HTTPS://CONFERENCE.HITB.ORG/HITBSECCONF2024BKK

Discovering and Investigating Propagated Vulnerabilities from Ethereum to Its Layer-2 Blockchains

Dr. Daoyuan Wu

Research Assistant Professor, HKUST

#HITB2024BKK

TRACK 1

30 AUG

Ethereum is the most popular blockchain for hosting smart contracts

Crowdfunded and Development begun in 2014.

The PoW (Proof-of-Work) network went live on 30 July 2015.

Switched to PoS (Proof-of-Stake) on 26 Feb 2023.

More than 2B (2,487,725,064 as of Aug 26) **transactions** sent.

More than 1M (1,344,143 as of Aug 26) **token contracts** created.

Ethereum's smart contract language, Solidity, appears in Top Programming Languages 2024.

Ethereum is also quite decentralized

Top 10 Countries

View All Nodes

Total **4,877** nodes found

#	Countries	Last 24 Hours	Last 24 Hours	Last 7 Days
1	United States	2,454 (49.78%)	▼ 45.98%	▼ 41.72%
2	Germany	662 (13.43%)	▲ 612.50%	▼ 46.19%
3	United Kingdom	155 (3.14%)	▲ 1100.00%	▼ 34.85%
4	Canada	145 (2.94%)	▲ 1700.00%	▼ 46.11%
5	France	138 (2.80%)	▲ 1200.00%	**▲ 10.24%**
6	South Korea	107 (2.17%)	▼ 99.07%	▼ 32.34%
7	Netherlands	101 (2.05%)	▲ 800.00%	▼ 28.27%
8	Iran	90 (1.83%)	▲ 300.00%	▼ 11.37%
9	Singapore	84 (1.70%)	▼ 96.43%	**▲ 5.42%**
10	Australia	65 (1.32%)	▼ 89.23%	▼ 15.07%

Node Stats

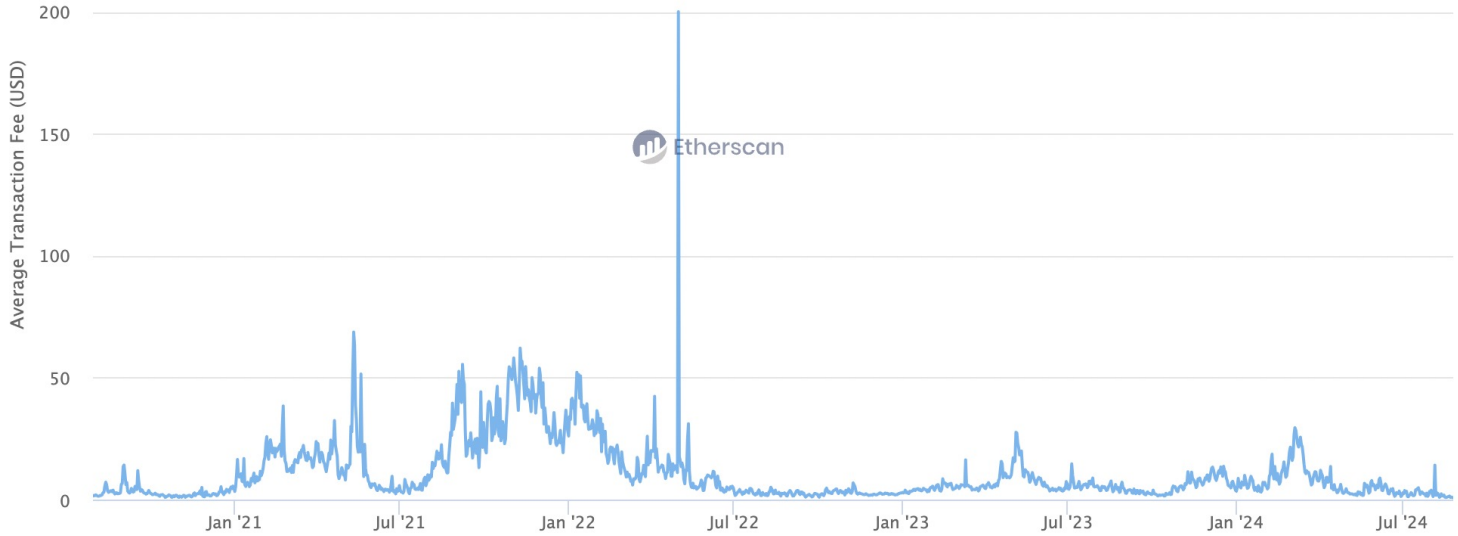
▲ 1.31%	▲ 12.80%	▼ 7.97%
Last 24 Hours	Last 7 Days	Last 30 Days

#HITB2024BKK

Ethereum suffers from low throughput and expensive transaction fees

Ethereum currently has **only 14.3 TPS** (transactions per second).

The transaction fee is now around **1USD** in the bear market but was high in the past.



