基于执行属性图的自动化智能合约动态分析 (WIP)

叶者
UC Berkeley

智能合约安全问题

BRIAN NEWAR DEC 0

DeFi disasters: $31M drained from MonoX and BadgerDAO losses top $120M

A disappointing week of exploits has put a temporary grim cloud over th BadgerDAO and Huobi-listed MonoX suffering major losses.

Crypto

Hackers abuse 'chaotic' Nomad exploit to drain almost $200M in crypto

Carly Page @carlypage_ / 2:03 PM GMT+2 • August 2, 2022 Comment

MOTHERBOARD
TECH BY VICE

Decentralized Crypto Exchange Offline After Hacker Steals $113M

The developers of the Maiar exchange took it offline and claim to have recovered the funds.

NEWS › CRYPTOCURRENCY NEWS

Crypto Worth Over $320 Million Taken in Wormhole Hack

Popular bridge linking Ethereum and Solana later retrieved the stolen assets

By MARK KOLAKOWSKI Published February 03, 2022

TECH

$100 million worth of crypto has been stolen in another major hack

PUBLISHED FRI, JUN 24 2022·6:38 AM EDT | UPDATED FRI, JUN 24 2022·9:28 AM EDT

Ryan Browne
@RYAN_BROWNE_

SHARE

智能合约常见安全审查流程

- 静态分析工具
- 手动安全审查

智能合约常见安全审查流程:静态分析工具

- 通过分析源代码检查程序性漏洞
- 常见工具包括:Slither、Oyente
- 常见漏洞:整数溢出、时间戳依赖
- 无法覆盖复杂的漏洞 [1]
  - 难以针对不同的dapp调整检测规则
  - 难以检测与合约状态相关的漏洞
  - 往往局限于单个智能合约分析

[1] Zhou L, Xiong X, Ernstberger J, et al. Sok: Decentralized finance (defi) attacks[J]. Cryptology ePrint Archive, 2022.

智能合约常见安全审查流程:手动安全审查

- 通过人工阅读源代码分析语义漏洞
- 依赖于审查人员的经验
- 周期长、开销大

MonoX 攻击

- MonoX (AMM) 在经过3次安全审查后仍然存在漏洞
- 损失高达三千万美元(2021年11月)
- 事故分析:
  - removeLiquidity 函数没有合适的权限控制
  - 攻击者可以操纵 MONO token 的价格

Audit Time	Audit Report	Auditor
May 3rd-15th, 2021	Halborn Audit Report	**Halborn**
Sep 6th-19th, 2021	Halborn Audit Report	**Halborn**
May 17th- June 8th, 2021	PeckShield Audit Report For Swap PeckShield Audit Report For Staking	**PeckShield**

事故分析（Post-mortem Analysis）

- 智能合约应用与协议十分复杂
- 现有的智能合约安全审查流程无法杜绝攻击的发生
- 攻击后，及时且有效的事故分析至关重要
    - 不同链上的相同或类似应用可能存在类似的漏洞
- 一个好的自动化事故分析工具可以极大地帮助入侵检测

常见的事故分析流程

- 通过区块链浏览器等工具获取攻击向量与粗略影响
  - Etherscan
- 通过程序追踪获取攻击执行细节
  - Openchain transaction tracer
- 通过自动化动态分析工具检查特定攻击
  - Sereum[1]:基于动态分析的重入攻击检测

[1] Rodler M, Li W, Karame G O, et al. Sereum: Protecting existing smart contracts against re-entrancy attacks[J]. arXiv preprint arXiv:1812.05934, 2018.

概况 | 内部交易 | 事件日志 (296) | 状况变化 | More

交易哈希:	0x9f14d093a2349de08f02fc0fb018dadb449351d0cdb7d0738ff69cc6fef5f299
状态:	成功
区块:	13715026 3418891 区块确认
时间戳:	512 天 10 小时前 (Nov-30-2021 01:27:51 PM +UTC)
Transaction Action:	Swap 4,029,106.880396 USDC For 847.206697433507365949 Ether On Uniswap V2 Swap 4,525,120.098829 USDT For 949.032579850826052421 Ether On Uniswap V2 Burn 11135368878676001 of ERC-1155 Burn 1019554304073565 of ERC-1155 Burn 666091399331238500 of ERC-1155 Mint 1027 of ERC-1155
Sponsored:	
发送方:	0xEcbE385F78041895c311070F344b55BfAa953258 (MonoX Finance Exploiter)
Interacted With (接收方):	0xf079d7911c13369E7fd85607970036D2883aFcfD 转移 0.1 以太币 发送方 0xf079d7...883aFcfD 接收方 Wrapped Ether
ERC-20 代币转移: 145	发送方 0xf079d7...883aFcfD 接收方 0x59653E...2B5ab2f4 For 0.1 $189.05 Wrapped Ethe... (WETH...) 发送方 0x59653E...2B5ab2f4 接收方 0xf079d7...883aFcfD For 79.98609431154262101 MonoX Token... (MONO...) 发送方 Null: 0x000...000 接收方 0x7B9aa6...08293fFC For 6.86217198681223029 Virtual Cash... (vCASH...) 发送方 0x59653E...2B5ab2f4 接收方 0x7B9aa6...08293fFC For 1,670.757229764922279713 MonoX Token... (MONO...) 发送方 Null: 0x000...000 接收方 0x81D98c...9FA4cf38 For 0.628300423692773565 Virtual Cash... (vCASH...) 发送方 0x59653E...2B5ab2f4 接收方 0x81D98c...9FA4cf38 For 152.974521385715493913 MonoX Token... (MONO...) 发送方 Null: 0x000...000 接收方 0xab5167...d1837355 For 410.478879590637971405 Virtual Cash... (vCASH...) 发送方 0x59653E...2B5ab2f4 接收方 0xab5167...d1837355 For 99,940.741365832694596828 MonoX Token... (MONO...) 发送方 0xf079d7...883aFcfD 接收方 0x59653E...2B5ab2f4 For 0.000000000196875656 MonoX Token... (MONO...) Scroll for more

