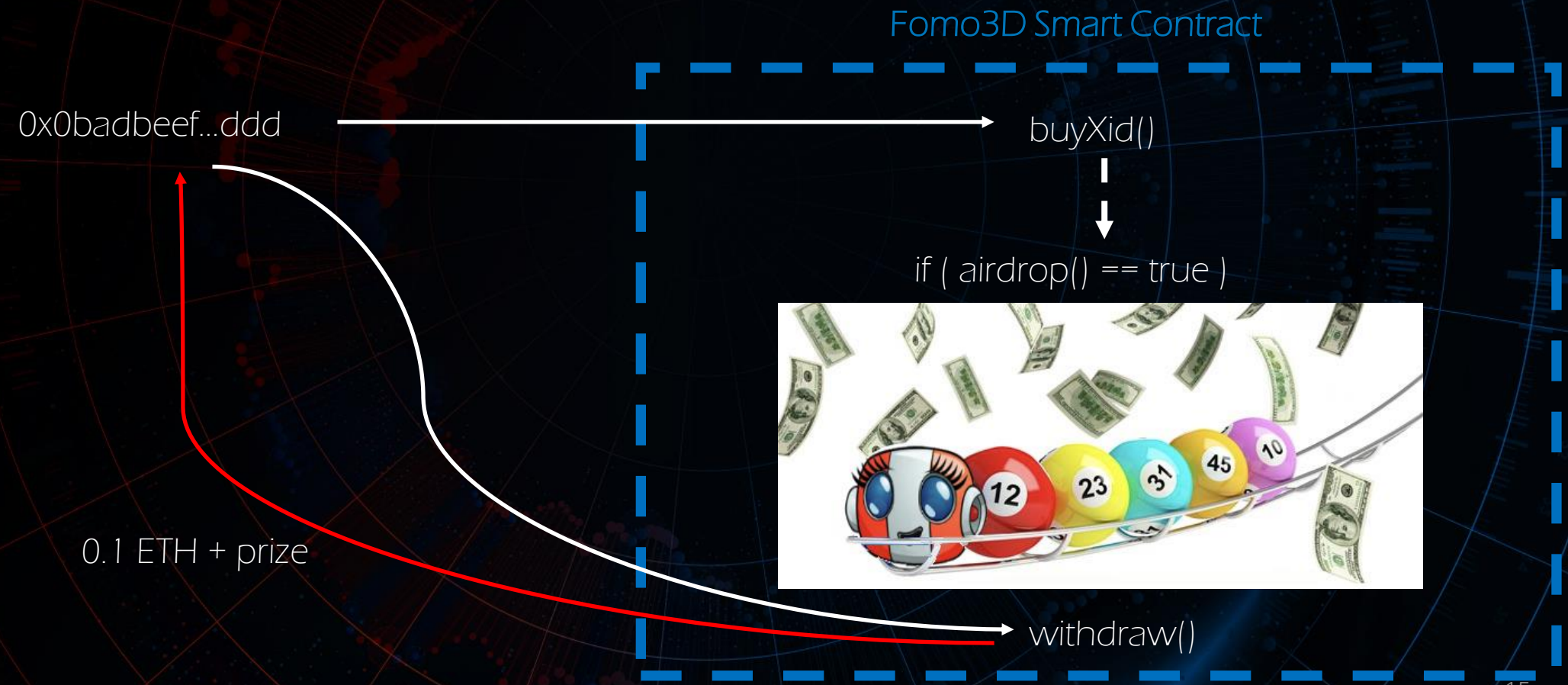
The essence of FOMO3D is that people can buy keys with Ether. The keys add time to the countdown timer. Each key increases the timer by 30 seconds up to a maximum of 24 hours. If no one buys a key before the timer runs out, the game ends, and that multi-million dollar pot is distributed among the various winners.

A player joins one of four different teams, but still competes individually. The teams instead affect how a player's resources are distributed between the keys, the P3D tokens and the exit scam itself.

PeckShield


Fomo3D
NEW! F3D: SOON
F3D: Long
P3D: Dividends+
Community
Misc
23:57:52
1.0000
0.6% (0.32 ETH)
Register a name
Tutorials
someone else is
EXIT SCAMMING
21594.4182
23:57:52
1x
Why isnt that your name up there?