#HITB2024BKK

Third-party layer-2 blockchain networks have emerged in recent years

#	NAME	RISKS	TYPE	STAGE	PURPOSE
1	Arbitrum One		Optimistic Rollup	STAGE 1	Universal
2	Base		Optimistic Rollup	STAGE 0	Universal
3	OP Mainnet		Optimistic Rollup	IN REVIEW	Universal
4	Blast		Optimistic Rollup	STAGE 0	Universal, DeFi
5	Scroll		ZK Rollup	STAGE 0	Universal
6	Mantle		Optimium	n/a	Universal
7	ZKsync Era		ZK Rollup	STAGE 0	Universal
8	Linea		ZK Rollup	STAGE 0	Universal
9	Starknet		ZK Rollup	STAGE 0	Universal

BINANCE SMART CHAIN

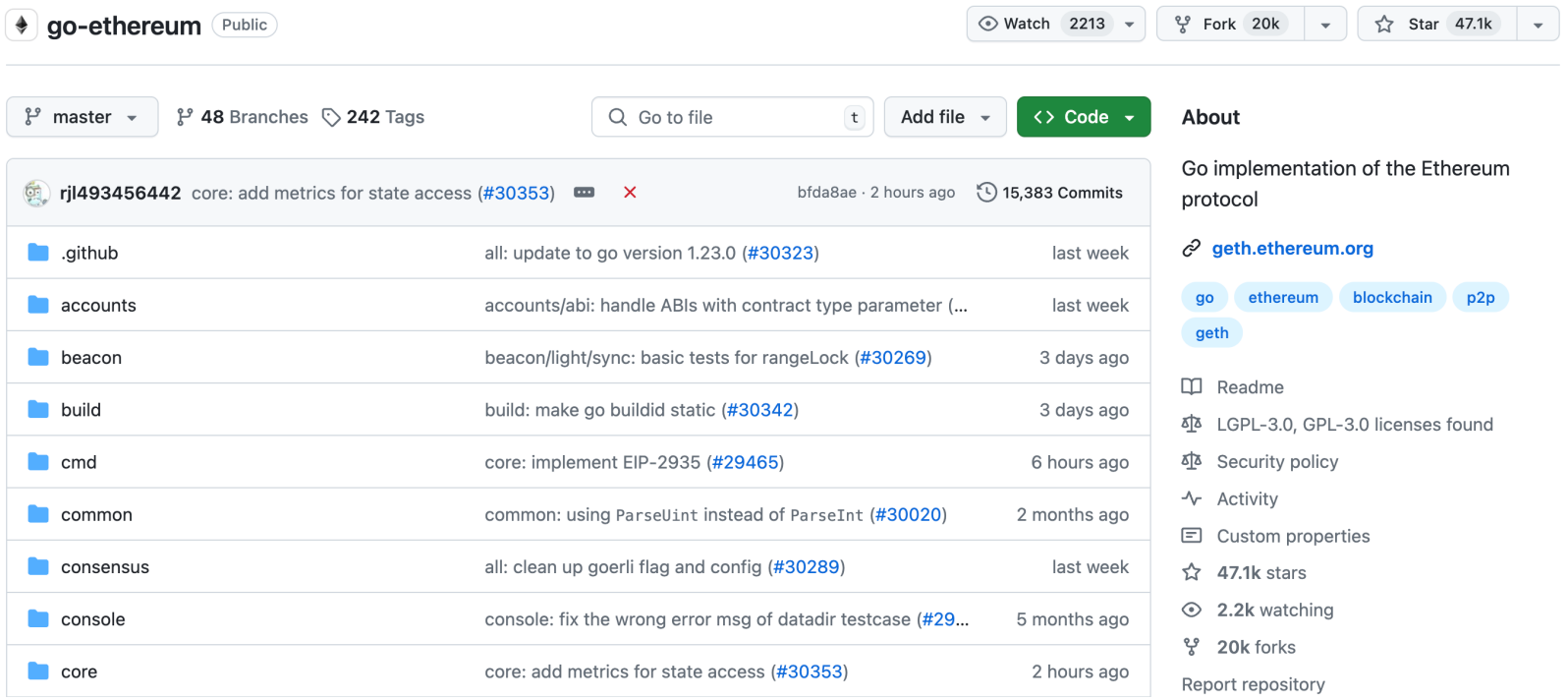
polygon

AVALANCHE

fantom

#HITB2024BKK

They copy & customize Ethereum



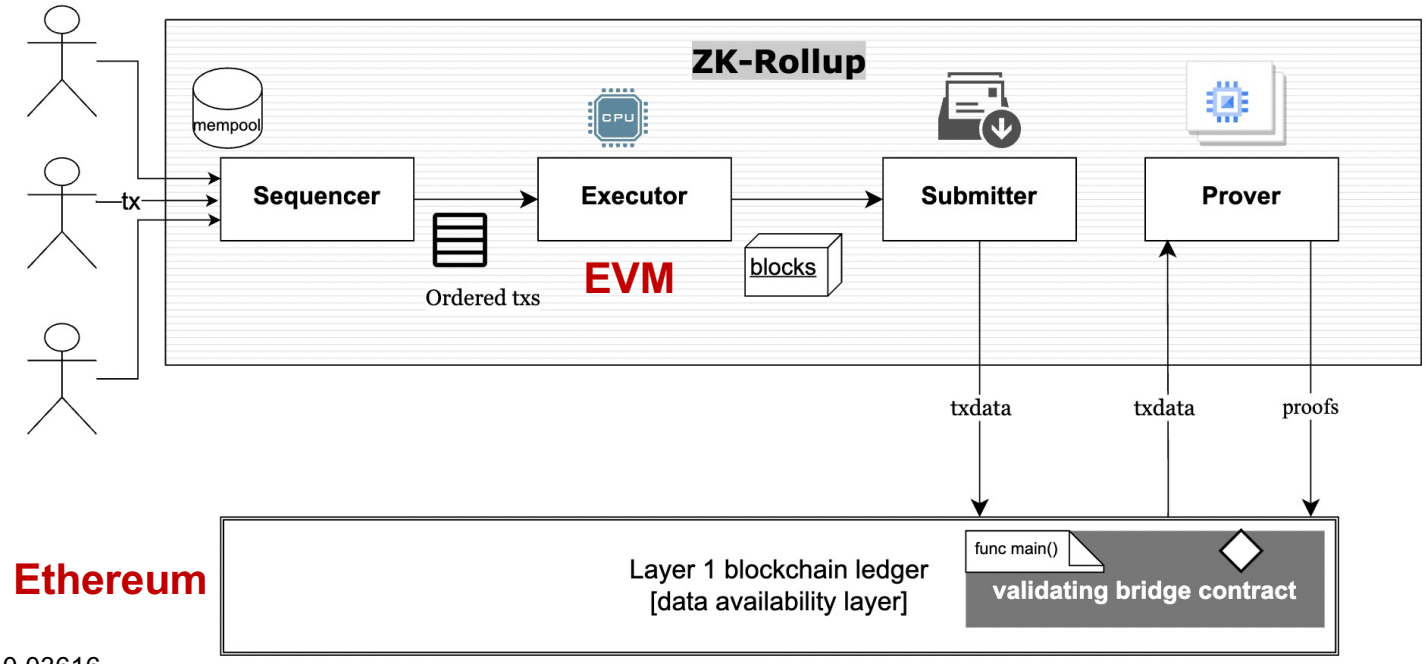
#HITB2024BKK