Transaction Info

Hash: 0x9f14d093a2349de08f02fc0fb018dadb449351d0cdb7d0738ff69cc6fef5f299

Status: Succeeded Timestamp: 2021-11-30 05:27:51 PST (1 year, 4 months ago) Block: 13715026

From: [] 0xEcbE385F78041895c311070F344b55BfAa953258 To: [] 0xf079d7911c13369E7fd85607970036D2883aFcfD

Value: 0.1 ETH (460.6438 USD) Transaction Fee: 0.472845680129136708 ETH (2,178.1344 USD)

Gas Used: 4684892/5856169 (79%) Gas Price: 100.929899799 gwei Max Priority Fee: 9.678223095 gwei Max Fee: 102.659711949 gwei

Nonce: 2 Type: EIP-1559 Index: 12

Value Changes

	Address	Change In Value
[⌄]	0x8f6A86f3aB015F4D03DDB13AbB02710e6d7aB31B	11,837,643.5772 USD
[⌄]	0xab5167e8cC36A3a91Fd2d75C6147140cd1837355	563,946.8275 USD
[⌄]	[Uniswap V2: USDT]	164,573.7210 USD
[⌄]	[Uniswap V2: USDC]	133,199.4298 USD
[⌄]	0x7B9aa6ED8B514C86bA819B99897b69b608293fFC	9,427.7488 USD
[⌄]	0x81D98c8fdA0410ee3e9D7586cB949cD19FA4cf38	863.2046 USD
[⌄]	[Wrapped Ether]	460.6438 USD
[⌄]	0x0000000000000000000000000000000000000000	0.0000 USD
[⌄]	0xf079d7911c13369E7fd85607970036D2883aFcfD	-167.7056 USD
[⌄]	0xEcbE385F78041895c311070F344b55BfAa953258	-460.6438 USD
[⌄]	[TransparentUpgradeableProxy]	-12,709,486.8035 USD

Decoded Actions

```
[swap] exchange=uniswap-v2, tokenIn=4029106880396 [Centre: USD Coin], amountOut=847206697433507365949 [Wrapped Ether], recipient=0xf079d7911c13369E7fd85607970036D2883aFcfD, actor=
[swap] exchange=uniswap-v2, tokenIn=4525120098829 [Tether: USDT Stablecoin], amountOut=949032579850826052421 [Wrapped Ether], recipient=0xf079d7911c13369E7fd85607970036D2883aFcfD,
```

