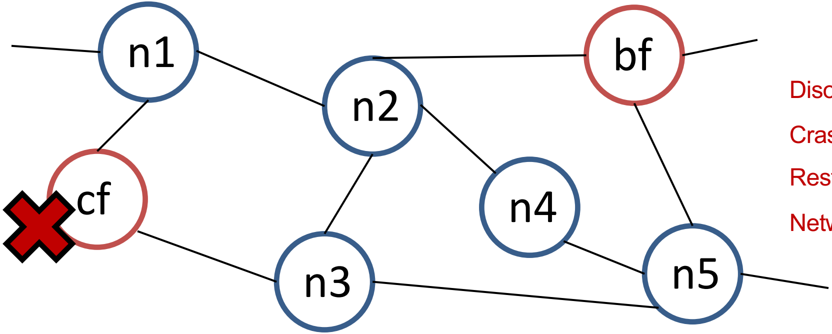
LOKI: State-Aware Fuzzing Framework for the Implementation of Blockchain Consensus Protocols

Fuchen Ma, Yuanliang Chen, Meng Ren, Yuanhang Zhou, Yu Jiang,
Ting Chen, Huizhong Li and Jiaguang Sun

Why we need Consensus Protocols

Originally talked in distributed systems…



Disorder

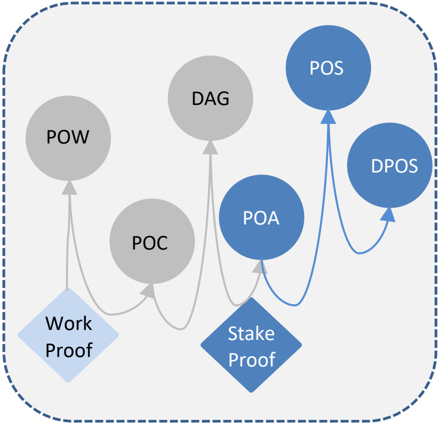
Crash

Restart

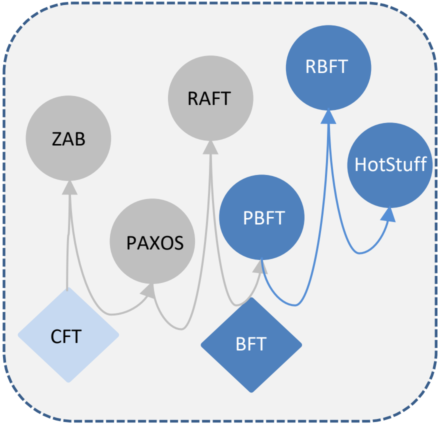
Network env…

A fundamental problem in distributed computing and multi-agent systems is to achieve overall system reliability in the presence of a number of faulty processes.

In Blockchain System, how we get consensus?



Based on Game Theory



Based on CAP theorem

An Example of Blockchain Consensus Protocols

PBFT(Practical Byzantine Fault Tolerant)? (OSDI 1999 By Lamport)

Preprepare phase：

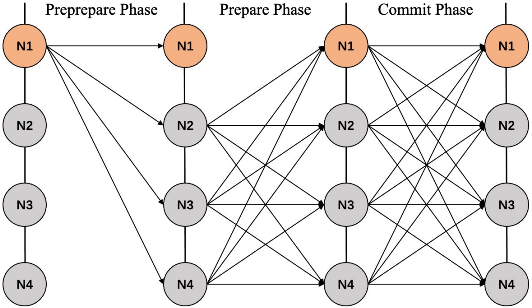
- Leader broadcasts prepare packet with new blocks.
- Other nodes execute the block and send sign packets.

Prepare phase：

- If the node has received 2f+1 sign packets, send the commit packet.

Commit phase：

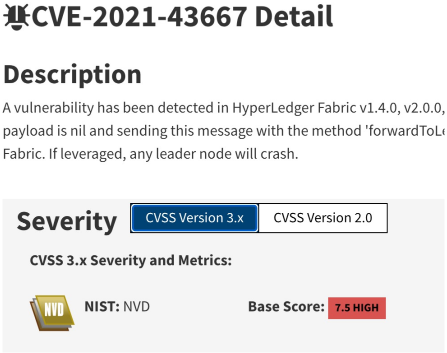
- If the node has received 2f+1 commit packets, record the blocks into the blockchain.



