

# Security Analysis of EOSIO Smart Contracts论文分享

汇报人：彭婷  
日期：2022年7月31日



北京大学  
PEKING UNIVERSITY



CONTENTS

# 目录

01



概要与介绍

02



背景

03



*EOSIO*智能合约漏洞

04



技术难点与解决

05



系统设计与实现

06



效果



# 01 概要与介绍

由于工作证明共识的带来的吞吐量有限（例如*TPS*），因此不能使用传统的区块链平台（例如比特币和以太坊）来支持高性能应用程序。

研究人员提出了不同的共识协议，例如权益证明（*PoS*）和委托权益证明（*DPoS*）以解决性能问题。

作为最具代表性的*DPoS*平台之一，*EOSIO*已成为最活跃的全球社区之一。*EOSIO*采用了基于*DPoS*共识协议的多线程机制，能够实现数百万的*TPS*。*EOSIO*的性能优势使其在去中心化应用程序（*DApps*）开发人员中很受欢迎。

智能合约是一种计算机协议，允许用户以便捷安全的方式进行数字协商。与传统合约法相比，智能合约的交易成本大大降低，并且共识协议确保了其执行的正确性。*EOSIO*智能合约可以用C++编写，然后将其编译为WebAssembly（又名Wasm）并在EOS虚拟机（EOS VM）中执行。

但是，要保证执行智能合约的安全性并不容易，特别是*EOSIO*。*EOSIO*智能合约中发现了许多漏洞，遭受了严重的攻击，造成了巨大的经济损失。



## 02 背景

作为第一个工业级规模的分布式操作系统，*EOSIO*平台可以实现高性能，即数百万个*TPS*，以有效执行复杂的*DApp*。它如此高效地执行，事实上很大程度归功于它使用的共识算法，即*DPoS*。与传统的*PoW*（比特币和以太坊）相比，它不会在不必要的挖掘过程中花费大量的计算资源。

*EOSIO*  
关键  
概念



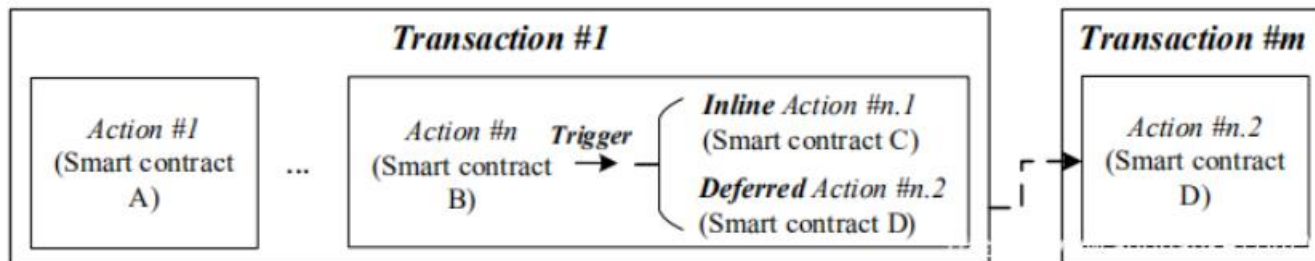
*EOSIO*中的帐户是识别实体的基本单位，它可以触发与*EOSIO*中其他帐户的交易。