# Fomo3D Lottery Airdrop Mechanism



PeckShield

# Don’t Implement Randomness in Ethereum Smart Contract!

```
1409    function airdrop()
1410        private
1411        view
1412        returns(bool)
1413    {
1414        uint256 seed = uint256(keccak256(abi.encodePacked(
1415
1416            (block.timestamp).add
1417            (block.difficulty).add
1418            ((uint256(keccak256(abi.encodePacked(block.coinbase)))) / (now)).add
1419            (block.gaslimit).add
1420            ((uint256(keccak256(abi.encodePacked(msg.sender)))) / (now)).add
1421            (block.number)
1422
1423        )));
1424        if((seed - ((seed / 1000) * 1000)) < airDropTracker_)
1425            return(true);
1426        else
1427            return(false);
1428    }
```

PeckShield

# What if msg.sender is a contract address?

```
contract human {
  constructor() {
    isHuman()
  }
}
```

```
270     modifier isHuman() {
271         address _addr = msg.sender;
272         uint256 _codeLength;
273
274         assembly {_codeLength := extcodesize(_addr)}
275         require(_codeLength == 0, "sorry humans only");
276         _;
277     }
```

0x0badbeef...ddd

7.1. **Subtleties.** Note that while the initialisation code is executing, the newly created address exists but with no intrinsic body code[4]. Thus any message call received by it during this time causes no code to be executed. If

[4]During initialization code execution, `EXTCODESIZE` on the address should return zero, which is the length of the code of the account while `CODESIZE` should return the length of the initialization code (as defined in H.2).

PeckShield

# Pwning Fomo3D: Pre-Calculated Contract Creation For Airdrop Prizes!

Now, the exploitation steps could be summarized in the following:

1. Pre-calculate the address *X* of the next contract that the attacker address is about to create [2];
2. If *X* can't be used to generate a *good seed* with the current *airDropTracker_*, goto step 1;
3. Create contract at address *X*;
4. Invoke *buyXid()* function from *X* to win the airdrop prize;
5. Invoke *withdraw()* function from *X* to get earnings calculated by the airdrop prize;



https://www.peckshield.com/2018/07/24/fomo3d/

PeckShield

# Attack in-the-wild

| TxHash | Block | Age | From | | To | Value | [TxFee] |
|---|---|---|---|---|---|---|---|
| 0x1bfa7ae79e2bd7... | 6080907 | 14 hrs 21 mins ago | 0x73b61a56cb93c1... | IN | 0x54833d94b55202... | 0.1 Ether | 0.0001995893 |
| 0x7eaccc82c50ed9... | 6076768 | 1 day 7 hrs ago | 0x73b61a56cb93c1... | IN | 0x54833d94b55202... | 0.1 Ether | 0.0037952116 |

| Type_TraceAddress | From | | To | Value | GasLimit |
|---|---|---|---|---|---|
| call_2 | 0x54833d94b55202... | | 0xa4bb1d29ca4efa... | 0.1 Ether | 2137597 |
| create_2_0 | 0xa4bb1d29ca4efa... | | 0xfd108e175998de... | 0.1 Ether | 2072360 |
| call_2_0_0 | 0xfd108e175998de... | | 0xa62142888aba83... | 0.1 Ether | 2032384 |
| call_2_0_0_3 | 0xa62142888aba83... | | 0xdd4950f977ee28... | 0.002 Ether | 1696395 |
| call_2_0_0_3_0 | 0xdd4950f977ee28... | | 0x4c7b8591c50f4a... | 0.002 Ether | 1660858 |
| call_2_0_0_4 | 0xa62142888aba83... | | 0xf9ba0955b0509a... | 0.001 Ether | 1663435 |
| call_2_0_1_0 | 0xa62142888aba83... | | 0xfd108e175998de... | 0.101039563771909293 Ether | 2300 |
| suicide_2_0_2 | 0xfd108e175998de... | | 0x73b61a56cb93c1... | 0.101039563771909293 Ether | 0 |