Call Trace

```
⌄ [call][5826774][] 0xf079d7911c13369E7fd85607970036D2883aFcfD.guessed_a70cfccf[0.1 ETH]([MonoToken], [0x7B9aa6ED8B514C86bA819B99897b69b608293fFC, 0x81D98c8fdA0410ee3e9D7586cB949c
  > [call][23974][] [Wrapped Ether].deposit[0.1 ETH]() → ()
  > [call][24420][] [Wrapped Ether].approve(guy=[TransparentUpgradeableProxy], wad=100000000000000000) → (true)
  > [call][173613][] [TransparentUpgradeableProxy].swapExactTokenForToken(tokenIn=[Wrapped Ether], tokenOut=[MonoToken], amountIn=100000000000000000, amountOutMin=1, to=0xf079d79
  > [call][24682][] [MonoToken].approve(spender=[TransparentUpgradeableProxy], amount=115792089237316195423570985008687907853269984665640564039457584007913129639935) → (true)
  > [staticcall][8434][] [TransparentUpgradeableProxy].pools([MonoToken]) → (pid=10, lastPoolValue=5310574652057472396O5262, token=[MonoToken], status=2, vcashDebt=0, vcashCredit=
  > [staticcall][3501][] [TransparentUpgradeableProxy].totalSupplyOf(pid=10) → (678246322513988066)
  > [staticcall][3384][] [TransparentUpgradeableProxy].balanceOf(account=0x7B9aa6ED8B514C86bA819B99897b69b608293fFC, id=10) → (11135368878676001)
  > [call][134469][] [TransparentUpgradeableProxy].removeLiquidity(_token=[MonoToken], liquidity=11135368878676001, to=0x7B9aa6ED8B514C86bA819B99897b69b608293fFC, minVcashOut=0, 
  > [staticcall][3384][] [TransparentUpgradeableProxy].balanceOf(account=0x81D98c8fdA0410ee3e9D7586cB949cD19FA4cf38, id=10) → (1019554304073565)
  > [call][113569][] [TransparentUpgradeableProxy].removeLiquidity(_token=[MonoToken], liquidity=1019554304073565, to=0x81D98c8fdA0410ee3e9D7586cB949cD19FA4cf38, minVcashOut=0, m
  > [staticcall][3384][] [TransparentUpgradeableProxy].balanceOf(account=0xab5167e8cC36A3a91Fd2d75C6147140cd1837355, id=10) → (666091399331238500)
  > [call][79369][] [TransparentUpgradeableProxy].removeLiquidity(_token=[MonoToken], liquidity=666091399331238500, to=0xab5167e8cC36A3a91Fd2d75C6147140cd1837355, minVcashOut=0, 
  > [staticcall][1501][] [TransparentUpgradeableProxy].totalSupplyOf(pid=10) → (0)
  > [call][125329][] [TransparentUpgradeableProxy].addLiquidity(_token=[MonoToken], _amount=196875656, to=0xf079d7911c13369E7fd85607970036D2883aFcfD) → (liquidity=927)
  > [staticcall][2434][] [TransparentUpgradeableProxy].pools([MonoToken]) → (pid=10, lastPoolValue=1027394637, token=[MonoToken], status=2, vcashDebt=0, vcashCredit=0, tokenBalan
    [call][629][] [MonoToken].balanceOf(account=0xf079d7911c13369E7fd85607970036D2883aFcfD) → (79986094311345745354)
  > [call][63618][] [TransparentUpgradeableProxy].swapExactTokenForToken(tokenIn=[MonoToken], tokenOut=[MonoToken], amountIn=196875655, amountOutMin=0, to=0xf079d7911c13369E7fd85
  > [staticcall][2434][] [TransparentUpgradeableProxy].pools([MonoToken]) → (pid=10, lastPoolValue=1027394637, token=[MonoToken], status=2, vcashDebt=0, vcashCredit=0, tokenBalan
    [call][629][] [MonoToken].balanceOf(account=0xf079d7911c13369E7fd85607970036D2883aFcfD) → (79986094311228576145)
  > [call][63618][] [TransparentUpgradeableProxy].swapExactTokenForToken(tokenIn=[MonoToken], tokenOut=[MonoToken], amountIn=314044864, amountOutMin=0, to=0xf079d7911c13369E7fd85
  > [staticcall][2434][] [TransparentUpgradeableProxy].pools([MonoToken]) → (pid=10, lastPoolValue=1027394637, token=[MonoToken], status=2, vcashDebt=0, vcashCredit=0, tokenBalan
    [call][629][] [MonoToken].balanceOf(account=0xf079d7911c13369E7fd85607970036D2883aFcfD) → (79986094311041674477)
  > [call][63618][] [TransparentUpgradeableProxy].swapExactTokenForToken(tokenIn=[MonoToken], tokenOut=[MonoToken], amountIn=500946532, amountOutMin=0, to=0xf079d7911c13369E7fd85
  > [staticcall][2434][] [TransparentUpgradeableProxy].pools([MonoToken]) → (pid=10, lastPoolValue=1027394637, token=[MonoToken], status=2, vcashDebt=0, vcashCredit=0, tokenBalan
    [call][629][] [MonoToken].balanceOf(account=0xf079d7911c13369E7fd85607970036D2883aFcfD) → (79986094310743539551)
  > [call][63618][] [TransparentUpgradeableProxy].swapExactTokenForToken(tokenIn=[MonoToken], tokenOut=[MonoToken], amountIn=799081458, amountOutMin=0, to=0xf079d7911c13369E7fd85
  > [staticcall][2434][] [TransparentUpgradeableProxy].pools([MonoToken]) → (pid=10, lastPoolValue=1027394637, token=[MonoToken], status=2, vcashDebt=0, vcashCredit=0, tokenBalan
    [call][629][] [MonoToken].balanceOf(account=0xf079d7911c13369E7fd85607970036D2883aFcfD) → (79986094310267971649)
  > [call][63618][] [TransparentUpgradeableProxy].swapExactTokenForToken(tokenIn=[MonoToken], tokenOut=[MonoToken], amountIn=1274649360, amountOutMin=0, to=0xf079d7911c13369E7fd85
```