- **The protocol can tolerate f byzantine nodes in a network with 3f+1 nodes;**
- **leaderid = (BlockHeight + viewid) % node_num.**
- **If the consensus time exceeds the timeout, viewChange**

Consensus Protocol Vulnerabilities in recent years

A vulnerability in Hyperledger Fabric



CVE-2021-43667 Detail

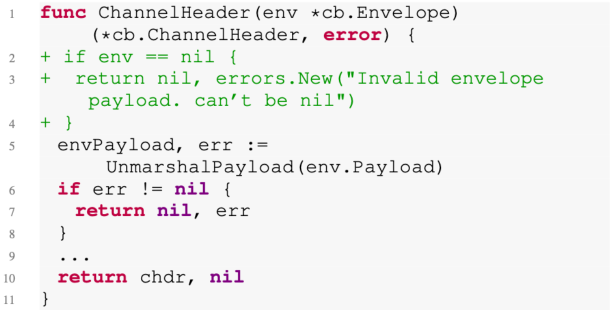
Description

A vulnerability has been detected in HyperLedger Fabric v1.4.0, v2.0.0,
payload is nil and sending this message with the method 'forwardToL
Fabric. If leveraged, any leader node will crash.

Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** **7.5 HIGH**

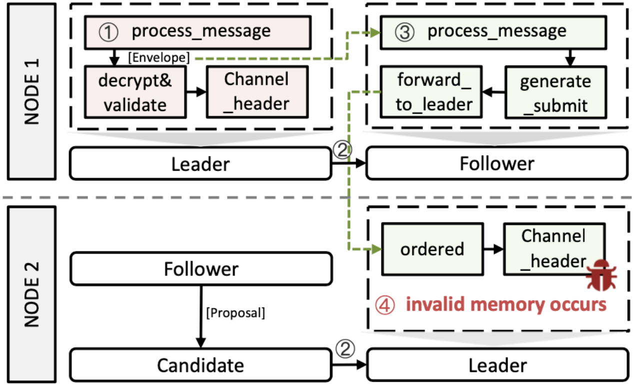


```
func ChannelHeader(env *cb.Envelope)
    (*cb.ChannelHeader, error) {
+ if env == nil {
+   return nil, errors.New("Invalid envelope
      payload. can't be nil")
+ }
  envPayload, err :=
      UnmarshalPayload(env.Payload)
  if err != nil {
    return nil, err
  }
  ...
  return chdr, nil
}
```

Though consensus protocols are proved to be correct and complete in theory, they may contain code flaws during implementation,

Consensus Protocol Vulnerabilities in recent years

4 basic steps to trigger this bug



① Node 1 decrypts and validates all the 'Envelope' messages.

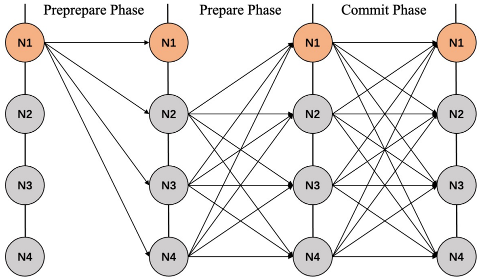
② Node 2 is elected as a new leader and Node 1 is the follower.

③ Some unprocessed 'Envelope' messages in Node 1 will be forwarded to the new leader.

④ Node2 handles these packets by 'ordered' function without validating them.

Challenges to Test Consensus Protocols

Challenge 1: How to model the consensus state in real-time?



States of consensus protocols are dynamic and complex.

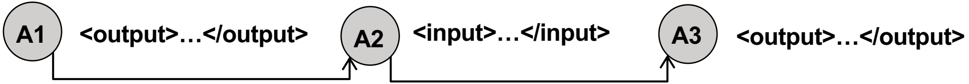
- Consensus phase is dynamically changing.
- Nodes may be in different consensus rounds.

Sending packets without knowing the states is ineffective.

- Cannot progress the consensus and test deeper logic.
- Most invalid packets.

How Peach acts ❌

State model is fixed, thus most of the testing inputs are invalid.



A1 <output>…</output>

A2 <input>…</input>

A3 <output>…</output>

How to Test Consensus Protocols

Challenge 2: How to construct multi-dimension inputs according to the state?

What to send? (Type & Content)

- Which type of the packet should be sent?
- How to generate the packet field more reasonably?

PIT **<DataModel>…</DataModel>**

Types are fixed and field are mutated ‘randomly’.

Where to send? (Target)

- Leader node or not?
- Send to one node or 1/3 of the nodes or all other nodes?