此外，为了确保帐户安全并防止身份欺诈，*EOSIO*实施了基于权限的高级访问控制系统。

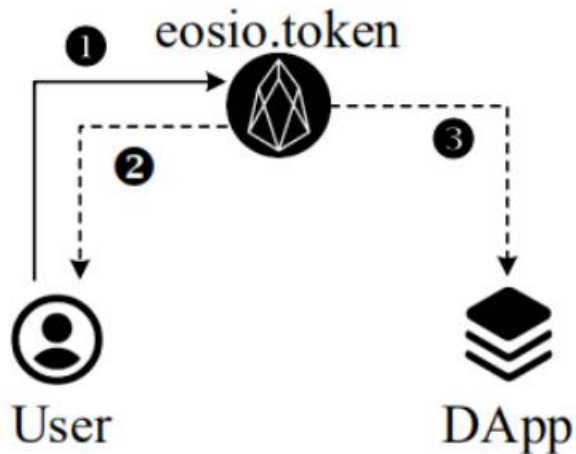


```
1 void apply(uint64_t receiver, uint64_t code,  
    uint64_t action) {  
2     if(action == N(onerror)) {  
3         check(code == N(eosio), "exception captured");  
4     }  
5     auto self = receiver;  
6     if((code == self || code == N(eosio.token))) {  
7         switch(action) {  
8             EOSIO_DISPATCH_HELPER(TYPE, MEMBERS)  
9         }  
10    }  
11 }
```

事务是由节点验证的基本单位，它被打包在块中。



除了交易和动作外，还有另一种排他性机制，即 *notification*。



- ① Invoke transfer in eosio.token  
Code: eosio.token  
Action: transfer  
Receiver: eosio.token
- ② Notify payer if the payment succeeds
- ③ Notify payee if the payment succeeds  
Code: eosio.token  
Action: transfer  
Receiver: DApp