They copy & customize Ethereum

(b) Ethereum and its forked projects (as of 31 August 2023).

#	Name	Code	Market Cap	Repository	Star
2	Ethereum	ETH	$229.87B	ethereum/go-ethereum	37.7K
5	Binance	BNB	$50.69B	bnb-chain/bsc	1.6K
14	Avalanche	AVAX	$7.65B	ava-labs/subnet-evm	1.6K
17	Polygon	MATIC	$5.15B	maticnetwork/bor	400
78	Celo	CELO	$604.02M	celo-org/celo-blockchain	382
199	Optimism	OP	$263.36M	ethereum-optimism/op-geth	1.2K
-	Kcc	-	-	kcc-community/kcc	43
-	Heco	-	-	stars-labs/heco-chain	250
-	Hoo	-	-	hoosmartchain/hoo-smartchain	12
-	BitTorrent	-	-	bttcprotocol/bttc	28

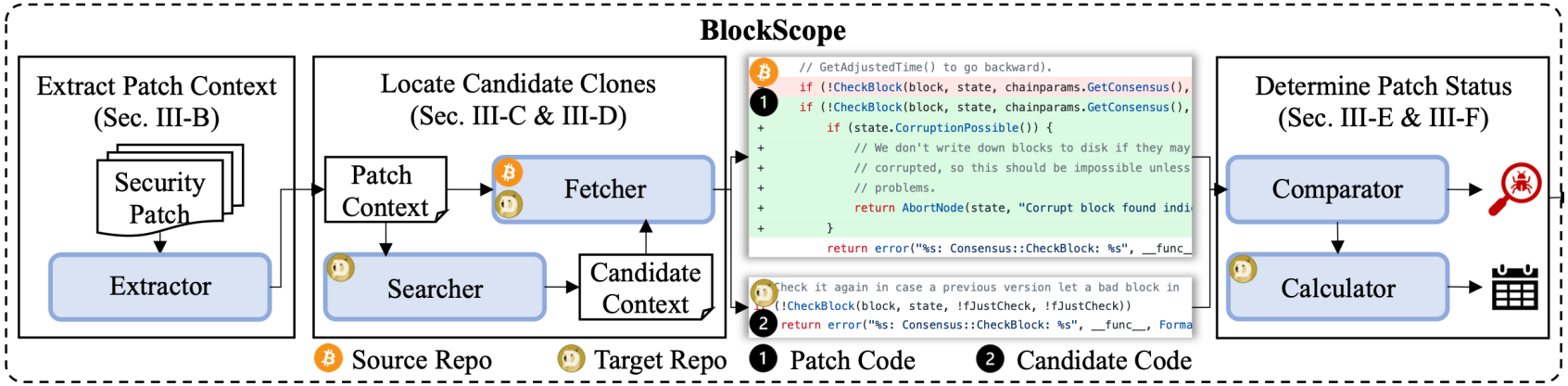
The architecture between Ethereum and its layer-2 blockchain networks



