Breaking Down Web3 Attack Surfaces

A Dive into Consensus, VMs, Smart Contracts, and Toolchains

Kevin Wang @ Anatomist Security

AGENDA

01 BACKGROUND

02 SMART CONTRACTS

03 TOOLCHAINS

04 VIRTUAL MACHINES

05 CONSENSUS

ANATOMIST
SECURITY

We are ANATOMIST

王建元 Kevin Wang

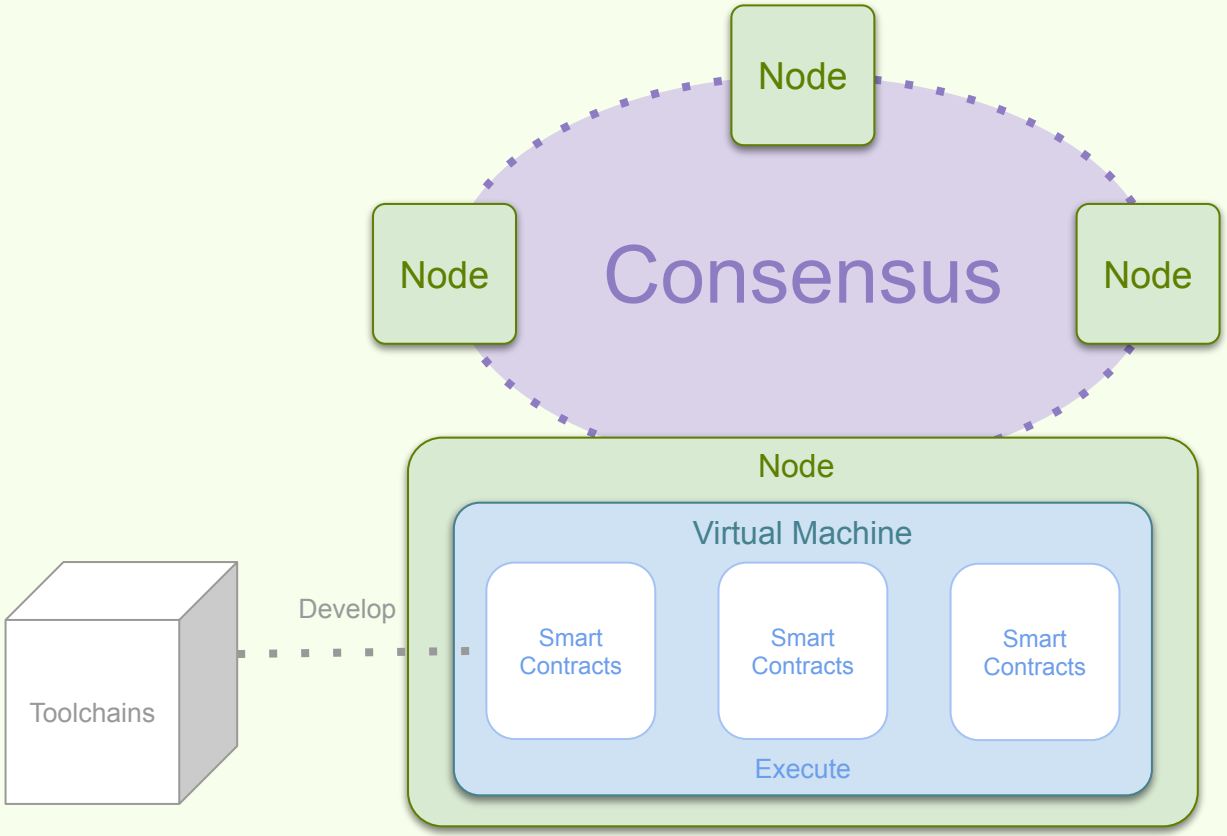
- Co-Founder of Anatomist Security
- Balsn CTF Team
- Top 3 at DEFCON CTF, HITCON CTF
- Speaker at HITCON, Cybersec
- Reported vulnerability to TP-Link, Netgear, Realtek, etc.