*EOSIO*智能合约用C++编写，然后编译为WebAssembly (Wasm) 字节码，该字节码将在EOS VM中执行。

*EOSIO Wasm*二进制文件称为模块。



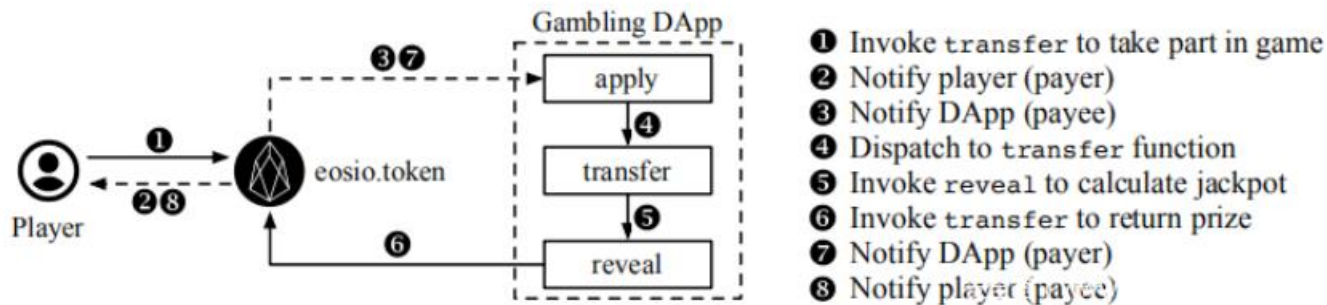
# 03 EOSIO智能合约漏洞

## EOSIO智能合约漏洞

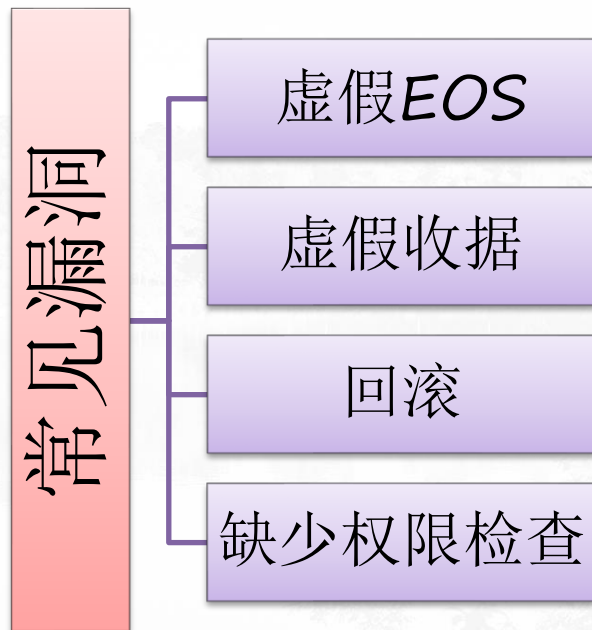


北京大学  
PEKING UNIVERSITY

在合约执行的生命周期内，可以随时进行攻击。因此首先以DApp为例，介绍智能合约执行的一般生命周期，如下图所示。

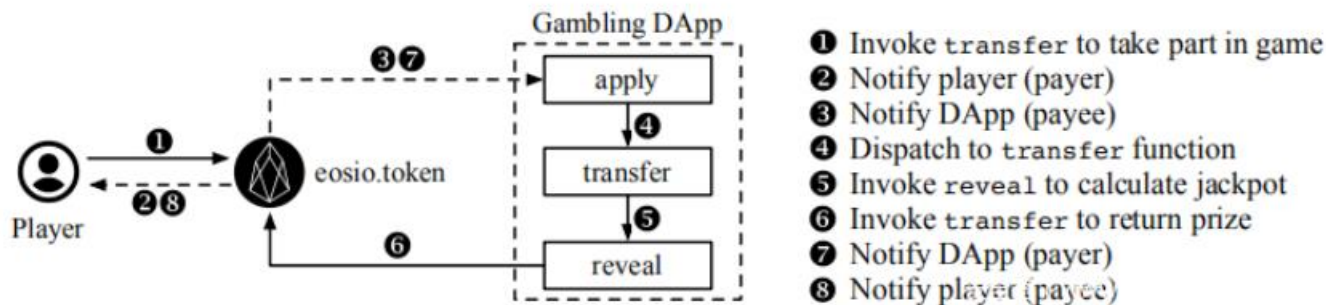


与合约执行生命周期有关的四种常见漏洞。





任何人都可以创建和发行称为EOS的token，因为token名称和符号在EOSIO中不需要符合唯一性。因此，在下图中的步骤3对code进行不正确验证可能会导致漏洞。



**漏洞描述：** 由于`eosio.token`的源代码是完全公开的，因此任何人都可以复制其源代码并发行具有相同名称，符号和代码的token。

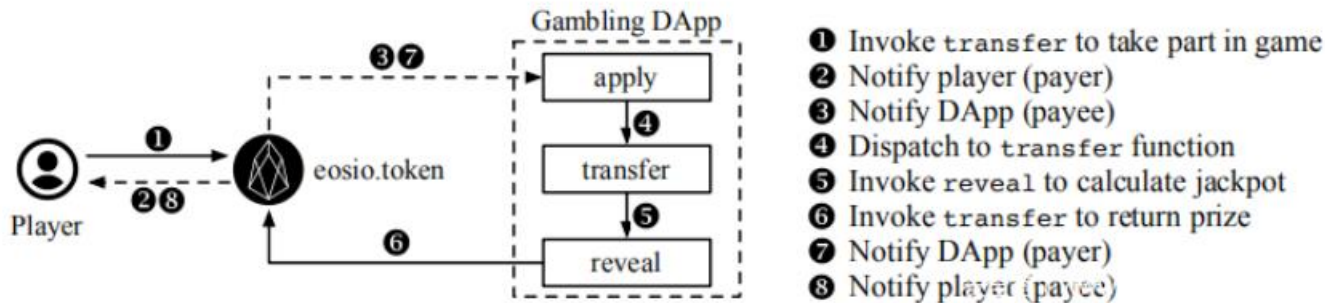


## 虚假收据



北京大学  
PEKING UNIVERSITY

如果DApp开发人员对code进行了全面检查，则通知将由调度程序转发到transfer，如下图中的步骤4所示。但是，如果开发人员在此步骤中未执行验证，则DApp可以被攻击。