现有的自动化事故分析流程问题

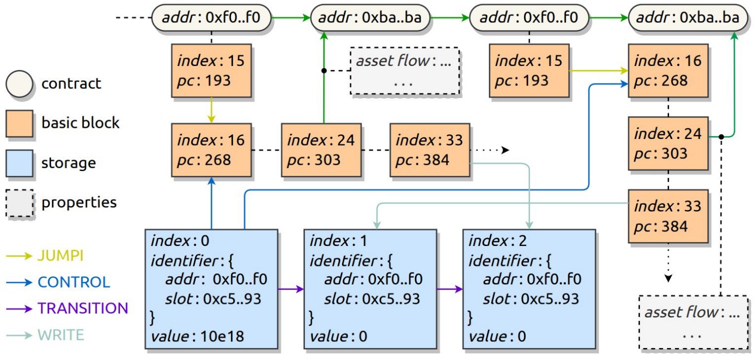
- 攻击向量往往很复杂或难以分析
  - 攻击者合约不开源
  - 字节码不可读
  - 反编译器难以用于大型合 约
- 工具局限于粗略或非结构化的信息
  - Token transfer 等信息往往无法准确描述攻 击的全貌
  - Program trace 往往十分庞大
    - MonoX 攻击交易一共执行 596,302 个字节码
- 工具局限于检测特定的漏洞
  - 难以针对不同dapp和不同攻击类型定制检测

问题陈述

- 能否建立一种比 program trace 更好的方式来表示智能合约的执行？
- 能否建立一种统一的语言用于表述不同的智能合约分析需求？
- 能否构建一个更具有拓展性的智能合约动态分析框架？

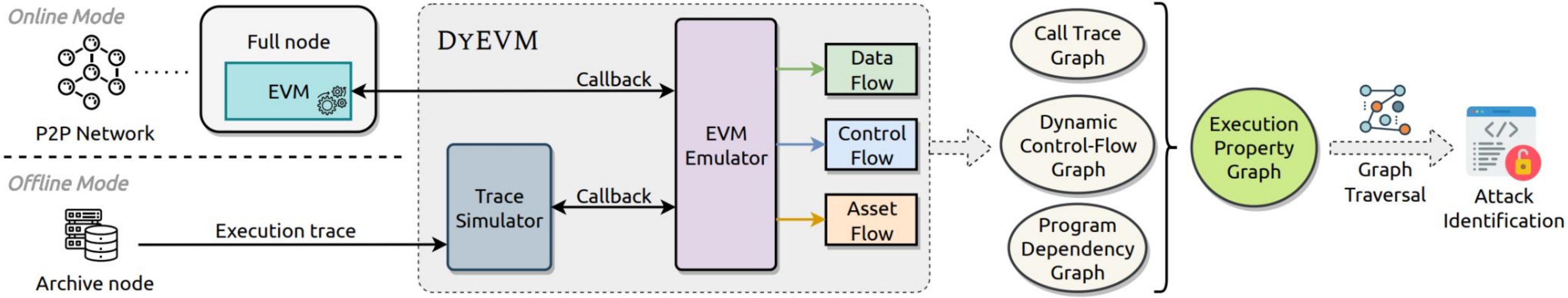
解决方案概览：执行属性图（Execution Property Graph）

- 将智能合约的执行信息以属性图的方式表达
- 执行属性图综合以下的信息：
  - 函数调用信息以及资产交易流
  - 动态控制流
  - 动态数据流
- 将所需的智能合约分析规则以图遍历的方式表达



Clue：基于 EPG 的智能合约自动化动态分析框架

- 将 program trace 转换为 EPG
  - 通过 program trace 模拟执行 EVM
- 离线模式用于事故分析
- 在线模式用于入侵检测

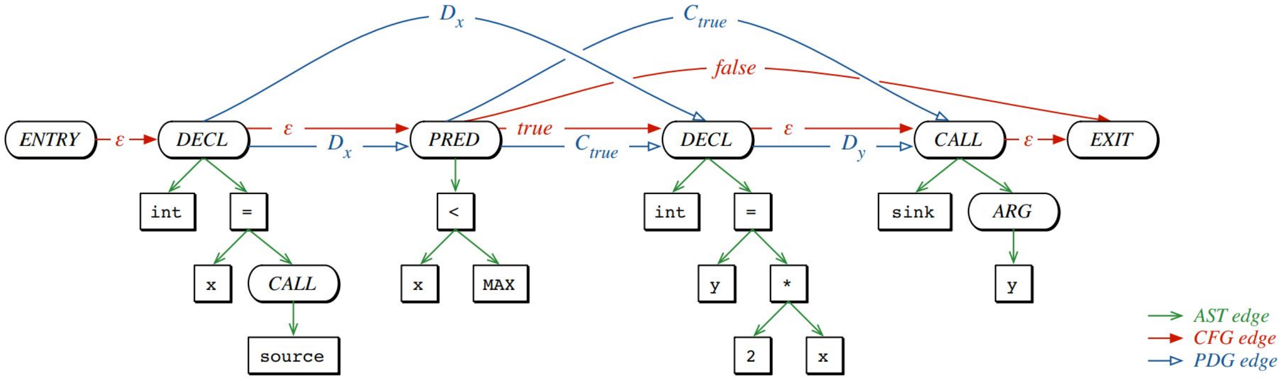


背景：以太坊虚拟机（EVM）

- 栈式虚拟机，执行字节码
- 字节码由源代码（solidity，viper）编译而来
- 两种账户：
  - EOA：用户控制
  - 智能合约：代码存于区块链上，不可修改
- 智能合约拥有持久化的状态（key-value mapping）
  - 可以在运行时修改