ANATOMIST SECURITY

01

Background

WEB3 ECOSYSTEM



02

Smart Contracts

SECURITY CRITERIA

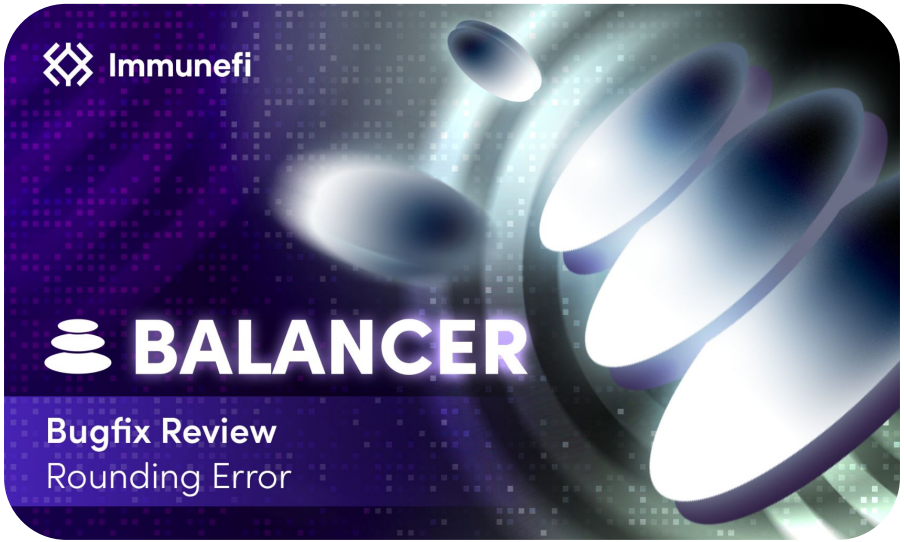
Implementation Correctness	• Arithmetic safety (overflow/underflow, rounding errors) • Guarantee accurate business logic
VM Specific Behaviors	• Ex: Reentrancy on EVM, account permission check on SVM
Environmental Risks	• Ex: Reduces risk of MEV, sandwich attacks

SECURITY CRITERIA

Implementation Correctness	• Arithmetic safety (overflow/underflow, rounding errors) • Guarantee accurate business logic
VM Specific Behaviors	• Ex: Reentrancy on EVM, account permission check on SVM
Environmental Risks	• Ex: Reduces risk of MEV, sandwich attacks

BALANCER

- Balancer is a decentralized automated market maker (AMM)
- Designed to be a flexible building block for programmable liquidity



THE PROBLEM

- Imagine an exchange where 1 A = 0.8 B
  - To get 1 B, you must pay 1.125 A
- But for very small swaps (e.g., 1 unit of B),
  - Due to precision limits, the exchange can't charge 1.125 unit of A

Ref: https://medium.com/immunefi/balancer-rounding-error-bugfix-review-cbf69482ee3d

THE PROBLEM

- Correct behavior:
  - Always round up to ensure the exchange receives more than it gives
  - Charge 2 unit of A for 1 unit of B
- Balancer's bug:
  - In certain cases, it rounds down to 1 unit of A
  - User gets more value than they pay → Loss of funds

Ref: https://medium.com/immunefi/balancer-rounding-error-bugfix-review-cbf69482ee3d