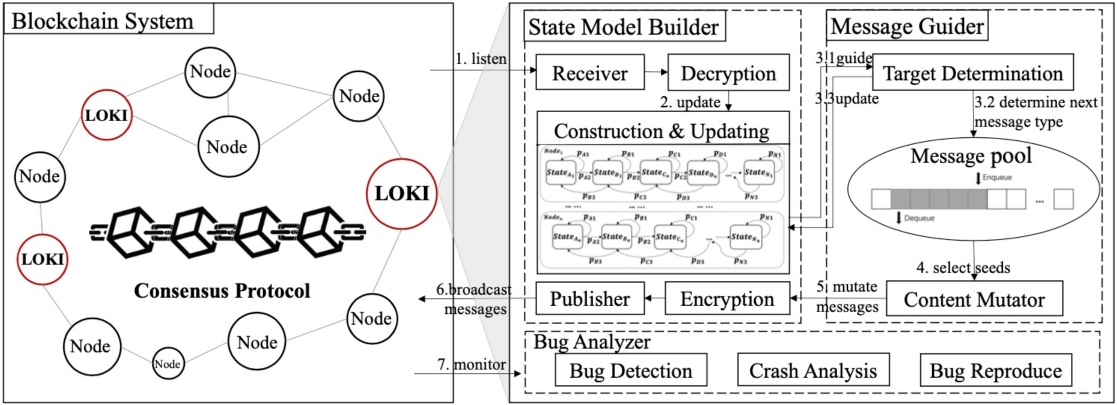
PIT **<Monitor class=“Process”>…</Monitor>**

Destinations are fixed in the data model

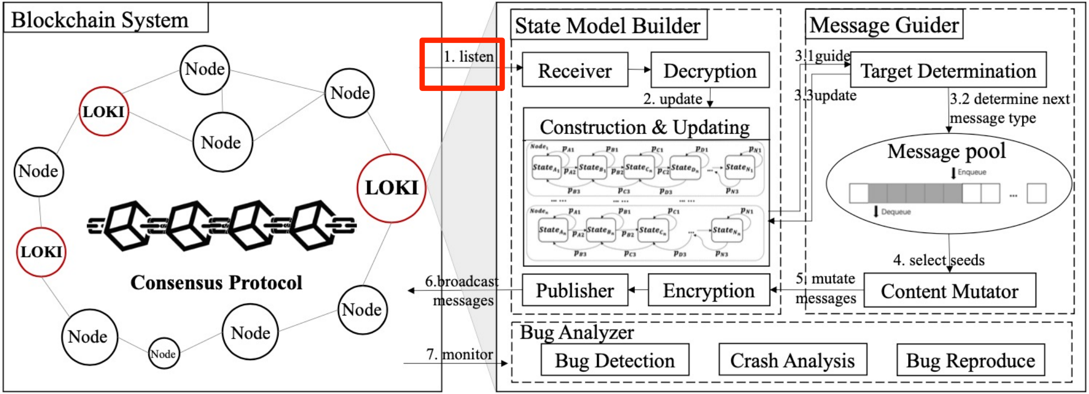
How Peach acts ❌

LOKI Overall Design

Key Insight:Masquerading as a normal node to fetch neighbour's states and fuzz

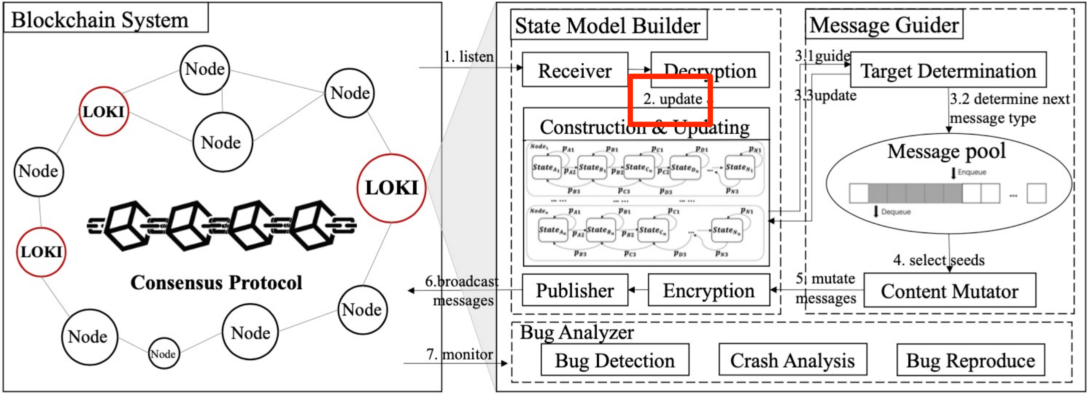


LOKI Step 1: listen from the blockchain node



1) LOKI first listens from the blockchain and decrypts the received Messages.