Attacker Contract

Boss Wallet Address

PeckShield

# Attack in-the-wild

| TxHash | Block | Age | From | | To | Value | [TxFee] |
|---|---|---|---|---|---|---|---|
| 0x1bfa7ae79e2bd7... | 6080907 | 14 hrs 21 mins ago | 0x73b61a56cb93c1... | IN | 0x54833d94b55202... | 0.1 Ether | 0.0001995893 |
| 0x7eaccc82c50ed9... | 6076768 | 1 day 7 hrs ago | 0x73b61a56cb93c1... | IN | 0x54833d94b55202... | 0.1 Ether | 0.0037952116 |

| Type_TraceAddress | From | To | Value | GasLimit |
|---|---|---|---|---|
| ✔ call_2 | 0x54833d94b55202... | 0xa4bb1d29ca4efa... | 0.1 Ether | 2137597 |
| ✔ create_2_0 | 0xa4bb1d29ca4efa... | 0xfd108e175998de... | 0.1 Ether | 2072360 |
| ✔ call_2_0_0 | 0xfd108e175998de... | 0xa62142888aba83... | 0.1 Ether | 2032384 |
| ✔ call_2_0_0_3 | 0xa62142888aba83... | 0xdd4950f977ee28... | 0.002 Ether | 1696395 |
| ✔ call_2_0_0_3_0 | 0xdd4950f977ee28... | 0x4c7b8591c50f4a... | 0.002 Ether | 1660858 |
| ✔ call_2_0_0_4 | 0xa62142888aba83... | 0xf9ba0955b0509a... | 0.001 Ether | 1663435 |
| ✔ call_2_0_1_0 | 0xa62142888aba83... | 0xfd108e175998de... | 0.101039563771909293 Ether | 2300 |
| ✔ suicide_2_0_2 | 0xfd108e175998de... | 0x73b61a56cb93c1... | 0.101039563771909293 Ether | 0 |

(1) This pre-created contact will create a "good" contract addr

(2) Create the "good" contract addr

(3) Send 0.1 ETH to buy keys

(4) Get the airdrop which is >= 0.1 ETH

(5) Withdraw

(6) Suicide and send everything to the boss

PeckShield

# EOS Fomo3D: 狼人游戏

## EOS版Fomo3D因漏洞终结 资金盘游戏背后现多重风险

2018-07-27 19:38 游戏

7月25日中午，火了两天的EOS狼人游戏(eosfo.io)遭到恶意攻击，致使奖励池内累积的94000个EOS变成负数。次日，该游戏开发团队声称，6万个EOS被攻击者套现。这个EOS版本的Fomo3D游戏在上线4天后宣告终结。

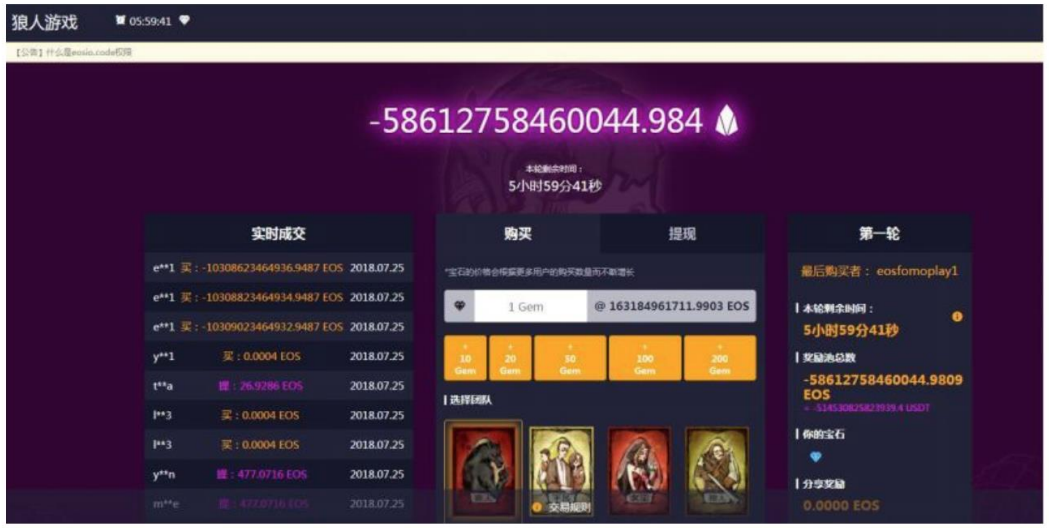
在该游戏停止运营前，国内有区块链安全团队提醒，由于该游戏的合约未开源，无法确认更多细节，存在风险。