背景:代码属性图(Code Property Graph [1])

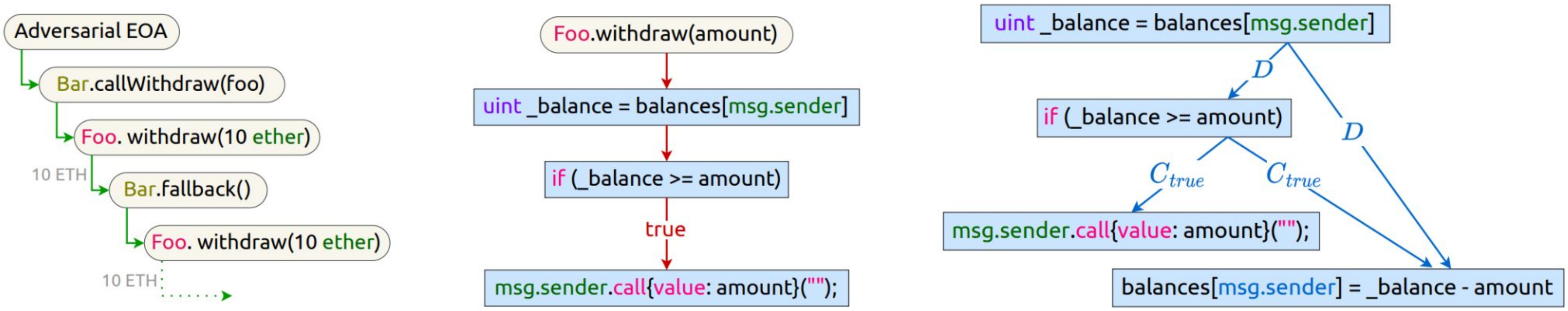
- 一种静态分析工具
- 综合以下信息:
  - 抽象代码树(AST)
  - 控制流程图(CFG)
  - 程序依赖图(PDG)



[1] Yamaguchi F, Golde N, Arp D, et al. Modeling and discovering vulnerabilities with code property graphs[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 590-604.

智能合约执行属性图（Execution Property Graph）

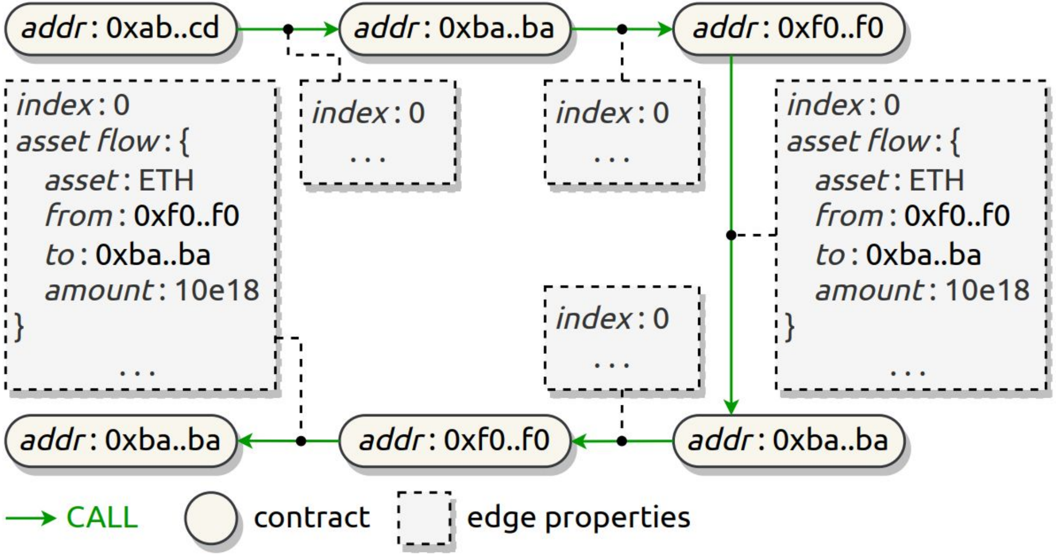
- 合约调用信息以及资产交易流
  - 捕捉合约交互层面的信息
- 动态控制流
  - 捕捉单函数内的控制流程依 赖
- 动态数据流
  - 捕捉合约执行过程中的数据依 赖