Source: arXiv:2310.03616

Our Tool: BlockScope

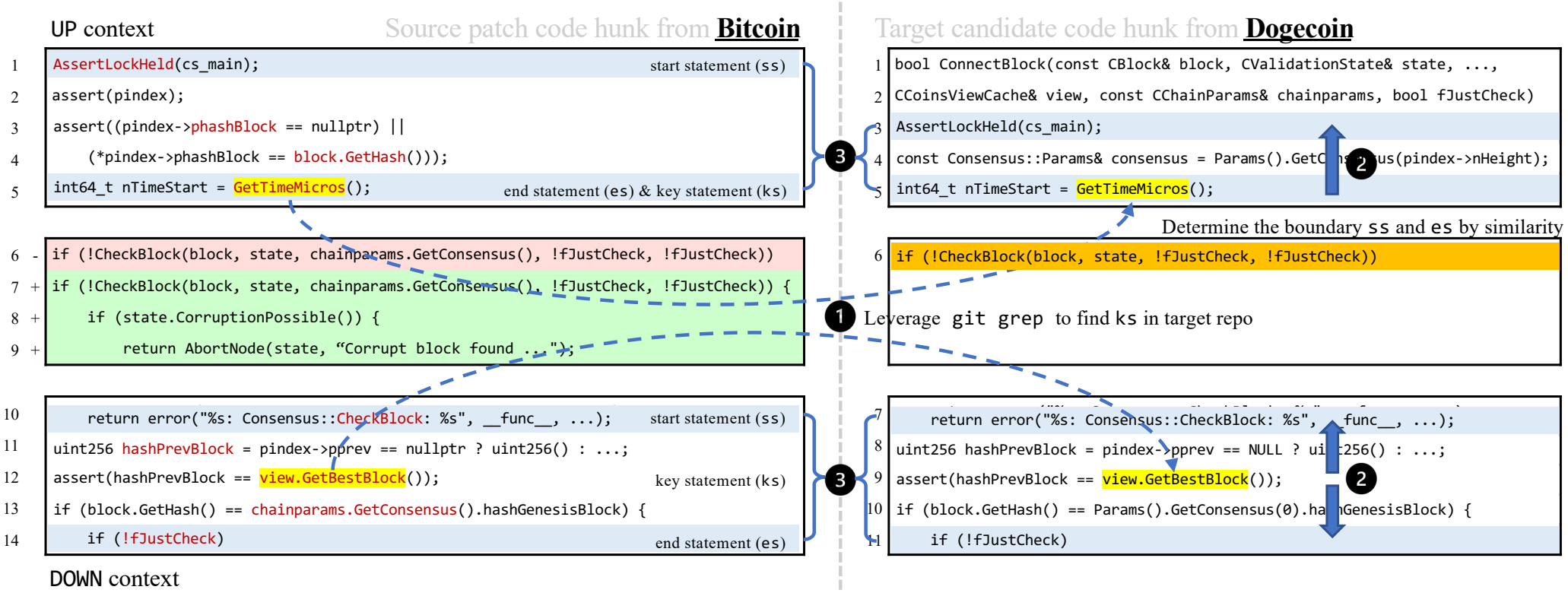
A novel patch-based clone detection tool for propagated vulnerabilities in forked blockchain projects.



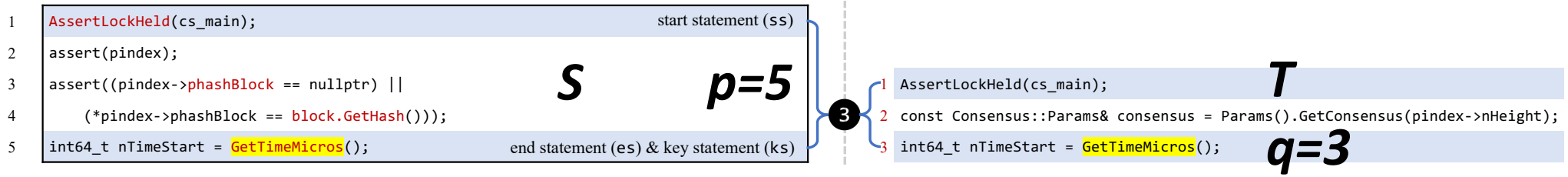
1. Leverage patch code contexts to locate only potentially relevant code

2. Adopt similarity-based code match for being immune to clone variants

Context-based Candidate Clone Search



A New Way of Calculating Code Similarity



- Source code ***S*** with ***p*** statements and target code ***T*** with ***q*** statements.
- 1. Pair-up each statement in *S* with the most similar statement in *T*, i.e.,
  - $\forall i \in [1,p]$, find j, s.t., $j = \underset{1\le k\le q}{\operatorname{argmax}} \operatorname{strsim}(S_i, T_k)$.

$p \neq q$ issue

- 2. Multiply $\operatorname{strsim}(S_i, T_j)$ by a **reward factor** $r \in [0,1]$, i.e., $\operatorname{strsim}(S_i, T_j) r^{|i-j|}$:
  - $r^{|i-j|}$ indicates: the greater $|i-j|$ the smaller the similarity between S_i and T_j.

code ordering issue

- 3. Add up all the weighted similarities and normalize into [0,1], i.e.,
  - $\text{SIMILARITY}(S,T) = \frac{1}{p}\sum_{i=1}^{p} \operatorname{strsim}(S_i, T_j) r^{|i-j|}$.