LOKI Step 2: State model construction and update



2) The State model builder constructs and updates the state model according to the received messages.

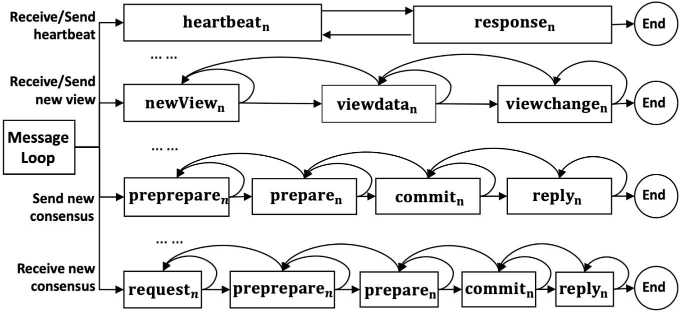
LOKI Step 2: State model construction and update

State model Construction:

- Before the fuzzing process.
- Decrypt and analyze the type of the received messages.
- Each edge has a same weight

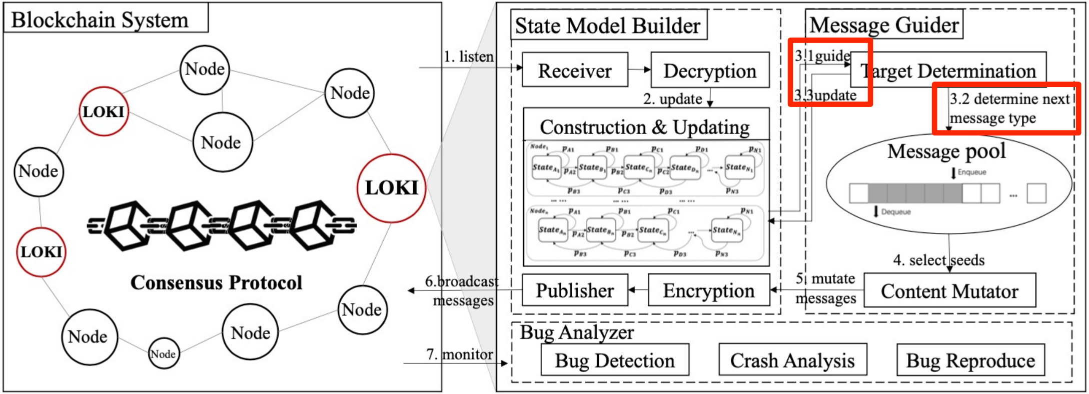
State model Update:

- Together with the fuzzing process.
- If a new message is received, LOKI adds a new edge and tracks the subsequent messages.