THE IMPACT

- Drain 20% of Balancer's $1 billion TVL at the time
- Pay 1,000,000 USDC bounty

03

Toolchains

SECURITY CRITERIA

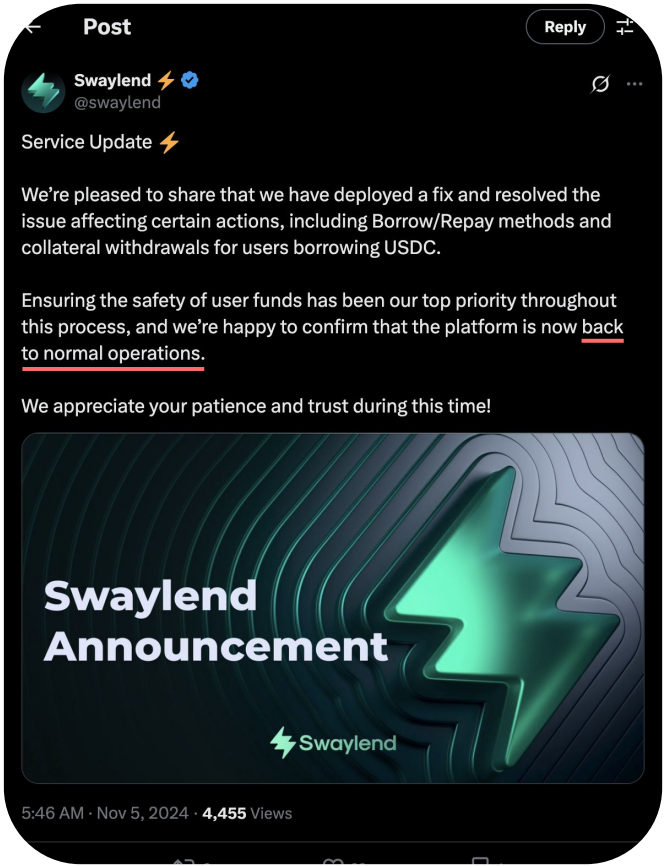
Correctness	The compiler must accurately translate high-level code to bytecode
Optimization Safety	Compiler optimizations should not introduce unintended side effects

SECURITY CRITERIA

Correctness	The compiler must accurately translate high-level code to bytecode
Optimization Safety	Compiler optimizations should not introduce unintended side effects

SWAY → SWAYLEND

- Sway is a programming language designed for implementing smart contracts on blockchain platforms
  - Most notably for the Fuel Virtual Machine (Fuel VM)
- SwayLend is a decentralized lending protocol



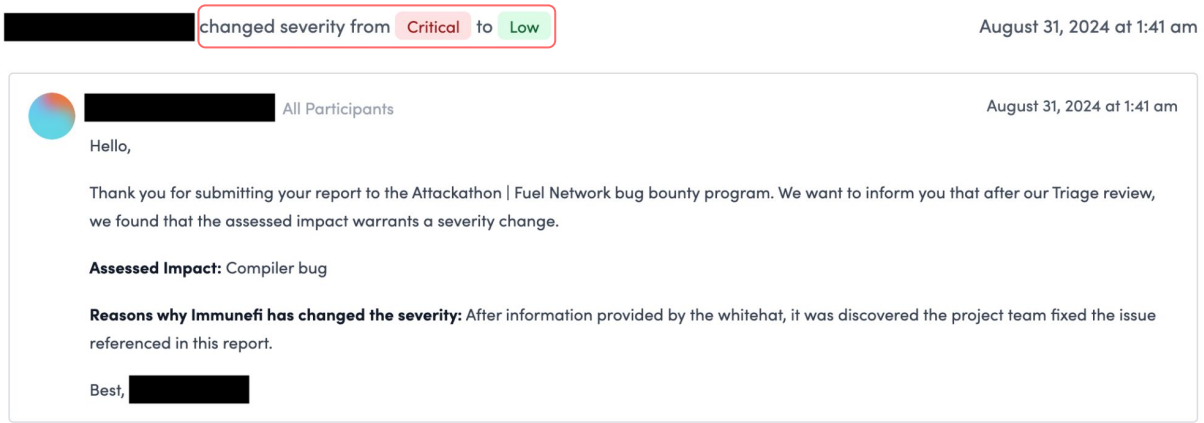
THE PROBLEM

- The Sway compiler allocates a memory chunk of fixed size 1024 for the Buffer structure
- Once we have a controllable input over 1024 bytes, we get a out-of-bounds write vulnerability

```
1  Intrinsic::EncodeBufferEmpty => {
2      assert!(arguments.is_empty());
3
4      let uint64 = Type::get_uint64(context);
5
6      // let cap = 1024;
7      let cap = Value::new_constant(
8          context,
9          Constant {
10             ty: uint64,
11             value: ConstantValue::Uint(1024),
12         },
13     );
14
```