(a) Call Trace Graph　(b) Dynamic Control Flow Graph　(c) Dynamic Dependence Graph

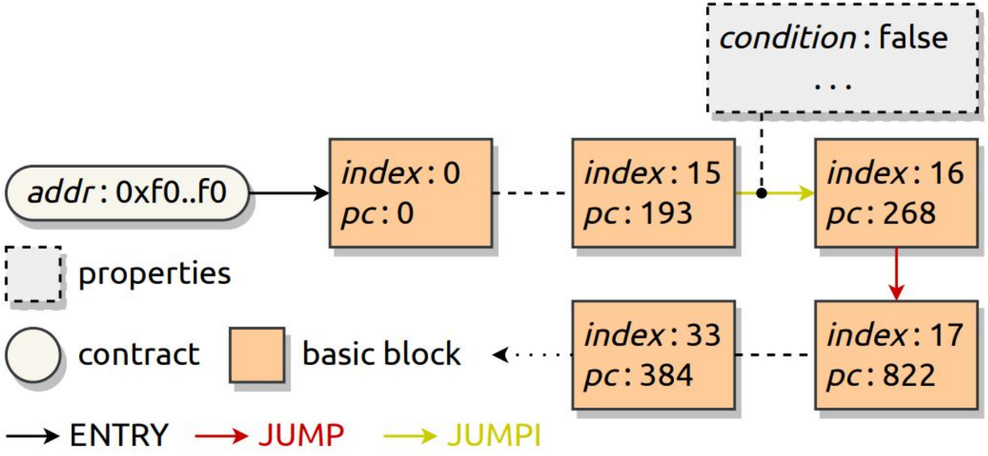
函数调用图

- 合约调用顺序与调用依赖
- 合约调用间的资产转移



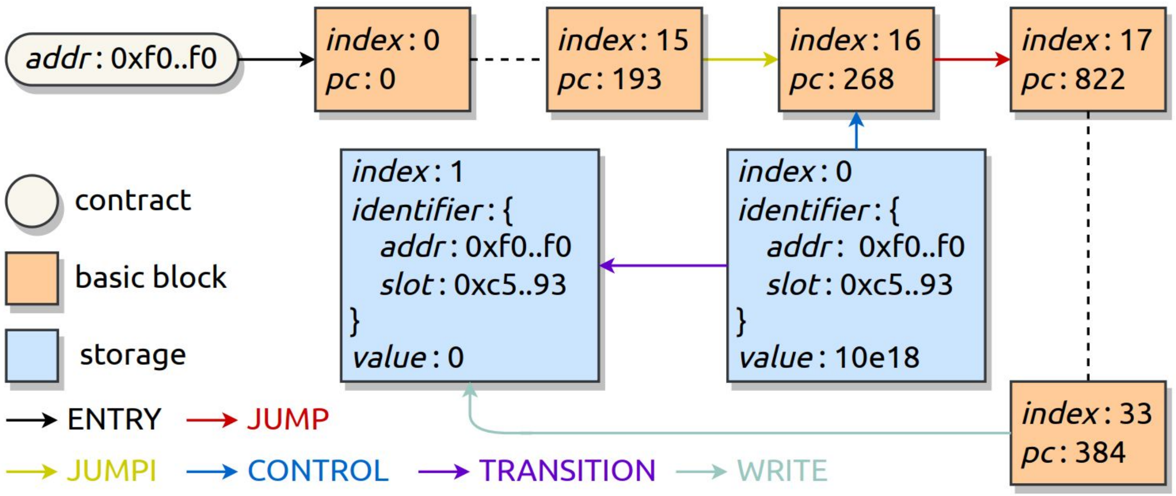
动态控制流程图

- 合约内单个函数的(动态)代码执行流程
- 代码块之间的跳转条件



动态数据流图

- 数据源
  - 状态（storage）
  - 输入/输出
  - 调用信息
  - …
- 数据依赖
  - 状态读/写
  - 数据源依赖
  - 控制依赖