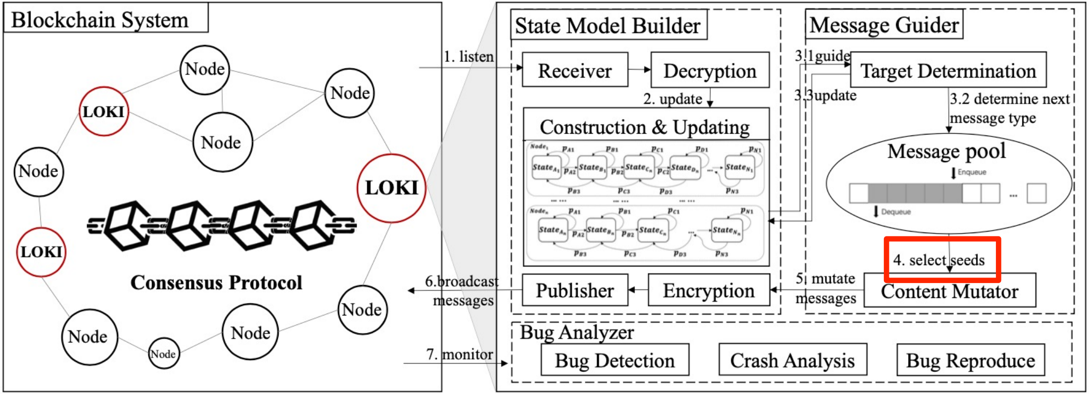
An example of PBFT state model

LOKI Step 3: Message type decision



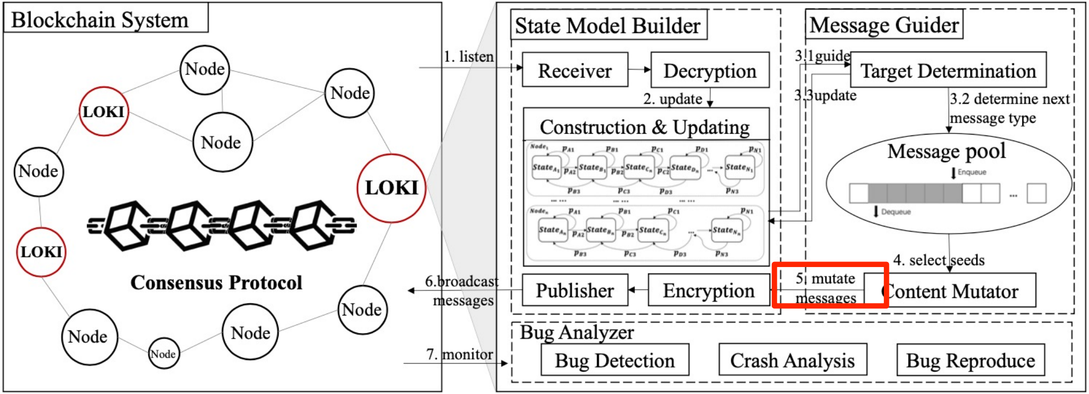
3) Message Guider determines next message type based on the state model and updates it.

LOKI Step 4: Seed Selection



4) LOKI selects the corresponding seeds from the message pool with the chosen message type.

LOKI Step 5: Message mutation



5) Content Mutator mutates seeds according to message specifications and generates new messages.

LOKI Step 5: Message mutation

Numeric type mutation:

- Randomly convert it to another number.
- Especially, to border values such as INT_MAX and 0.

String type mutation:

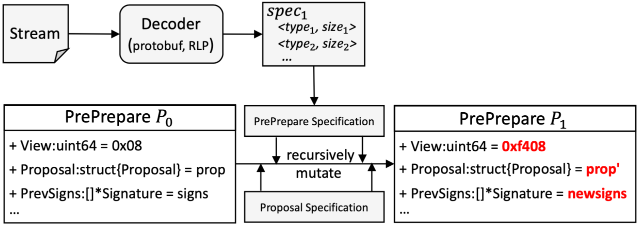
- Bytes/bits flip.

Structure type mutation:

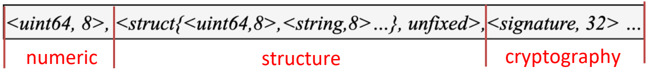
- Recursively mutates each field.

Cryptography field mutation:

- Use the inherent cryptographic components.
- Mutate other content before the cryptography related fields.



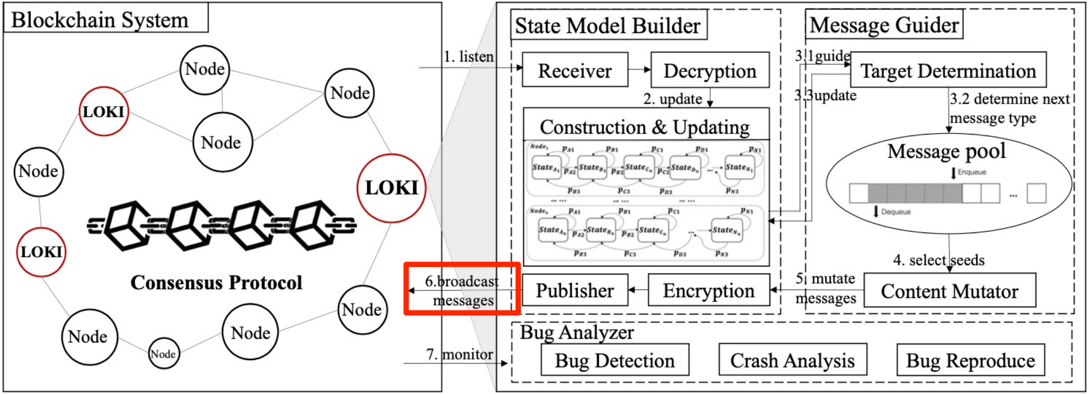
An example of message mutation



$<uint64,\ 8>$, $<struct\{<uint64,8>,<string,8>\ldots\},\ unfixed>$, $<signature,\ 32>\ \ldots$
numeric
structure
cryptography