BlockScope vs. State-of-the-art Tools

Patch-based code clone detection

ReDeBug [SP'12] → **Language Dependent** → VUDDY [SP'17] → MVP [Usenix'20] → VGraph [EuroSP'20]

Hash tokenized contexts
Cannot detect Type-2

Add variable abstraction
Cannot detect Type-3

More "program analysis"

Hash-based "exact" code matching for the basic unit

```
1  void foo() {
2      int x = intput();
3      if (x > MIN) {
4          int y = x * 10;
5          output(y);
6      }
7  }
```

Original Vuln Code

BlockScope

```
1  void foo() {
2    int x = intput();
3    if (x > minimum) {
4      int y = x * 10;
5      output(y);
6    }
7  }
```

Type-2 Clone

Similarity-based code match (to cover more vulns)

```
1  void foo() {
2    int x = intput();
3    if (x > MIN) {
4      int z = x;
5      int y = x * 10;
6      output(y);
7    }
8  }
```

Type-3 Clone

Our path: do not use "hash" the basic unit but design a better way to calculate their "similarity"

Detection Results (32 Patches from BTC & 6->19 from ETH)

Forked Project	LOC	BlockScope (GPT)					ReDeBug				
		TP	FN	TN	FP	Time	TP	FN	TN	FP	Time
Dogecoin	326.9K	16/16	-/-	15/14	1/2	7.6s	7	9	15	1	12.5s
Bitcoin Cash	607.1K	1/1	-/-	30/28	1/3	10.5s	-	1	31	-	22.2s
Litecoin	423.3K	6/6	-/-	26/26	-/-	8.3s	5	1	26	-	16.4s
Bitcoin SV	221.1K	11/9	1/3	18/17	2/3	10.6s	2	10	19	1	9.9s
Dash	380.3K	9/8	1/2	22/17	-/5	13.9s	7	3	21	1	17.7s
Zcash	199.4K	9/9	2/2	19/17	2/4	8.4s	1	10	21	-	10.7s
Bitcoin Gold	381.7K	10/10	1/1	21/20	-/1	8.8s	10	1	21	-	17.4s
Horizen	178.9K	9/8	2/3	20/19	1/2	7.7s	1	10	21	-	12.6s
Qtum	569.0K	-/-	-/-	31/30	1/2	12.0s	-	-	32	-	33.5s
DigiByte	416.3K	10/10	1/1	21/20	-/1	10.7s	10	1	21	-	15.8s
Ravencoin	504.2K	14/14	1/1	16/17	1/-	11.4s	10	5	17	-	20.9s
Sum	**4.2M (382.6K)***	**95/91**	**9/13**	**239/225**	**9/23**	**109.9s (3.4s)◇**	**53**	**51**	**245**	**3**	**189.6s (5.9s)◇**
Binance	565.3K	7/6	-/1	12/9	-/3	2.2s	-	1	5	-	30.2s
Avalanche	1070.1K	3/1	-/2	14/11	2/5	2.5s	-	-	6	-	55.2s
Polygon	592.0K	9/7	-/2	7/7	3/3	2.3s	-	-	6	-	31.3s
Celo	631.0K	6/6	-/-	10/8	3/5	2.7s	1	-	5	-	44.5s
Optimism	583.5K	3/1	-/2	13/11	3/5	3.6s	3	1	2	-	43.3s
Kcc	562.8K	8/5	-/3	11/8	-/3	3.6s	3	1	2	-	43.3s
Heco	576.2K	6/5	-/1	10/7	3/6	3.6s	3	1	2	-	43.3s
Hoo	537.7K	10/9	-/1	7/6	2/3	3.6s	3	1	2	-	43.3s
BitTorrent	562.7K	8/8	-/-	8/6	3/5	3.6s	3	1	2	-	43.3s
Sum	**5.7M (631.3K)***	**60/48**	**-/12**	**92/73**	**19/38**	**13.3s (2.2s)◇**	**4**	**2**	**24**	**-**	**204.5s (34.1s)◇**

The Breakdown for Three Clone Types

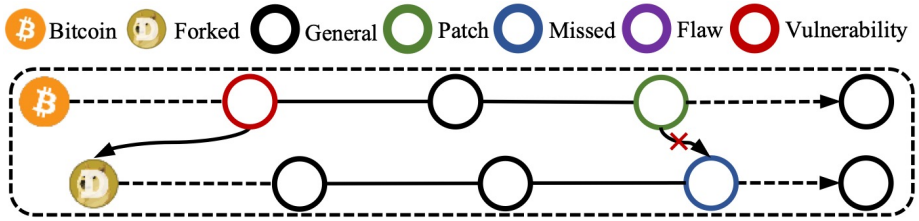
- **Type-1&3** clones occupy **95.5%** of all the cases.

- BlockScope accuracy:
  - Type-1: 100%;
  - Type-2: 80%;
  - Type-3: 85.7%.
- ReDeBug accuracy:
  - Type-1: 85.7%;
  - Type-2: 0%;
  - Type-3: 26.8%.