**漏洞描述：**通知可以转发，并且code不会更改。因此，DApp可能会被同时扮演发起者和共犯双重角色（帐户）的攻击者欺骗。

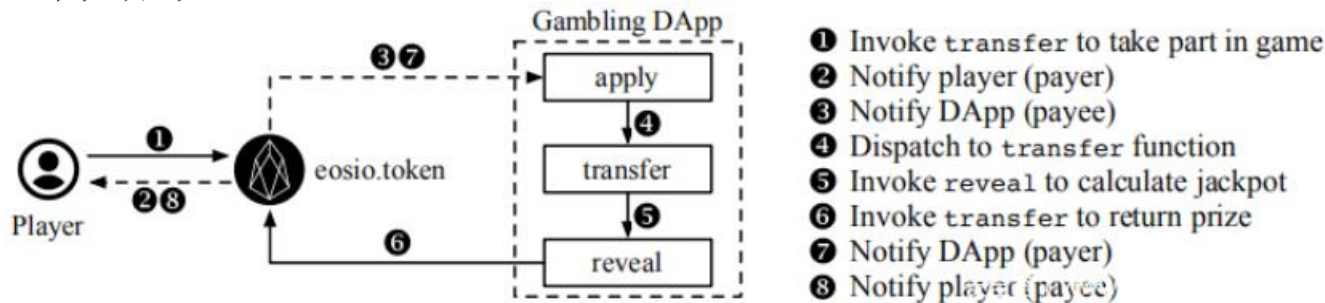
## 回滚



如下图，*transfer*和*reveal*是关键函数。

在*transfer*中，*DApp*处理随玩家转帐而收到的压注；

在*reveal*中，开发人员经常使用各种链上状态值作为种子来生成伪随机数，并最终通过将生成的数字与玩家的输入进行比较来获得结果。

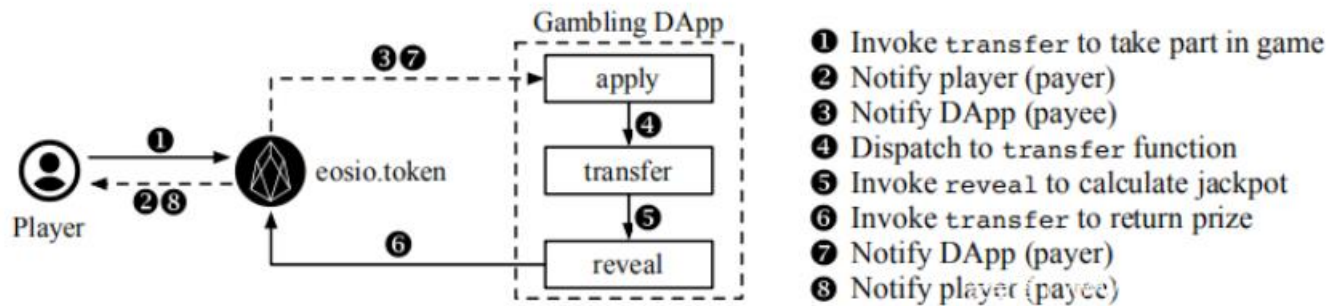


**漏洞描述：**即使开发人员仔细检查了输入的每个参数，并在采取任何敏感措施之前检查了调用者的权限，与前图中的模型匹配的游戏仍然可能受到攻击。



## 缺少权限检查

在执行任何敏感操作之前，开发人员应检查动作是否携带了相应的权限。例如，在下图的步骤5之前，*DApp*应检查调用者是否可以代表实际付款人参加游戏。

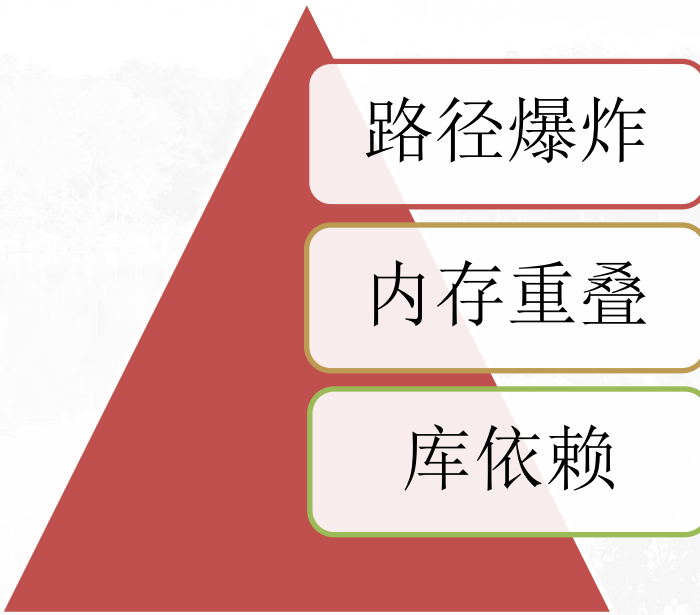


**漏洞描述：** 权限检查由EOSIO中的`require_auth(acct)`强制执行，用于检查调用者是否已被`acct`授权来触发相应的函数。



# 04 技术难点与解决

在EOSIO中主要存在3个挑战，如下图所示：



路径爆炸

内存重叠

库依赖

在EOSIO中，路径爆炸主要是由于两种情况造成的：  