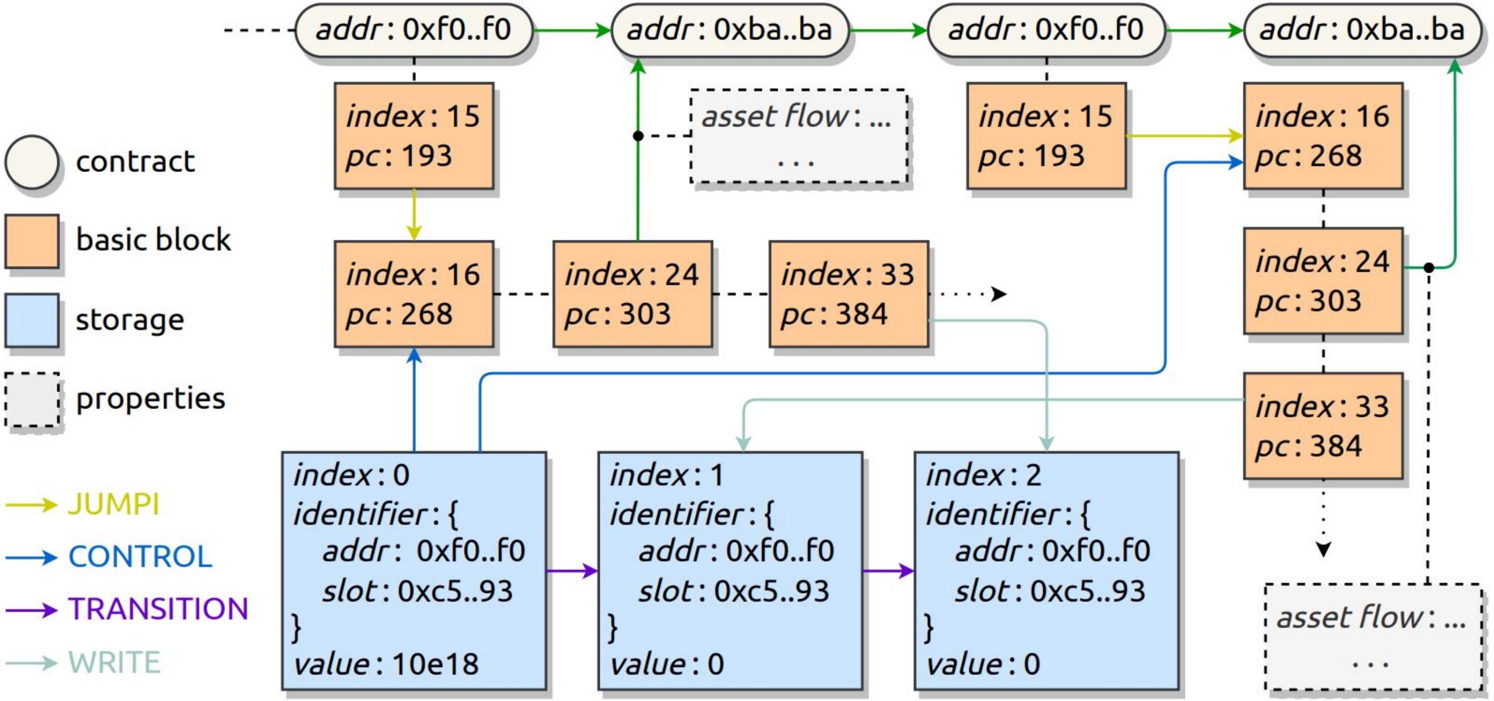


例子:一个简单的重入攻击

- 行9存在外部调用
- 行10存在状态更新
- 攻击流程:
  - Bar 调用 callWithdraw 进入 Foo
  - Foo 在行10回调 Bar
  - Bar 通过 fallback 重入 Foo

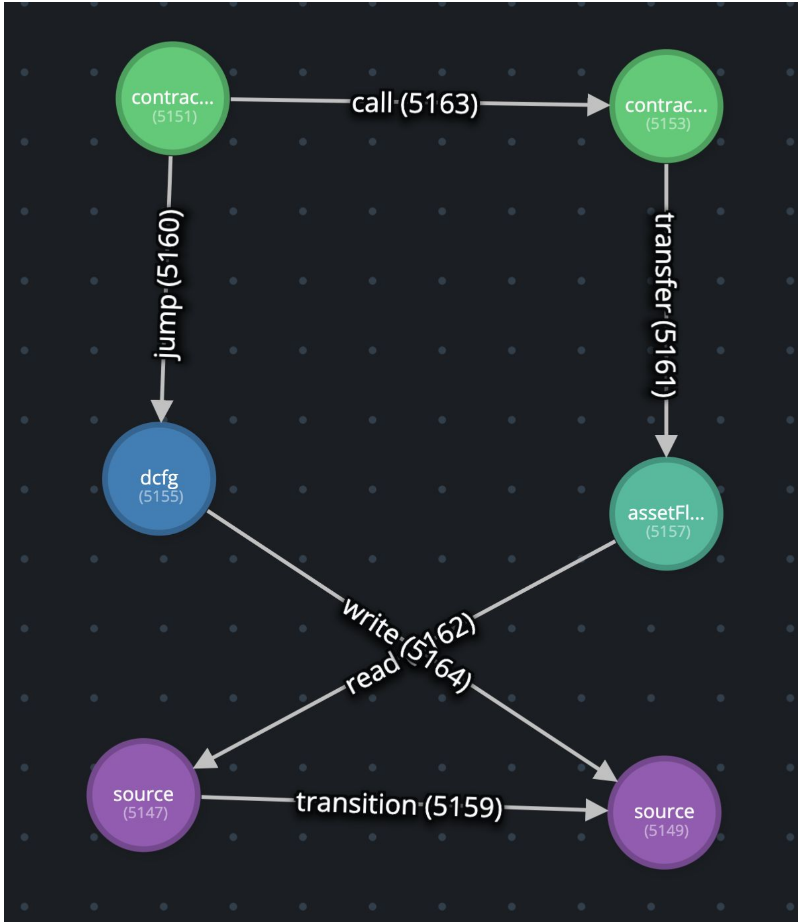
```
1  pragma solidity ^0.8.0;
2
3  contract Foo {
4    mapping (address => uint) public balances;
5
6    function withdraw (uint amt) public {
7      uint _balance = balances[msg.sender];
8      if (_balance >= amt) {
9        msg.sender.call{value: amt}("");
10       balances[msg.sender] = _balance - amt;
11     } else {
12       revert("insufficient balance");
13     }
14   }
15 }
16
17 contract Bar {
18   function callWithdraw(address foo) public {
19     Foo(foo).withdraw(10 ether);
20   }
21
22   fallback() external payable {
23     if (address(this).balance < 99999 ether) {
24       callWithdraw(msg.sender);
25     }
26   }
27 }
```