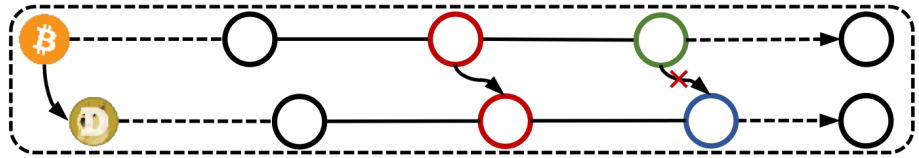
Forked Project	Type-1		Type-2		Type-3		Sum	
	T	B;R	T	B;R	T	B;R	T	B;R
Dogecoin	6	(6;4)	-	-	10	(10;3)	16	(16;7)
Bitcoin Cash	1	(1;-)	-	-	-	-	1	(1;-)
Litecoin	5	(5;5)	-	-	1	(1;-)	6	(6;5)
Bitcoin SV	1	(1;-)	-	-	11	(10;2)	12	(11;2)
Dash	7	(7;7)	-	-	3	(2;-)	10	(9;7)
Zcash	1	(1;-)	2	(1;-)	8	(7;1)	11	(9;1)
Bitcoin Gold	9	(9;8)	-	-	2	(1;2)	11	(10;10)
Horizen	-	-	2	(2;-)	9	(7;1)	11	(9;1)
Qtum	-	-	-	-	-	-	-	-
DigiByte	7	(7;7)	1	(1;-)	3	(2;3)	11	(10;10)
Ravencoin	7	(7;7)	-	-	8	(7;3)	15	(14;10)
Sum	**44**	**(44;38)**	**5**	**(4;-)**	**55**	**(47;15)**	**104**	**(95;53)**
Binance	-	-	-	-	1	(1;-)	1	(1;-)
Avalanche	-	-	-	-	-	-	-	-
Polygon	-	-	-	-	-	-	-	-
Celo	1	(1;1)	-	-	-	-	1	(1;1)
Optimism	4	(4;3)	-	-	-	-	4	(4;3)
Sum	**5**	**(5;4)**	**-**	**-**	**1**	**(1;-)**	**6**	**(6;4)**

T, B, and R represent: the total number of vulnerabilities of each clone type, the number of vulnerabilities detected by BlockScope, and the number of vulnerabilities detected by ReDeBug, respectively.

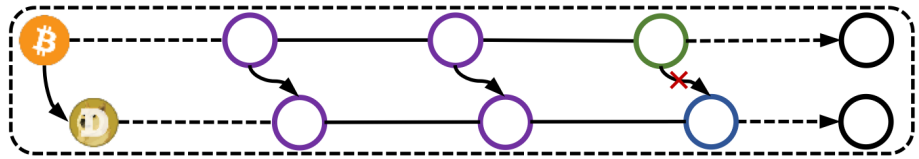
Investigation of Propagated Vulnerabilities



(a) The `fork` type: vulnerabilities directly forked in the beginning.



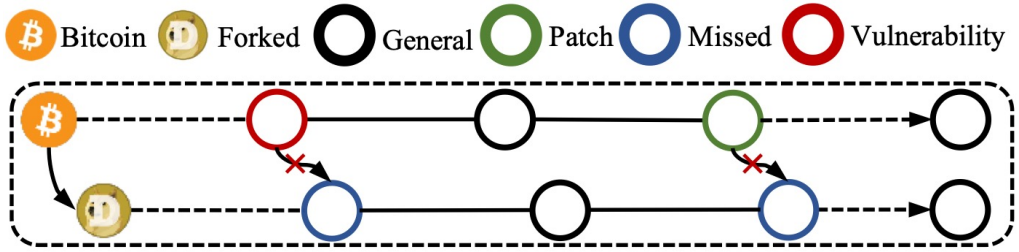
(b) The `fetch` type: vulnerabilities fetched from vulnerable commits.



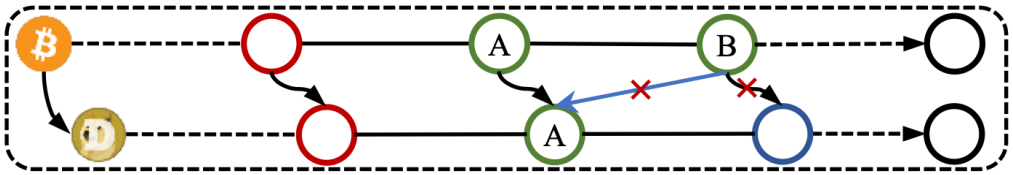
(c) The `mixed` type: vulnerabilities infected with no explicitly vulnerable commits.

- 41 cases, e.g., CVE-2022-29177, CVE-2021-41173.
- 25 cases, e.g., CVE-2021-3401, CVE-2020-26265, CVE-2020-26264, CVE-2020-26260.
- 44 cases, e.g., Bitcoin PR#16512.

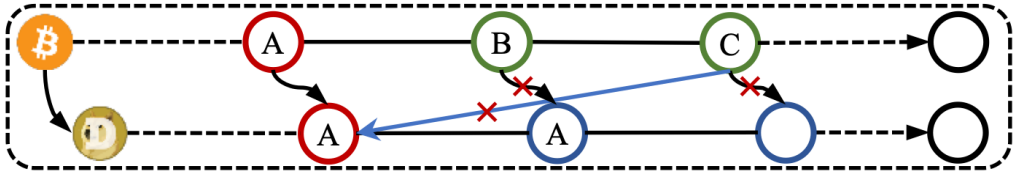
Our Limitation



(a) FP-I: no clone, and thus no vulnerability.