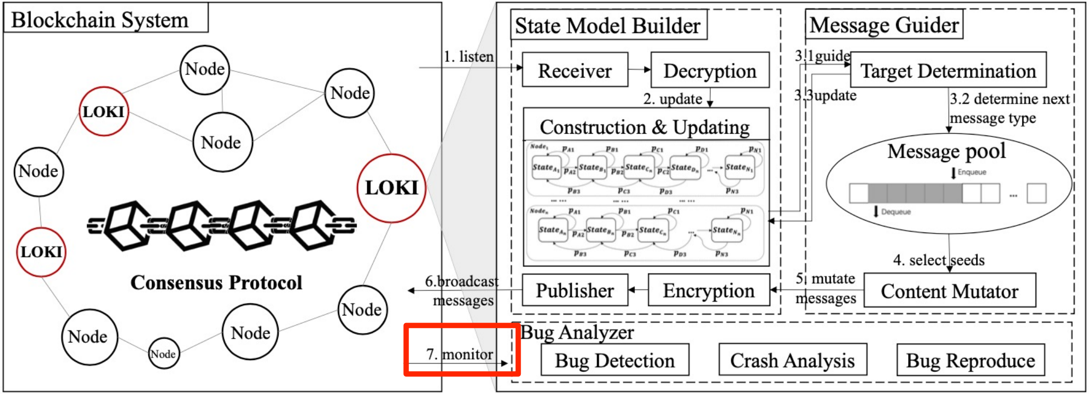
An example of structure type mutation

LOKI Step 6: Send mutated messages



6) Finally, LOKI encrypts and signs the messages and then sends them.

LOKI Step 7: Bug analyzer monitoring



7) The bug analyzer monitors the execution information of blockchain system all the time.

LOKI Step 7: Bug analyzer monitoring

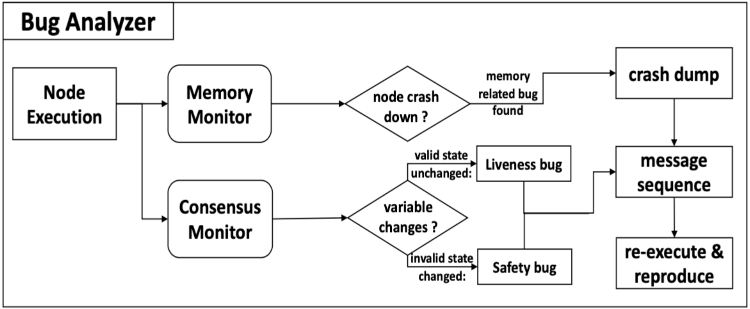
2 types of bugs supported

Memory-related bugs:

- ASAN instrumented. Debug with GDB/LLDB.
- Lead to node crash.

Consensus Logic bugs:

- Liveness bugs and Safety bugs.
- Lead to consensus failure or invalid data store.



20 Bugs Found in 4 Blockchain Systems

20 bugs in 4 blockchain systems with 9 CVE ids

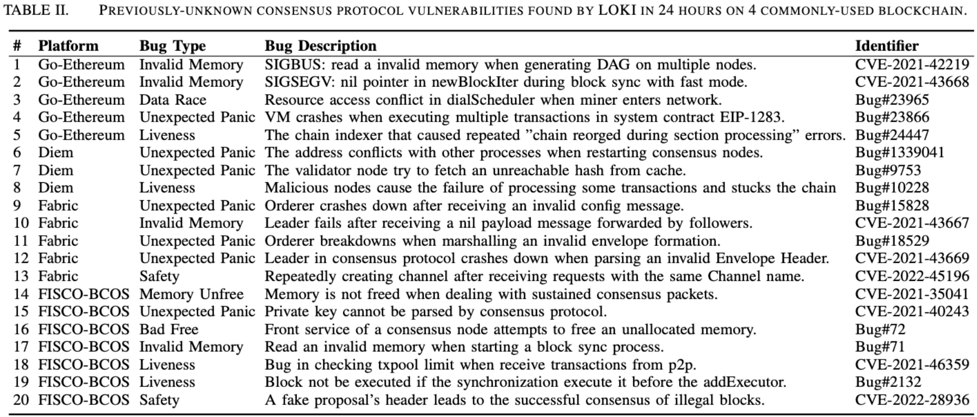


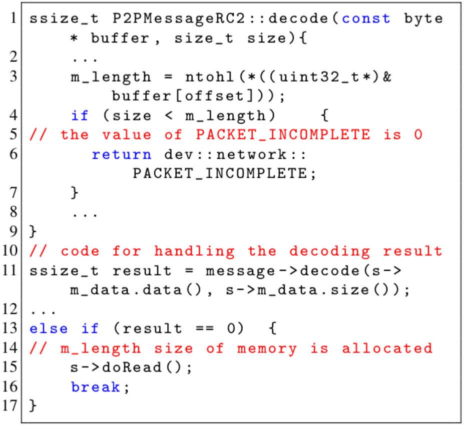
TABLE II. PREVIOUSLY-UNKNOWN CONSENSUS PROTOCOL VULNERABILITIES FOUND BY LOKI IN 24 HOURS ON 4 COMMONLY-USED BLOCKCHAIN.

