原创

以太坊智能合约OPCODE逆向之理论基础篇

3 0

2018-07-12 阅读 691

作者：Hcamael@知道创宇404区块链安全研究团队

在我们对etherscan等平台上合约进行安全审查时，常常会遇到没有公布Solidity源代码的合约，只能获取到合约的OPCODE，所以一个智能合约的反编译器对审计无源码的智能合约起到了非常重要的作用。

目前在互联网上常见的反编译工具只有porosity1，另外在Github上还找到另外的反编译工具ethdasm2，经过测试发现这两个编译器都有许多bug，无法满足我的工作需求。因此我开始尝试研究并开发能满足我们自己需求的反编译工具，在我看来如果要写出一个优秀的反汇编工具，首先需要有较强的OPCODE逆向能力，本篇Paper将对以太坊智能合约OPCODE的数据结构进行一次深入分析。

基础

智能合约的OPCODE是在EVM(Ethereum Virtual Machine)中进行解释执行，OPCODE为1字节，从 `0x00 - 0xff` 代表了相对应的指令，但实际有用的指令并没有0xff个，还有一部分未被使用，以便将来的扩展

具体指令可参考Github3上的OPCODE指令集，每个指令具体含义可以参考相关文档4

IO

在EVM中不存在寄存器，也没有网络IO相关的指令，只存在对栈(stack)，内存(mem)，存储(storage)的读写操作

stack

使用的push和pop对栈进行存取操作，push后面会带上存入栈数据的长度，最小为1字节，最大为32字节，所以OPCODE从 `0x60-0x7f` 分别代表的是 `push1-push32`

`PUSH1` 会将OPCODE后面1字节的数据放入栈中，比如字节码是 `0x6060` 代表的指令就是 `PUSH1 0x60`

除了 `PUSH` 指令，其他指令获取参数都是从栈中获取，指令返回的结果也是直接存入栈中

mem

内存的存取操作是 `MSTORE` 和 `MLOAD`

```
MSTORE(arg0, arg1)  从栈中获取两个参数，表示
MEM[arg0:arg0+32] = arg1
```

```
MLOAD(arg0)  从栈中获取一个参数，表示
PUSH32(MEM[arg0:arg0+32])
```

因为 `PUSH` 指令，最大只能把32字节的数据存入栈中，所以对内存的操作每次只能操作32字节

但是还有一个指令 `MSTORE8` ，只修改内存的1个字节

```
MSTORE(arg0, arg1)  从栈中获取两个参数，表示
MEM[arg0] = arg1
```

内存的作用一般是用来存储返回值，或者某些指令有处理大于32字节数据的需求

作者介绍

Seebug漏洞平台

关注 专栏

文章	阅读量	获赞	作者排名
301	181.8K	1.4K	636

精选专题

腾讯云原生专题

云原生技术干货，业务实践落地。

活动推荐

目录

以太坊智能合约OPCODE逆向之理论基础篇

storage

上面的stack和mem都是在EVM执行OPCODE的时候初始化，但是storage是存在于区块链当中，我们可以类比为计算机的存储磁盘。

所以，就算不执行智能合约，我们也能获取智能合约storage中的数据:

```
eth.getStorageAt(合约地址, slot)
# 第3个函数还有第三个参数，默认为"latest"，还可以设置为"earliest"或者"per
```

storage用来存储智能合约中所有的全局变量

使用 `SLOAD` 和 `SSTORE` 进行操作

`SSTORE(arg0, arg1)` 从栈中获取两个参数，表示
`eth.getStorageAt(合约地址, arg0) = arg1`

`SLOAD(arg0)` 从栈中获取一个参数，表示
`PUSH32(eth.getStorageAt(合约地址, arg0))`

变量

智能合约的变量从作用域可以分为三种, 全局公有变量(public), 全局私有变量(private), 局部变量

全局变量和局部变量的区别是，全局变量储存在storage中，而局部变量是被编译进OPCODE中，在运行时，被放在stack中，等待后续使用

公有变量和私有变量的区别是，公有变量会被编译成一个constant函数，后面会分析函数之前的区别

因为私有变量也是储存在storage中，而storage是存在于区块链当中，所以相当于私有变量也是公开的，所以不要想着用私有变量来储存啥不能公开的数据。

全局变量的储存模型

不同类型的变量在storage中储存的方式也是有区别的，下面对各种类型的变量的储存模型进行分析

1. 定长变量

第一种我们归类为定长变量，所谓的定长变量，也就是该变量在定义的时候，其长度就已经被限制住了

比如定长整型(int/uint.....), 地址(address), 定长浮点型(fixed/ufixed.....), 定长字节数组(bytes1–32)