THE REAL PROBLEM

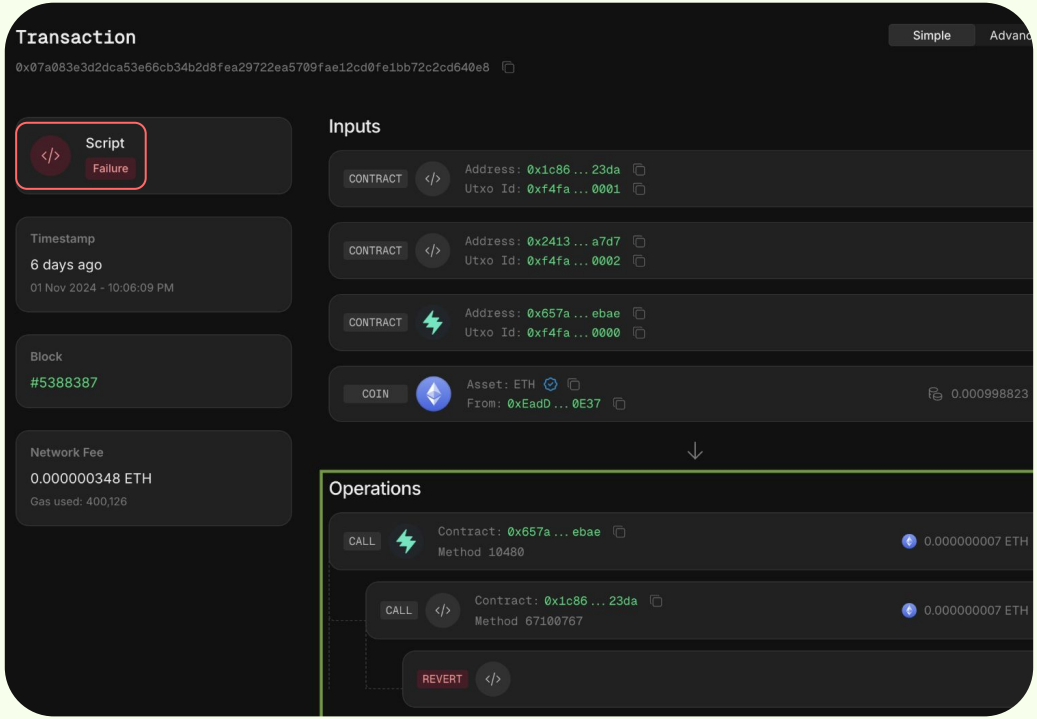
- In our report, the bug was first considered out of scope
- After some discussions, they acknowledged it but downgraded the severity from critical to low
    - And didn't patch it, of course



CONSEQUENTLY...

SWAYLEND SHUTDOWN

2 Days



CONSEQUENTLY...

SWAYLEND SHUTDOWN

2 Days

04

Virtual Machines

SECURITY CRITERIA

Memory Safety	Prevents unauthorized memory access or modifications
State Isolation	Contracts should not arbitrarily modify each other's state
Resource Control	Prevents infinite loops and Denial-of-Service attacks via metering

SECURITY CRITERIA

Memory Safety	Prevents unauthorized memory access or modifications
State Isolation	Contracts should not arbitrarily modify each other's state
Resource Control	Prevents infinite loops and Denial-of-Service attacks via metering

SOLANA

- High-performance blockchain written in Rust
- Monolithic architecture with a custom runtime optimized for speed and parallelism