一个未经长时间测试、没有安全审计就上线的智能合约游戏，如同定时炸弹一样，令用户资产随时存在被炸可能。

类“Fomo3D”游戏涌现，这些被看作庞氏骗局的博彩Dapp依旧吸引大批玩家，高赔率背后是币圈投机贪婪的图集。

EOS狼人游戏失窃6万EOS

2天时间，94000个EOS，EOS狼人游戏短时内筹集了近550万人民币。昨日（7月26日）狼人游戏团队以一纸公告宣称有人中奖，也承认了“溢出漏洞”的存在导致了“数据混乱”,漏洞致使攻击者提现了6万多个EOS。




EOS狼人游戏奖池币量出现负数

PeckShield

# EOS Fomo3D: 狼人游戏

## EOS Asset Multiplication Integer Overflow Vulnerability

```
66      asset& operator*=( int64_t a ) {
67         eosio_assert( a == 0 || (amount * a) / a == amount, "multiplication overflow or underflow" );
68         eosio_assert( -max_amount <= amount, "multiplication underflow" );
69         eosio_assert( amount <= max_amount,  "multiplication overflow" );
70         amount *= a;
71         return *this;
72      }
```

tokens (such as the official EOS token or some custom tokens defined by user). Recently we discovered a bug in asset's multiplication operator （operator *=） which makes the integer overflow check in the function to have no effect. If a developer uses asset multiplication in his EOS smart contract, he may need to face the risk of integer overflow.

http://blogs.360.cn/blog/eos-asset-multiplication-integer-overflow-vulnerability/

PeckShield

# Vulnerabilities and Traps

- Vulnerabilities in Ponzi-Like Crypto Games
  - Fomo3D
  - EOS Fomo3D
- Smart Contract Traps
  - tradeTrap
  - Other honeypot contracts

PeckShield

# tradeTrap: Trap of Highly-Manipulatable ERC20 Tokens

Quoted from our last blog [1], "publicly tradable ERC-20 tokens have considerable high market value. Various exchanges, either centralized (e.g., *Binance*, *Huobi.pro*, and *OKex*) or decentralized (e.g., *IDEX*, *EtherDelta*, *ForkDelta*), provide the marketplace by listing them, especially with high-liquidity ones, for public trading. Evidently, the transparency and security of their corresponding smart contracts is paramount. In practice, there is a de-facto requirement for these contract to be publicly verifiable on etherscan.io.

Moreover, reflecting the fundamental 'code-is-law' spirit and trust of blockchain technology, these contracts once deployed should not be further subject to centralized control or manipulation."

https://peckshield.com/2018/06/11/tradeTrap/

PeckShield

# Security Issue 1: Arbitrary Increase in Token Balance

```
131     function mintToken(address target, uint256 mintedAmount) onlyOwner {
132         balanceOf[target] += mintedAmount;
133         Transfer(0, owner, mintedAmount);
134         Transfer(owner, target, mintedAmount);
135     }
```

https://peckshield.com/2018/06/11/tradeTrap/

PeckShield

# Security Issue 2: Manipulatable Prices and Unfair Arbitrage

```
217     function setPrices(uint256 newSellPrice, uint256 newBuyPrice) onlyOwner public {
218         sellPrice = newSellPrice;
219         buyPrice = newBuyPrice;
220     }
221
222     /// @notice Buy tokens from contract by sending ether
223     function buy() payable public {
224         uint amount = msg.value / buyPrice;               // calculates the amount
225         _transfer(this, msg.sender, amount);              // makes the transfers
226     }
227
228     /// @notice Sell `amount` tokens to contract
229     /// @param amount amount of tokens to be sold
230     function sell(uint256 amount) public {
231         require(this.balance >= amount * sellPrice);      // checks if the contract has enough ether to buy
232         _transfer(msg.sender, this, amount);              // makes the transfers
233         msg.sender.transfer(amount * sellPrice);          // sends ether to the seller. It's important to do t
234     }
```