执行条件跳转指令（例如`br_if`）或调用函数调用。

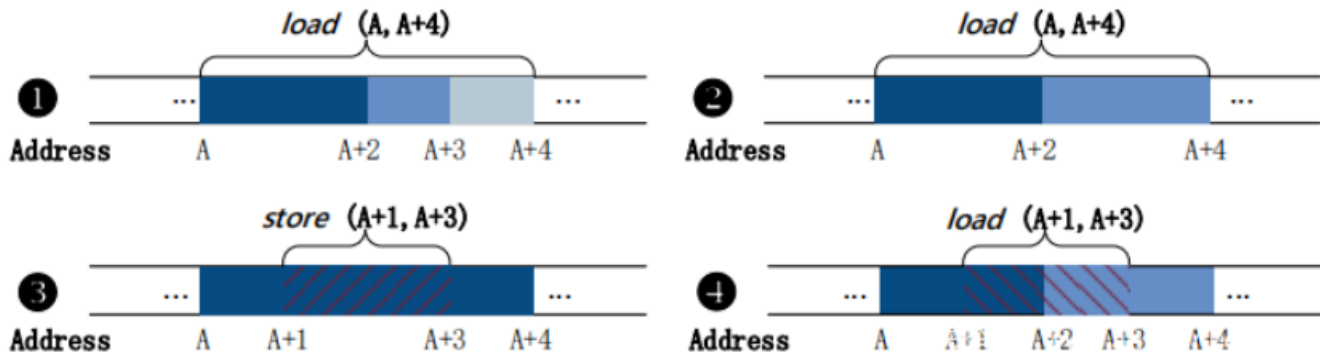
采用启发式指导的剪枝方法 (*heuristic-guided pruning approach*) 来解决挑战。

## 内存重叠



*Wasm*的存储区可以看作是未解释字节的向量，这意味着用户可以通过*load*和*store*以不同的值类型来解释这些原始位。

模拟内存的传统方法是使用线性数组，但是由于模拟了*EOSIO*智能合约的稀疏内存布局，非常耗费内存。因此决定使用*key-value*映射来模拟内存，此策略可能导致内存重叠，如下图所示：





通过合并分配的内存可以解决内存重叠问题。

Wasm提供了20多个与内存访问相关的指令。

首先为遇到的所有与存储相关的指令创建键-值映射，其中值是按位存储的数据。

然后，可以根据所提出的内存合并算法处理范围相邻或重叠的两个键的情况，该算法将更新相应的数据块以保证执行的准确性。

总之就是，尽力确保任意两个任意键之间的间隔至少为一位。通过这样做可以成功克服上图中提出的挑战。



为了促进智能合约的开发，*EOSIO*允许将外部函数作为库导入，这意味着这些导入函数的主体将不会编译为*Wasm*字节码。*EOSIO*官方为*DApp*开发人员提供了大量诸如系统库之类的函数。它们已在许多（如果不是大多数）智能合约中得到广泛使用。结果，由于缺少那些导入的函数调用的主体，因此分析将被错误地终止。