(b) FP-II: patch outdated.



(c) FN: target code outdated.

- FP-I: 7 cases, e.g., CVE-2018-17145, CVE-2019-15947, Bitcoin PR#12561, Bitcoin PR#14249.
- FP-II: 2 cases, e.g., Bitcoin PR#12561, Bitcoin PR#13808.
- FN: 9 cases, e.g., Bitcoin PR#10345, Bitcoin PR#11568, Bitcoin PR#13907.

Vulnerability Report Response

- Reported 110 vulnerabilities (101 TP + 9 FN);
  - 74 positive response;
  - CVE-2021-37491 of Dogecoin & CVE-2021-37492 of Ravencoin
  - 1 bug bounty from Binance;
  - Dogecoin, Ravencoin, Dash, Bitcoin Gold, Litecoin, and Binance are the most active ones;
  - Bitcoin Cash, DigiByte, and Optimism did not respond to any of our reports.

Forked Project	Fixed	Accepted	ACK	Pending	Reject	Sum
Dogecoin	11	3	2	-	-	16
Bitcoin Cash	-	-	-	1	-	1
Litecoin	2	-	3	1	-	6
Bitcoin SV	-	-	8	2	2	12
Dash	1	5	3	1	-	10
Zcash	-	-	9	1	1	11
Bitcoin Gold	7	-	1	3	-	11
Horizen	-	-	4	7	-	11
Qtum	-	-	-	-	-	-
DigiByte	-	-	-	11	-	11
Ravencoin	9	1	3	1	1	15
Sum	**30**	**9**	**33**	**28**	**4**	**104**
Binance	-	1	-	-	-	1
Avalanche	-	-	-	-	-	-
Polygon	-	-	-	-	-	-
Celo	-	-	1	-	-	1
Optimism	-	-	-	4	-	4
Sum	**-**	**1**	**1**	**4**	**-**	**6**

Our vulnerability discovery in BSC/Optimism/Base/Mantle

- DoS caused by malicious P2P message (2 (Med Risk))
  Submitted 1 minute ago
- Chain split caused by consensus flaw in Geth (2 (Med Risk))
  Submitted 3 minutes ago
- DoS caused by malicious GetProofsV2 request (2 (Med Risk))
  Submitted 5 minutes ago
- Incorrect DAG generation result caused by index overflow (3 (High Risk))
  Submitted 6 minutes ago
- Chain split caused by memory corruption in EVM (3 (High Risk))
  Submitted 9 minutes ago

CVE-2022-29177 in Binance BSC

DoS caused by malicious P2P message

Submitted about 2 years ago • Last activity about 2 years ago

ID	`d7c50ffe-885a-4d87-99af-c5bb1ea23b51`
Submitted	21 Jun 2022 04:50:37 UTC
Target Location	`https://github.com/bnb-chain/bsc`
Target category	Other
VRT	Application-Level Denial-of-Service (DoS) > App Crash
Priority	P3
Bug URL	https://github.com/bnb-chain/bsc/blob/70d08a5791d0650322e79591ac1fb869df607586/p2p/peer.go#L343
Description	**DoS caused by malicious P2P message** We recently found that the current version of **Binance** (bnb-chain/bsc) has a DoS vulnerability, where a vulnerable node, if configured to use high verbosity logging, can be made to crash when handling specially crafted p2p messages sent from an attacker node. Specifically, the disconnect reason `DiscReason` in `p2p/peer_error.go` is defined as `uint`, which may lead to a crash when decoding the message at line 343 of `p2p/peer.go`. The consequence of this vulnerability is the same as CVE-2022-29177. A simple fix is to change the definition of `DiscReason` into `uint8` at line 57 of `p2p/peer_error.go` and the type of `reason` into `struct{R DiscReason}` at line 340 of `p2p/peer.go`. The MagicSecurity team (https://github.com/MagicLabHK; https://twitter.com/0xMagicSec)

#HITB2024BKK

CVE-2022-29177 in Binance BSC

A vulnerable node, if configured to use high verbosity logging, can be made to crash when handling specially crafted p2p messages sent from an attacker node.

The disconnect reason `DiscReason` in `p2p/peer_error.go` is defined as `uint`, which may lead to a crash when decoding the message at line 343 of `p2p/peer.go`.

A fix is to change the definition of `DiscReason` into `uint8` at line 57 of `p2p/peer_error.go` and the type of `reason` into `struct{R DiscReason}` at line 340 of `p2p/peer.go`.

```
56
57      type DiscReason uint
58
59      const (
60              DiscRequested DiscReason = iota
61              DiscNetworkError
```

```
339             case msg.Code == discMsg:
340                     var reason [1]DiscReason
341                     // This is the last message. We don't
342                     // check errors because, the connectic
343                     rlp.Decode(msg.Payload, &reason)
344                     return reason[0]
```

#HITB2024BKK

CVE-2020-26265 in Optimism

Chain split caused by consensus flaw in Geth

We recently found that the current version of **Optimism** (ethereum-optimism/optimism) has a consensus flaw in `l2geth`, where a particular sequence of transactions could cause a consensus failure.