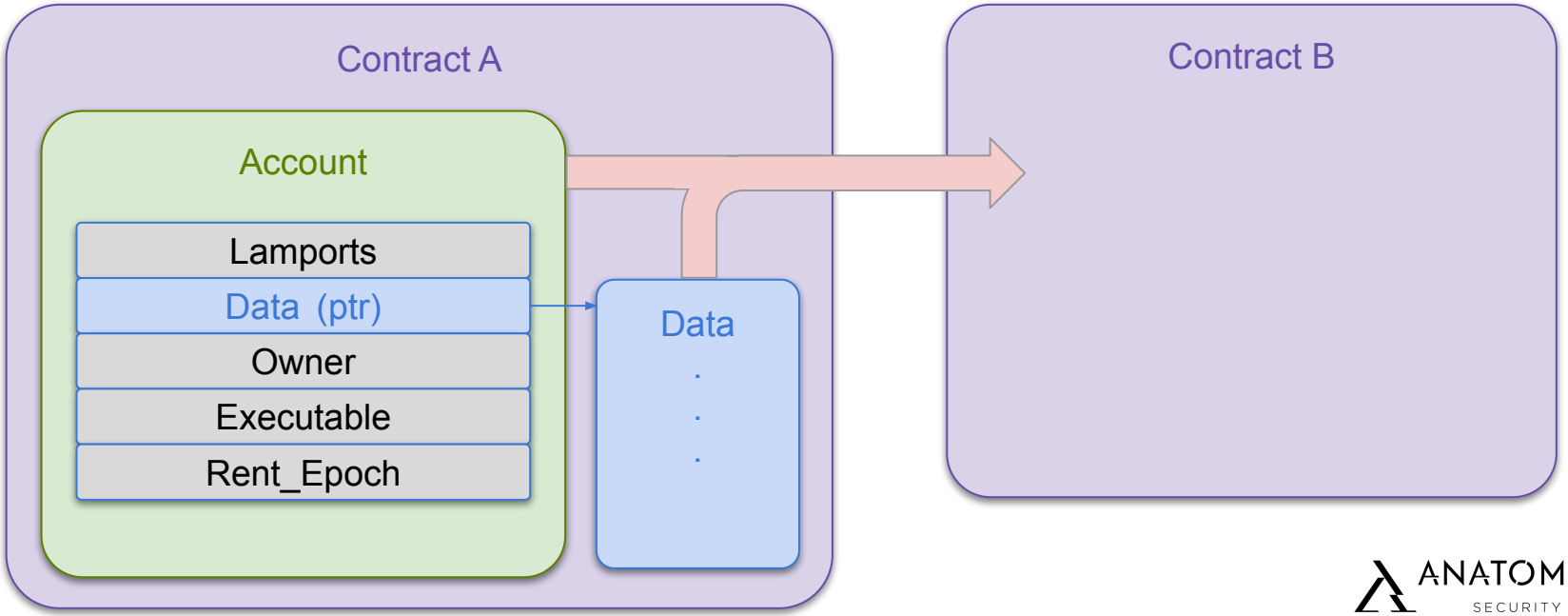
```
// guaranteed to be unique.
let cache = unsafe { &mut *self.cache.get() };

let mut src = &value as *const _ as *const u8;

let mut region = match self.find_region(cache, vm_addr) {
    Some(region) if ensure_writable_region(region, &self.cow_cb) => {
        // fast path
        if let ProgramResult::Ok(host_addr) = region.vm_to_host(vm_addr, len) {
            // Safety:
            // vm_to_host() succeeded so we know there's enough space to
            // store `value`
            unsafe { ptr::write_unaligned(host_addr as *mut _, value) };
            return ProgramResult::Ok(host_addr);
        }
```