#	Platform	Bug Type	Bug Description	Identifier
1	Go-Ethereum	Invalid Memory	SIGBUS: read a invalid memory when generating DAG on multiple nodes.	CVE-2021-42219
2	Go-Ethereum	Invalid Memory	SIGSEGV: nil pointer in newBlockIter during block sync with fast mode.	CVE-2021-43668
3	Go-Ethereum	Data Race	Resource access conflict in dialScheduler when miner enters network.	Bug#23965
4	Go-Ethereum	Unexpected Panic	VM crashes when executing multiple transactions in system contract EIP-1283.	Bug#23866
5	Go-Ethereum	Liveness	The chain indexer that caused repeated ”chain reorged during section processing” errors.	Bug#24447
6	Diem	Unexpected Panic	The address conflicts with other processes when restarting consensus nodes.	Bug#1339041
7	Diem	Unexpected Panic	The validator node try to fetch an unreachable hash from cache.	Bug#9753
8	Diem	Liveness	Malicious nodes cause the failure of processing some transactions and stucks the chain	Bug#10228
9	Fabric	Unexpected Panic	Orderer crashes down after receiving an invalid config message.	Bug#15828
10	Fabric	Invalid Memory	Leader fails after receiving a nil payload message forwarded by followers.	CVE-2021-43667
11	Fabric	Unexpected Panic	Orderer breakdowns when marshalling an invalid envelope formation.	Bug#18529
12	Fabric	Unexpected Panic	Leader in consensus protocol crashes down when parsing an invalid Envelope Header.	CVE-2021-43669
13	Fabric	Safety	Repeatedly creating channel after receiving requests with the same Channel name.	CVE-2022-45196
14	FISCO-BCOS	Memory Unfree	Memory is not freed when dealing with sustained consensus packets.	CVE-2021-35041
15	FISCO-BCOS	Unexpected Panic	Private key cannot be parsed by consensus protocol.	CVE-2021-40243
16	FISCO-BCOS	Bad Free	Front service of a consensus node attempts to free an unallocated memory.	Bug#72
17	FISCO-BCOS	Invalid Memory	Read an invalid memory when starting a block sync process.	Bug#71
18	FISCO-BCOS	Liveness	Bug in checking txpool limit when receive transactions from p2p.	CVE-2021-46359
19	FISCO-BCOS	Liveness	Block not be executed if the synchronization execute it before the addExecutor.	Bug#2132
20	FISCO-BCOS	Safety	A fake proposal’s header leads to the successful consensus of illegal blocks.	CVE-2022-28936

Case 1: FISCO-BCOS Consensus Protocol Bug (CVE-2021-35041)

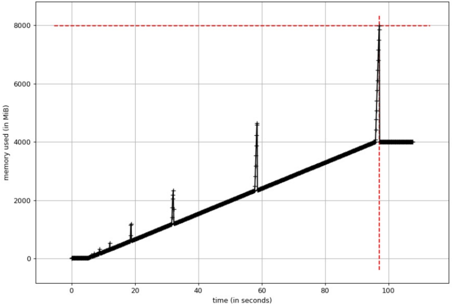
Node Killed by OS when handling malicious packets

Listing 1: Code that constantly allocate new memory



```
ssize_t P2PMessageRC2::decode(const byte
    * buffer, size_t size){
    ...
    m_length = ntohl(*((uint32_t*)&
        buffer[offset]));
    if (size < m_length)    {
// the value of PACKET_INCOMPLETE is 0
      return dev::network::
          PACKET_INCOMPLETE;
    }
    ...
}
// code for handling the decoding result
ssize_t result = message->decode(s->
    m_data.data(), s->m_data.size());
...
else if (result == 0)   {
// m_length size of memory is allocated
    s->doRead();
    break;
}
```

- The node will consistently allocate for a big memory and be killed by the OS finally.
- An attacker can craft a packet with a big value of m_length.