https://peckshield.com/2018/06/11/tradeTrap/

PeckShield

# Additional Overflows

```
230     function sell(uint256 amount) public {
231         require(this.balance >= amount * sellPrice);      // checks if the contra
232         _transfer(msg.sender, this, amount);              // makes the transfers
233         msg.sender.transfer(amount * sellPrice);          // sends ether to the se
234     }

390     //user is buying grx
391     function buy() payable returns (uint256 amount){
392         if(!usersCanTrade && !canTrade[msg.sender]) revert();
393         amount = msg.value * buyPrice;                    // calculates the amoun
394
395         require(balanceOf[this] >= amount);               // checks if it has eno
396         balanceOf[msg.sender] += amount;                  // adds the amount to bu
397         balanceOf[this] -= amount;                        // subtracts amount fro
398         Transfer(this, msg.sender, amount);               // execute an event ref
399         return amount;                                    // ends function and re
400     }
```

https://peckshield.com/2018/06/11/tradeTrap/

PeckShield

# Additional Overflows

After publishing our blog [1], we have been contacted by a number of affected cryptocurrency exchanges. As we believe the corresponding mitigation mechanism is now in place, it is the time to disclose the details of **tradeTrap**. As emphasized in [1], once smart contracts of publicly tradable ERC-20 tokens are deployed, they should not be further subject to centralized control or manipulation. Unfortunately, tradeTrap plagues 700+ ERC20 tokens and we have so far confirmed at least dozens of them are publicly tradable on current exchanges, including *Binance*, *Huobi.pro*, *OKex*, *OKCoinKR*, *CoinEgg*, *Kucoin*, *Allcoin*, *HitBTC*, *Bitbns*, *ZB*, *OTCBTC*, *CoinBene*, *COSS*, *EtherDelta*, *ForkDelta*, *IDEX*, *YEX*, *Tidex*, *Radar Relay*, *Yobit*, *WazirX*, *CoinExchange*, *CoinSpot*, *Bluetrade*, *CEX*, and *Livecoin*. The full list of tradeTrap-affected ERC20 tokens is available here.

https://peckshield.com/2018/06/11/tradeTrap/

PeckShield

# Other honeypot Contracts

## An analysis of a couple Ethereum honeypot contracts

Etherscan is an Ethereum blockchain explorer that, besides other features, allows developers to submit the code of the smart contracts they deploy. The main benefit of this feature is that it allows users to check what contracts do by reading their source code. Etherscan makes sure that the code matches the smart contract as deployed.

The list of verified contracts is long. As of this writing, Etherscan offers the source code for 26055 contracts, which can be browsed here.

On a lazy Sunday afternoon I decided to casually browse it to see what kind of contracts people were running and get a sense of what people use the blockchain for, and how well written and secure these contracts are. Most contracts I found implemented tokens, crowdsales, multi-signature wallets, ponzis, and.. honeypots!

Honeypot contracts are the most interesting findings to me. Such contracts hold ether, and *pretend* to do so insecurely. In short, they are scam contracts that try to fool you into thinking you can steal the ether they hold, while in fact all you can do is *lose* ether.

https://medium.com/@jsanjuas/an-analysis-of-a-couple-ethereum-honeypot-contracts-5c07c95b0a8d

## 以太坊蜜罐智能合约分析

Knownsec知道创宇 2018-06-28 共87401人围观 区块链安全

### 0×00 前言

**在学习区块链相关知识的过程中，拜读过一篇很好的文章《The phenomenon of smart contract honeypots》，作者详细分析了他遇到的三种蜜罐智能合约，并将相关智能合约整理收集到Github项目smart-contract-honeypots。**

本文将对文中和评论中提到的 smart-contract-honeypots 和 Solidlity-Vulnerable 项目中的各蜜罐智能合约进行分析，根据分析结果将蜜罐智能合约的欺骗手段分为以下四个方面：