为了解决依赖关系，提出了一种按需且语义感知的方法来模拟导入的函数。



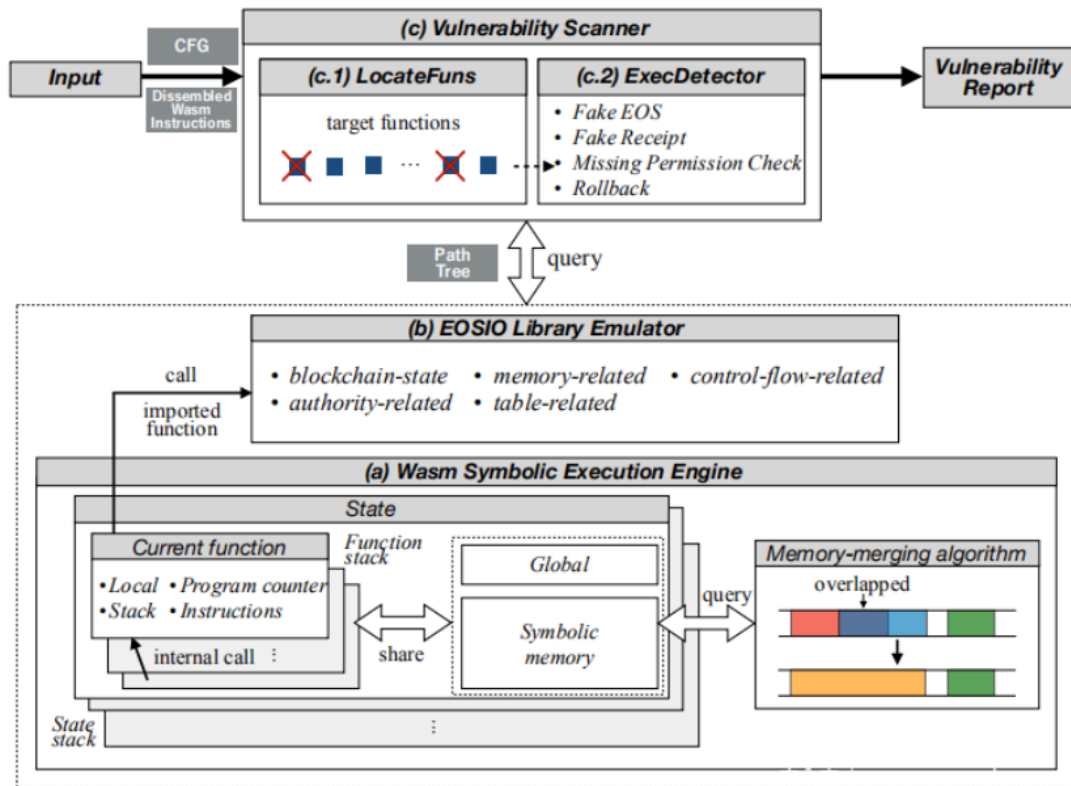
05

# 系统设计与实现

# EOSAFE架构



北京大学  
PEKING UNIVERSITY



如图所示，描绘了EOSAFE的总体架构，该架构以EOSIO智能合约的Wasm字节码作为输入，并最终确定该字节码是否易受攻击。

- Wasm符号执行引擎
- EOSIO 库模拟器
- 漏洞扫描程序 (Scanner)

使用按需和语义感知的方法来解决EOSIO库依赖关系。已经手动分析了排名前100位的流行DApp的智能合约和现有已知的易受攻击的智能合约，以从Function部分提取所有导入的函数。然后，根据其主要函数（如下表所示）将所有导入的函数分为五类进行模拟。最后，可以从模拟的导入函数中检索副作用。

Category	Imported Function Examples
blockchain-state	tapos_block_num
	current_time
memory-related	memcpy
	memmov
control-flow-related	eosio_exit
	eosio_assert
authority-related	require_auth
	require_auth_2
table-related	db_get_i64
	db_update_i64

为了检测多个漏洞，*Scanner*被设计为执行检测的通用框架。它主要包括两个步骤，即查找可疑函数和检测漏洞。

只需要关注有价值的函数，这些函数可以调用具有更改链上状态的外部函数，包括`send_inline`，`db_update_i64`和`db_store_i64`。根据观察，在大多数情况下，这些有价值的函数可以试探性地视为目标函数，从而可以显著减少分析时间。结果，借助*CFG*和路径树（由约束和有价值的函数组成），可以高效、准确地识别漏洞。