将三类图合并成执行属性图



重入攻击检测

- 特征1:重入内有资产转移
- 特征2:状态更新在重入之后
- 遍历规则:
  - 寻找所有子调用内有资产转移的重入
  - 检查资产转移所依赖的数据源
  - 检查数据源是否在父 调用内被修改



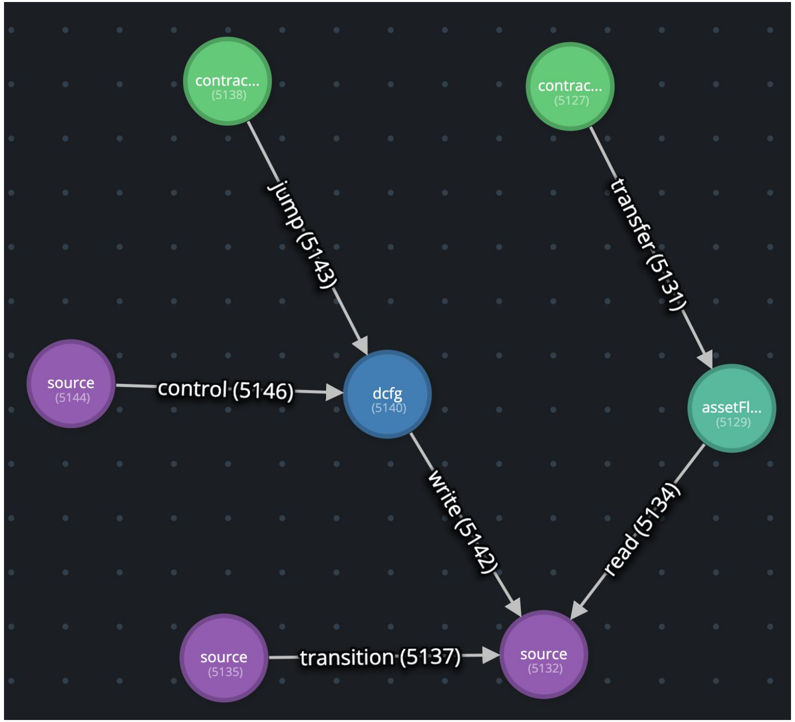
```
g.V().hasLabel('contractCall').as('victim').where(
    out('transfer'). // Get asset flows
    out('read').dedup(). // Get amount data sources
    emit().repeat(in('dependency')).until(inE('dependency').count().is(0)).dedup(). // Get all dependencies
    has('type', 'storage'). // Filter out storage data sources
    repeat(out('transition')).until(outE('transition').count().is(0)).emit(). // Get all histories onwards
    in('write').dedup(). // Get all basic blocks that modify the storage
    repeat(in('jump')).until(inE('jump').count().is(0)).emit().
    hasLabel('contractCall'). // Get the contract call of the basic block
    where(eq('victim')).by('address'). // Filter calls to have the same address
    repeat(out('call')).until(outE('call').count().is(0)).emit().
    where(eq('victim')).count().is(gt(0)) // Filter calls that have paths to the victim call
)
```

控制权限缺失攻击检测

- 资产转移是否发生应当受特殊数据源控制
- 该数据源应该包括调用者
  - ORIGIN
  - 第0层的CALLER
- 遍历:
  - 从资产转移出发
  - 寻找资产转移的基本代码块
  - 检查写入代码块的所有控制数据源

价格操纵攻击检测

- 资产转移的数量受到某个状态的控制
- 该状态可以被攻击者人为操纵
    - 状态写入的基本代码块没有适当的权限控制
- 遍历：
    - 从资产转移出发
    - 寻找控制数量的数据源
    - 检查数据源的写入代码块
    - 检查写入代码块的权限控制



实验结果

- 使用 SoK: Decentralized Finance Attack [1] 中的数据集
- 对照数据集为被攻击的合约相关的交易
  - 范围为2020年1月至2022年4月
- 重入攻击检测：

Dataset	*Attack*	*Gas*	*Random*	
# Transactions	87	1,077[1]	19,996[1]	
# FN/FP	7 (8.05%)	2 (0.19%)	1 (0.005%)	
FN/FP Type	FN-I	FP-I	FP-II	TP[2]
# FN/FP Per Type	8	2	1	11
Avg. Gas Cost	3,333,219.34	2,129,177.73	237,546.81	
Avg. Detection Time	107.63ms	19.72ms	6.95ms	

[1] Remove reentrancy attack transactions in *Random* and *Gas* datasets.

[2] True positives because these attacks are not labeled in the additionally collected transactions.

[1] Zhou L, Xiong X, Ernstberger J, et al. Sok: Decentralized finance (defi) attacks[J]. Cryptology ePrint Archive, 2022.

总结

- 执行属性图
    - 以属性图的形式表示智能合 约的动态执行信息
- 自动化动态分析
    - 以图遍历的形式表达所需的分析 规则
- Clue:基于执行属性图的自动化智能合约动态分析框架
- Q&A