古老的欺骗手段

神奇的逻辑漏洞

新颖的赌博游戏

黑客的漏洞利用

基于已知的欺骗手段，我们通过内部的以太坊智能合约审计系统一共寻找到 118 个蜜罐智能合约地址，一共骗取了 34.7152916 个以太币（2018/06/26 价值 102946 元人民币），详情请移步文末附录部分。

http://www.freebuf.com/articles/blockchain-articles/175915.html

PeckShield

# PrivateBank: Another TheDAO ?

```
27      function CashOut(uint _am)
28      {
29          if(_am<=balances[msg.sender])
30          {
31
32              if(msg.sender.call.value(_am)())
33              {
34                  balances[msg.sender]-=_am;
35                  TransferLog.AddMessage(msg.sender,_am,"CashOut");
36              }
37          }
38      }
39
40      function() public payable{}
41
42  }
```

PeckShield

# PrivateBank: Another TheDAO ?

```
3  contract Private_Bank
4  {
5      mapping (address => uint) public balances;
6
7      uint public MinDeposit = 1 ether;
8
9      Log TransferLog;
10
11     function Private_Bank(address _log)
12     {
13         TransferLog = Log(_log);
14     }
```

```
44 contract Log
45 {
46
47     struct Message
48     {
49         address Sender;
50         string  Data;
51         uint Val;
52         uint  Time;
53     }
54
55     Message[] public History;
56
57     Message LastMsg;
58
59     function AddMessage(address _adr,uint _val,string _data)
60     public
61     {
62         LastMsg.Sender = _adr;
63         LastMsg.Time = now;
64         LastMsg.Val = _val;
65         LastMsg.Data = _data;
66         History.push(LastMsg);
67     }
68 }
```

PeckShield

# PrivateBank: Another TheDAO ?

| | |
|---|---|
| TxHash: | 0x091baded88d9858ff68f037245f33902e |
| TxReceipt Status: | Success |
| Block Height: | 5061587 (1059081 block confirmations) |
| TimeStamp: | 181 days 6 hrs ago (Feb-09-2018 11:05:5 |
| From: | 0x3476e2628006a7f74285c44b501437b |
| To: | [Contract 0x95d34980095380851902ccd |
| Value: | 0 Ether ($0.00) |
| Gas Limit: | 489374 |
| Gas Used By Txn: | 407812 |
| Gas Price: | 0.000000002 Ether (2 Gwei) |
| Actual Tx Cost/Fee: | 0.000815624 Ether ($0.29) |
| Nonce & {Position}: | 2 \| {81} |
| Input Data: | |

```
ffffffffffff1681526020018281526
0000000000000000000000000081525060200193505050506000604051808303816000878 03b151561045757600080fd5b6102c65a0
3f1151561046857600080fd5b5050505b5600a165627a7a723058200c0ba4aaf72a552e930e4406847884a17a862658ce48f787d955f5
ca92acab34002900000000000000000000000f8681dad1ce4f1f414fb07fc07f81a3a82e91d8f
```

View Input As

Contract 0xF8681Dad1cE4f1f414FB07FC07f81a3A82E91D8f

Home / Accounts / Address

Sponsored Link: genEOS: next-gen blockchain for businesses. Testnet launched. Fast, scalable, tech support available. Join now!

**Contract Overview**

| | |
|---|---|
| Balance: | 0 Ether |
| Ether Value: | $0 |
| Transactions: | 1 txn |

**Misc**

More Options

| | |
|---|---|
| Address Watch: | Add To Watch List |
| Contract Creator: | 0x78737ca0804983... at txn 0x63e3a140b0bea6... |

Transactions | Code | Events | Comments

Are you The Contract Creator? Verify And Publish *New your Contract Source Code Today!