# 06 效果

利用*Octopus*构建Wasm字节码的*CFG*，并使用*Z3*定理证明（版本4.8.6）作为约束求解器。所有其他主要组件，包括Wasm符号执行引擎，库模拟器和漏洞扫描程序，均已设计和实现的前提下，对其效果进行了测试。评估以下三个方面

- *EOSAFE*在检测*EOSIO*智能合约的漏洞方面的准确性如何？
- 这些漏洞在生态系统中是否普遍存在？
- 攻击者利用了多少智能合约，这些攻击的影响是什么？



## 漏洞检测准确性



北京大学  
PEKING UNIVERSITY

在52份智能合约中，*EOSAFE*将26份标记为易受攻击，只有一个漏报性案例（属于回滚），没有误报性，准确率和召回率分别为100%和96.97%。

Vulnerability	# Samples(Vul/Safe)	Precision	Recall	F1-measure
Fake EOS	14 (7/7)	100.00%	100.00%	100.00%
Fake Receipt	10 (5/5)	100.00%	100.00%	100.00%
Rollback	18 (9/9)	100.00%	88.89%	94.12%
Permission	10 (6/4)*	100.00%	100.00%	100.00%
<b>Total</b>	<b>52 (27/25)</b>	<b>100.00%</b>	<b>96.97%</b>	<b>98.46%</b>



## 漏洞的普遍程度



北京大学  
PEKING UNIVERSITY

使用*DAppTotal*—一种可靠的多平台*DApp*浏览器来标记游戏*DApp*，并使用此类合约（17,394）进行回滚漏洞检测。对于其他三种漏洞，将扫描器应用于所有53,666个合约（请参见下表）。

Type	# Contracts	# Vulnerable (%)	# Unique	# Vulnerable (%)
Fake EOS	53,666	1,457 (2.71%)	5,574	272 (4.88%)
Fake Receipt	53,666	7,143 (13.31%)	5,574	2,192 (39.33%)
Rollback	17,394	1,149 (6.61%)	913	84 (9.20%)
Permission	53,666	8,373 (15.60%)	5,574	662 (11.88%)
<b>Total</b>	<b>53,666</b>	<b>13,752 (25.63%)</b>	<b>5,574</b>	<b>2,759 (49.50%)</b>

上表显示了总体结果，在53,666个智能合约中，有25%以上是易受攻击的（请参阅第3列）。

## 存在攻击



北京大学  
PEKING UNIVERSITY

Type	# Attacks	# Attacker / Victims	Financial Loss (\$)	# Verified
Fake EOS	9	10 / 9	652,428.48	8
Fake Receipt	27	28 / 17	1,020,831.94	7
Rollback	12	12 / 9	52,984.00	12
Permission	183	- / 144	-	-
<b>Total</b>	<b>48*</b>	<b>50 / 34*</b>	<b>1,726,244.42</b>	<b>27</b>

总体结果如上表所示。总共确定了**48**个攻击，包括**9**个虚假EOS攻击，**27**个虚假收据攻击和**12**个回滚攻击。此外，还确定了**183**个未执行权限检查的调用操作（属于**144**个合约）。值得注意的是，对于这些缺少权限检查的操作，其中一些是预料之中的。很难区分它们是否是攻击，并且无法估计经济损失。因此将其视为滥用行为，而不是攻击。

这篇论文是首个有关检测EOSIO智能合约中的安全漏洞工作。在文中提出了EOSAFE，是一个准确且可扩展的框架，该框架能够检测EOSIO特定的漏洞。

实验结果表明，EOSAFE具有良好的性能。

大规模评估研究进一步揭示了生态系统中的严重安全问题，即超过25%的智能合约易受攻击，并且成功造成了许多著名攻击事件。

# 2022

## 汇报完毕 感谢您的聆听

汇报人：彭婷



北京大学  
PEKING UNIVERSITY