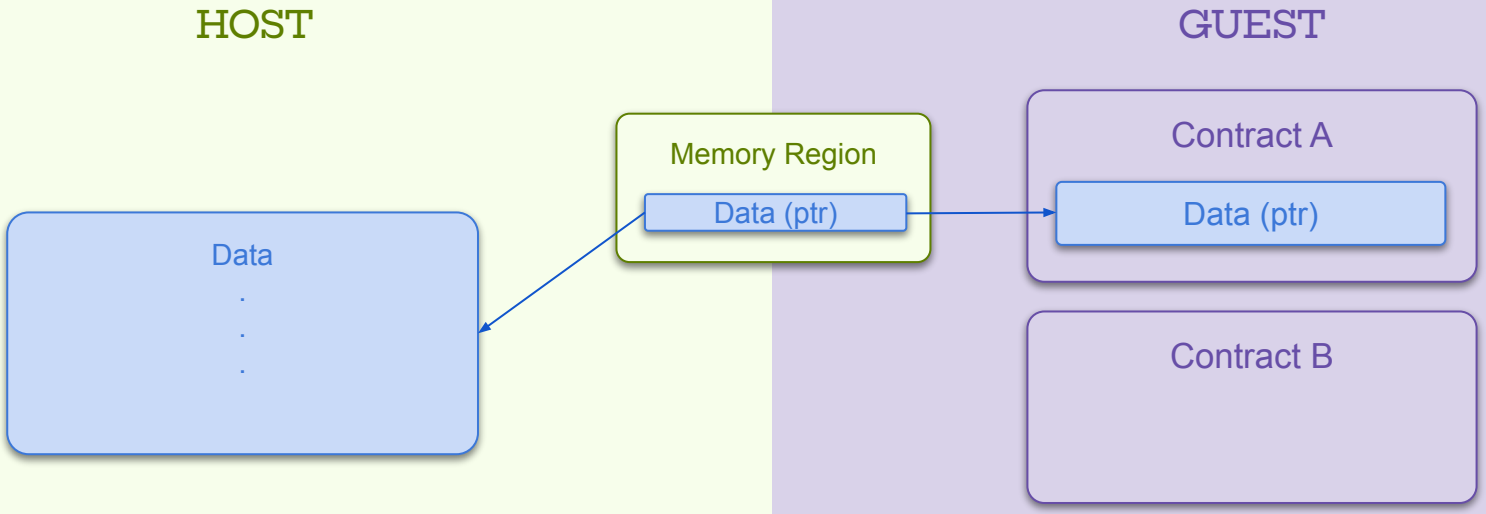
SOLANA VIRTUAL MACHINE

- Calling an external contract is expensive, requiring serializing and copying large chunks of data