Switch To Opcodes View | Find Similiar Contracts

```
0x6060604052600436106100c5576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff1680632fac8979146100c7578063
474dacce146100dc5780634b9067141461012057806340c2f04a41461019a57806389977cdf1461021f5780639003e39a1461027457806390a68455146102e55780639303
39be1461031e578063a21f03681461036f578063be9474bb14610469578063c144811f146104ad578063edbbdf2e146104b7578063f85f97dd14610508575b005b341561
00d257600080fd5b6100da61055d565b005b34156100e757600080fd5b61011e600480803573ffffffffffffffffffffffffffffffffffffffff1690602001909190803
151590602001909190505061061856 5b005b6101986004808035 73ffffffffffffffffffffffffffffffffffffffff1690602001909190803590602001909190803590602
001908201803590602001908080601f0160208091040260200160405190810160405280939291908181526020018383808284378201915050505050509190505061072 9
565b005b34156101a557600080fd5b61021d600480803573ffffffffffffffffffffffffffffffffffffffff1690602001909190803590602001909190803590602001909
820180359060200190808060 1f0160208091040260200160405190810160405280939291908181526020018383808284378201915050505050509190505061082056 5b00
5b341561022a57600080fd5b610232610a70565b604051808273ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
168152602001915050604051809103 90f35b6102e3600480803573ffffffffffffffffffffffffffffffffffffffff1690602001909190803590602001908201803590602
001908080601f0160208091040260200160405190810160405280939291908181526020018383808284378201915050505050509190505061 0a95565b005b34156102f0
57600080fd5b61031c600480803573ffffffffffffffffffffffffffffffffffffffff1690602001909190505061 0b88565b005b3415610329576000 80fd5b6103556004
80803573ffffffffffffffffffffffffffffffffffffffff1690602001909190505061 0c27565b604051808215151515815260200191505060405180910390f35b341561
```

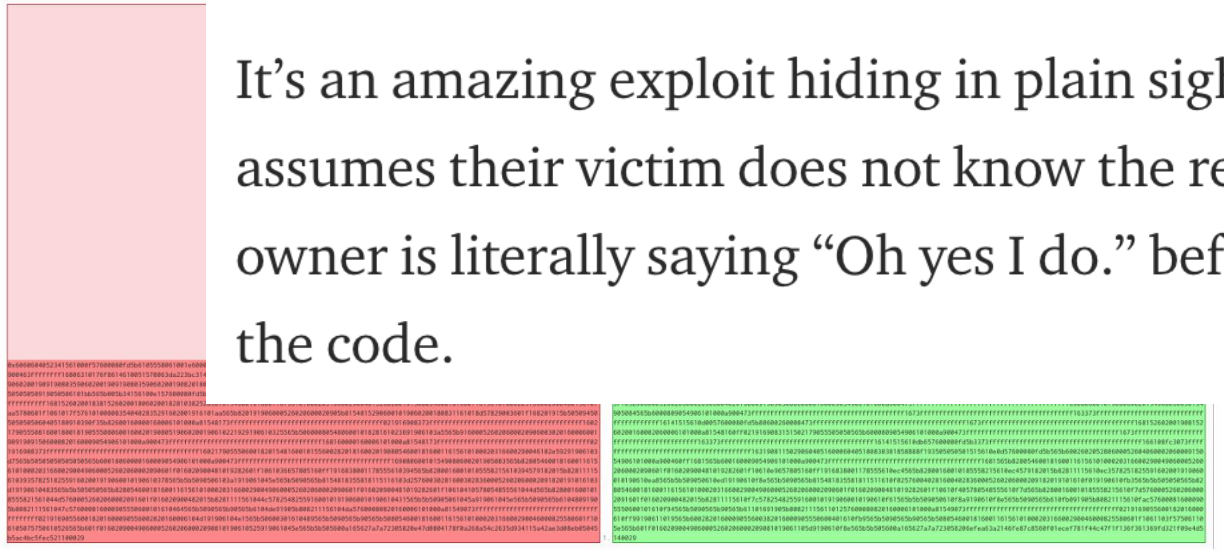
PeckShield

# PrivateBank: Another TheDAO ?

Here is the diff check between the bytecode in the Honey Pot and the code that has actually been deployed on the network.

(Left is the by

Simple, they exploit their contract's re-entrancy bug! But they've essentially made themselves the only address allowed to execute that exploit.



It's an amazing exploit hiding in plain sight. The attacker immediately assumes their victim does not know the re-entrancy exploit but the Honey Pot owner is literally saying "Oh yes I do." before the would-be hacker first sees the code.

It contains almost three times as much logic. So what else is going on?



https://medium.com/coinmonks/dissecting-an-ethereum-honey-pot-7102d7def5e0

PeckShield

# Mitigations?

# Better Blockchain Ecosystem Together


Website: https://peckshield.com
Email: contact@peckshield.com