这类的变量在storage中都是按顺序储存

```
uint a;       // slot = 0
address b;    // 1
ufixed c;     // 2
bytes32 d;    // 3
##
a == eth.getStorageAt(contract, 0)
d == eth.getStorageAt(contract, 3)
```

上面举的例子，除了 `address` 的长度是160bits，其他变量的长度都是256bits，而storage是256bits对齐的，所以都是一个变量占着一块storage，但是会存在连续两个变量的长度不足256bits的情况

```
address a;      // slot = 0
uint8 b;        // 0
address c;      // 1
uint16 d;       // 1
```

在opcode层面，获取a的值得操作是:

`SLOAD(0) & 0xffffffffffffffffffffffffffffffffffffffff`

Seebug漏洞平台
301 篇文章

以太坊智能合约OPCODE逆向之理论基础篇

获取d值得操作是:

```
SLOAD(1) // 0x1000000000000000000000000000000000000000 &
0xffff
```

因为 长度+a的长度不足256bits，变量a和b是连续的，所以他们在同一块storage中，[illegible]在编译的过程中进行区分变量a和变量b，但是后续在加上变量c，长度就超过了2[illegible] ts，因此把变量c放到下一块storage中，然后变量d跟在c之后

从上面我们可以看出，storage的储存策略一个是256bits对齐，一个是顺序储存。(并没[illegible]到充分利用每一字节的储存空间，我觉得可以考虑把d变量放到b变量之后)

2. 映射变量

```
m[illegible]ing(address => uint) a;
```

映射变量就没办法想上面的定长变量按顺序储存了，因为这是一个键值对变量，EVM采用的机制是:

```
SLOAD(sha3(key.rjust(64, "0")+slot.rjust(64, "0")))
```

比如: `a["0xd25ed029c093e56bc8911a07c46545000cbf37c6"]` 首先计算sha3哈希值:

```
>>> from sha3 import keccak_256
>>> data = "d25ed029c093e56bc8911a07c46545000cbf37c6".rjust(64
>>> data += "00".rjust(64, "0")
>>> keccak_256(data.encode()).hexdigest()
'739cc24910ff41b372fbcb2294933bdc3108bd86ffd915d64d569c68a8512
#
a["0xd25ed029c093e56bc8911a07c46545000cbf37c6"] == SLOAD("739c
```

我们也可以使用以太坊客户端直接获取:

```
> eth.getStorageAt(合约地址, "739cc24910ff41b372fbcb2294933bdc3
```

还有slot需要注意一下:

以太坊智能合约 OPCODE 逆向之调试器篇

作者：Hcamael@知道创宇404区块链安全研究团队 时间：
201 9/04

Seebug漏洞平台

以太坊智能合约开发第二篇：理解以太坊相关概念

Marser

以太坊合约审计 CheckList 之变量覆盖问题

2018年11月6日，DVP上线了一场“地球OL真实盗币游戏”，其中第二题是一道智能合约题目，题目中涉及到的了一个很有趣的问题，这里拿出来详细说说看。

Seebug漏洞平台

Solidity的Bytecode和Opcode简介

随着我们更深入地编写智能合约，我们将遇到诸如“ PUSH1”，“ SSTORE”，“ CALLVALUE”等术语。他们是什么，我们什么时候应该使用到他们?

程序那些事

@程序员，如何淋漓尽致地敲出Solidity安全...

区块链技术的发展要与安全挂钩，齐头并进，让迅速的发展约束在可靠的范围之内，才能真正让新科技稳步推广，深...

区块链大本营

以太坊 – 深入浅出虚拟机

创业有三种状态：投机，寻租以及自恋。屌丝程序员发现区块链是快速改变自己阶级命运的神器，一股脑冲进去，这...

Tiny熊

以太坊合约审计 CheckList 之“以太坊智能...

在知道创宇404区块链安全研究团队整理输出的《知道创宇以太坊合约审计CheckList》中，我们把超过10个问题点归...

Seebug漏洞平台

从EVM到Ewasm，硬核对比以太坊虚拟机.....

以太坊是一种内置图灵完备编程语言的区块链。任何人都可以利用以太坊的智能合约创造去中心化应用。

区块链大本营

从零构建以太坊（Ethereum）智能合约到项...

以太坊提供了便于交易的加密货币以太币（Ether），可透过智能合约解决交易上的信任问题，同时也可撰写DAPP来提...

Seebug漏洞平台
301 篇文章

以太坊智能合约OPCODE逆向之理论基础篇

转

尽管还有几天才结束，“The DAO”成为史上最大的众筹项目已是板上钉钉的事了，目前融资额已高达1.6亿美元。DAO...

Henry Zhang

3

0

更多文章