Compared with Existing Tools: Peach, Fluffy & Twins

Coverage Comparison among other tools

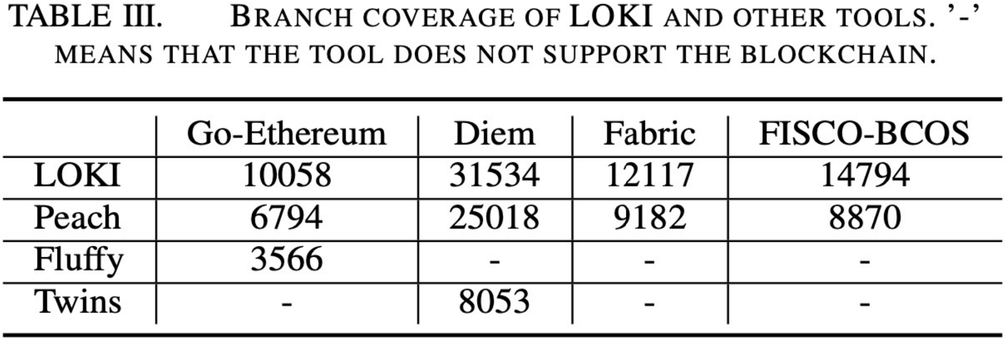
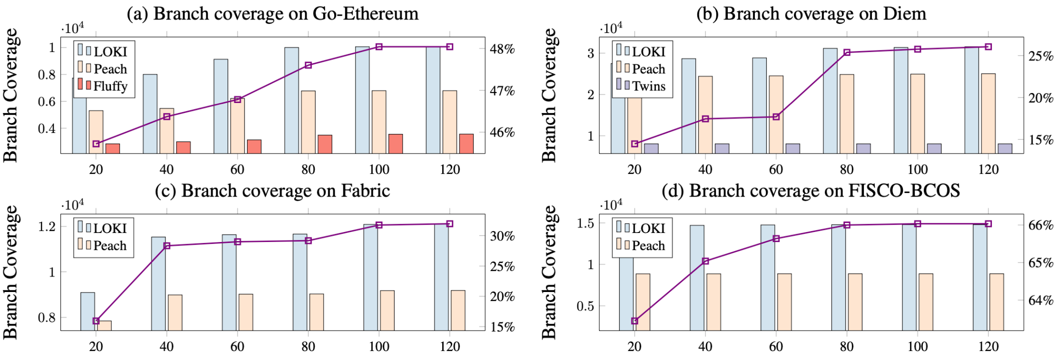


TABLE III. BRANCH COVERAGE OF LOKI AND OTHER TOOLS. '-' MEANS THAT THE TOOL DOES NOT SUPPORT THE BLOCKCHAIN.

	Go-Ethereum	Diem	Fabric	FISCO-BCOS
LOKI	10058	31534	12117	14794
Peach	6794	25018	9182	8870
Fluffy	3566	-	-	-
Twins	-	8053	-	-

Compared with Existing Tools: Peach, Fluffy & Twins

Coverage Trends Comparison among other tools



(a) Branch coverage on Go-Ethereum

(b) Branch coverage on Diem

(c) Branch coverage on Fabric

(d) Branch coverage on FISCO-BCOS

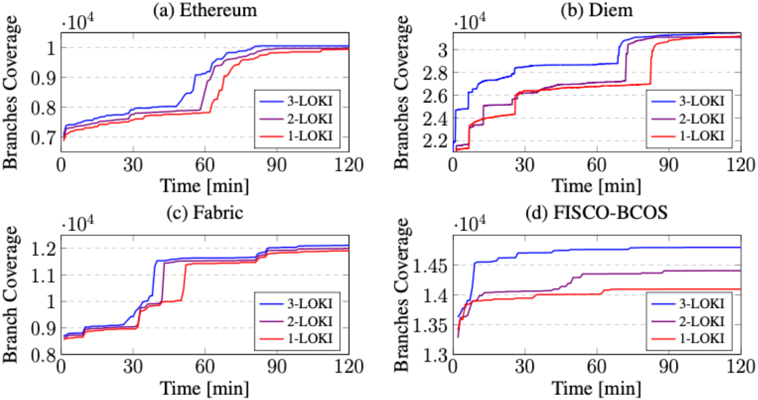
Bars refer to the coverage while the lines describe the increment percentage of LOKI compared with Peach.

Multiple LOKI nodes

Coverage under different numbers of LOKI nodes

Results

- Coverage finally converges at a similar value. (with +/- 5% of the variance)
- More LOKI nodes may slightly accelerate the coverage increment.



Future Work

- More Oracles
    - Hard fork?
    - Leader fairness?
- Cooperation among LOKI nodes
    - Share the message pool
    - Targeted at the same nodes
- More Blockchains
    - Quorum

Thank You

Please contact: mafc19@mails.tsinghua.edu.cn for more details