- Tx1
  - `sender` invokes `caller`.
  - `caller` invokes `0xaa`. `0xaa` has 3 wei, does a self-destruct-to-self.
  - `caller` does a 1-wei-call to `0xaa`, who thereby has 1 wei (the code in `0xaa` still executed, since the tx is still ongoing, but doesn't redo the self-destruct, it takes a different path if callvalue is non-zero)
- Tx2
  - `sender` does a 5-wei call to `0xaa`. No exec (since no code).

In `l2geth`, the result would be that `0xaa` had 6 wei, whereas OE reported (correctly) 5 wei. Furthermore, in `l2geth`, if the second tx was not executed, the `0xaa` would be destructed, resulting in 0 wei. Thus obviously wrong.

CVE-2020-26265 in Optimism

The problem lies in line 307-311 and line 316 of `l2geth/consensus/ethash/algorithm.go`. Substitute the `uint32` into `uint64` could fix the issue.

```
306                 // Calculate the data segment this thread should generate
307                 batch := uint32((size + hashBytes*uint64(threads) - 1) /
308                 first := uint32(id) * batch
309                 limit := first + batch
310                 if limit > uint32(size/hashBytes) {
311                     limit = uint32(size / hashBytes)
312                 }
313             // Calculate the dataset segment
314             percent := uint32(size / hashBytes / 100)
315             for index := first; index < limit; index++ {
316                 item := generateDatasetItem(cache, index, keccak512)
317                 if swapped {
318                     swap(item)
319                 }
320                 copy(dataset[index*hashBytes:], item)
```

CVE-2020-26264 in Base

It had a DoS vulnerability in LES, which can make a LES server crash via a malicious `GetProofsV2` request from a connected LES client.

The problem lies in line 595 of `l2geth/les/server_handler.go`, where the `header` could potentially be `nil` and leads to a panic.

A simple solution is to move line 595 after line 579.

```
576         // Gather state data until the fetch or network limits is reached
577         var (
578             lastBHash common.Hash
579             root      common.Hash
580         )
581         reqCnt := len(req.Reqs)
582         if accept(req.ReqID, uint64(reqCnt), MaxProofsFetch) {
583             wg.Add(1)
584             go func() {
585                 defer wg.Done()
586                 nodes := light.NewNodeSet()
587
588                 for i, request := range req.Reqs {
589                     if i != 0 && !task.waitOrStop() {
590                         sendResponse(req.ReqID, 0, nil, task.servingTime)
591                         return
592                     }
593                     // Look up the root hash belonging to the request
594                     var (
595                         header *types.Header
596                         trie   state.Trie
597                     )
```

#HITB2024BKK

CVE-2021-39137 in Mantle

It has a memory corruption vulnerability in EVM, which can cause a consensus error

Vulnerable nodes obtain a different `stateRoot` when processing a maliciously crafted transaction. This, in turn, would lead to the chain being split into two forks.

The problem lies in four functions, i.e., `opCall`, `opCallCode`, `opDelegateCall`, and `opStaticCall` of `core/vm/instructions.go`.

A simple solution is to use `common.CopyBytes` to copy `ret` safely before use, e.g., add `ret = common.CopyBytes(ret)` before line 698.

Acknowledgement

This work is made possible with my former PhD student, Xiao Yi (now a Researcher at Huawei HKRC), and Research Assistant, Yuzhou Fang (now a PhD student at USC).

BlockScope is now open-source at https://github.com/VPRLab/BlockScope.

Whitepaper: https://www.ndss-symposium.org/ndss-paper/blockscope.

BlockScope: Detecting and Investigating Propagated Vulnerabilities in Forked Blockchain Projects

Xiao Yi[1], Yuzhou Fang[1], Daoyuan Wu[1*], and Lingxiao Jiang[2]
[1]The Chinese University of Hong Kong
[2]Singapore Management University

***Abstract*—Due to the open-source nature of the blockchain ecosystem, it is common for new blockchains to fork or partially reuse the code of classic blockchains. For example, the popular Dogecoin, Litecoin, Binance BSC, and Polygon are all variants of Bitcoin/Ethereum. These "forked" blockchains thus could** Ethereum was also forked by a number of EVM (Ethereum Virtual Machine)-compatible chains, such as Binance Smart Chain (BSC), Polygon, Avalanche Contract Chain, and Optimism (Ethereum's Layer-2 rollup network).

Takeaway

introduced our recent efforts to discover how Ethereum's CVE vulnerabilities could propagate from Ethereum to BSC/Optimism/Base/Mantle.

Developed BlockScope (https://github.com/VPRLab/BlockScope), a novel search-based patch vs. code similarity analysis tool for discovering 100+ vulnerabilities in top blockchains.

Analyzed vulnerabilities in BSC/Optimism/Base/Mantle (1 for BSC, 4 for Optimism, and 5 for Base/Mantle).

Thank you!
Q&A ☺
Contact: @MagkDao

#HITB2024BKK