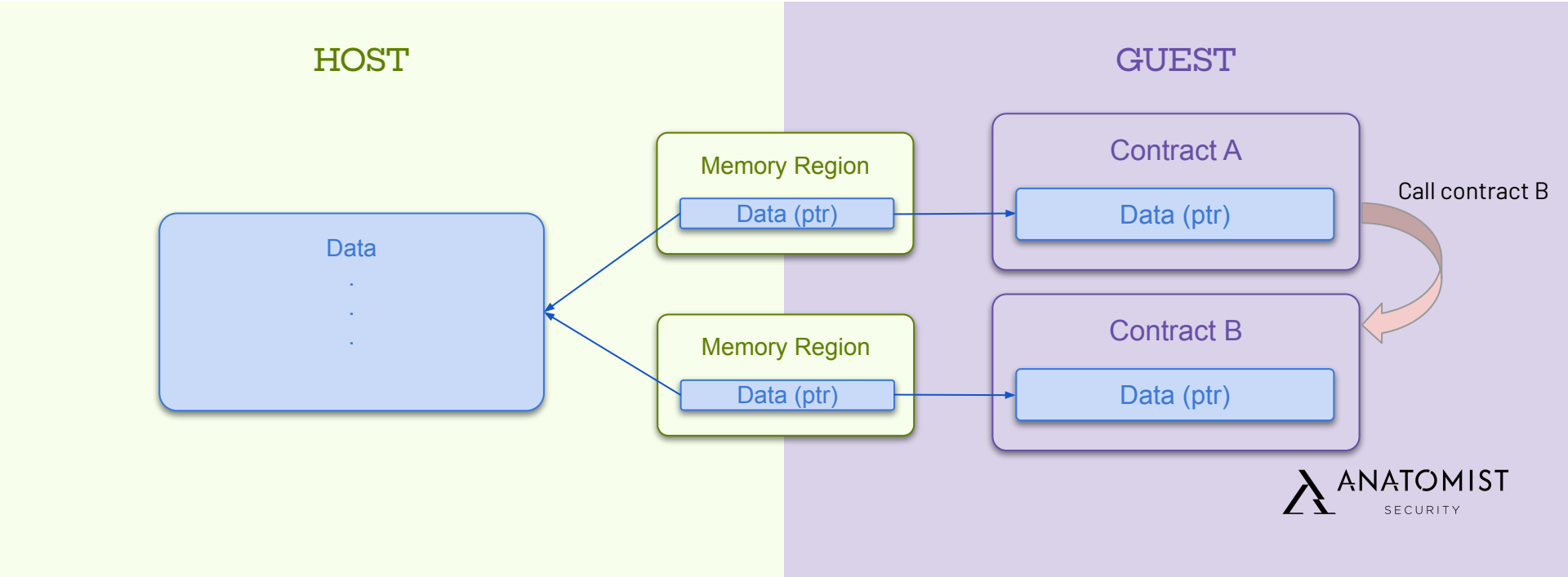
SOLANA VIRTUAL MACHINE

- How about having shared memory in the host?



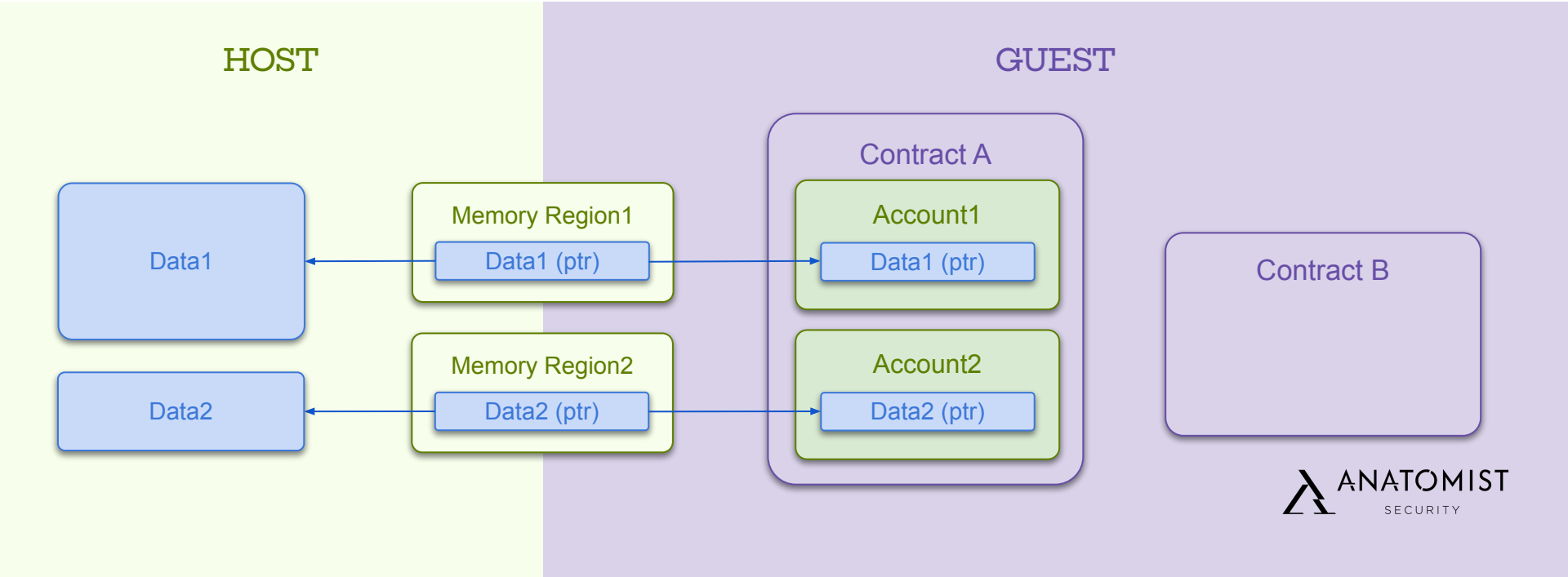
SOLANA VIRTUAL MACHINE

- Add pointer without copying data when calling contract B



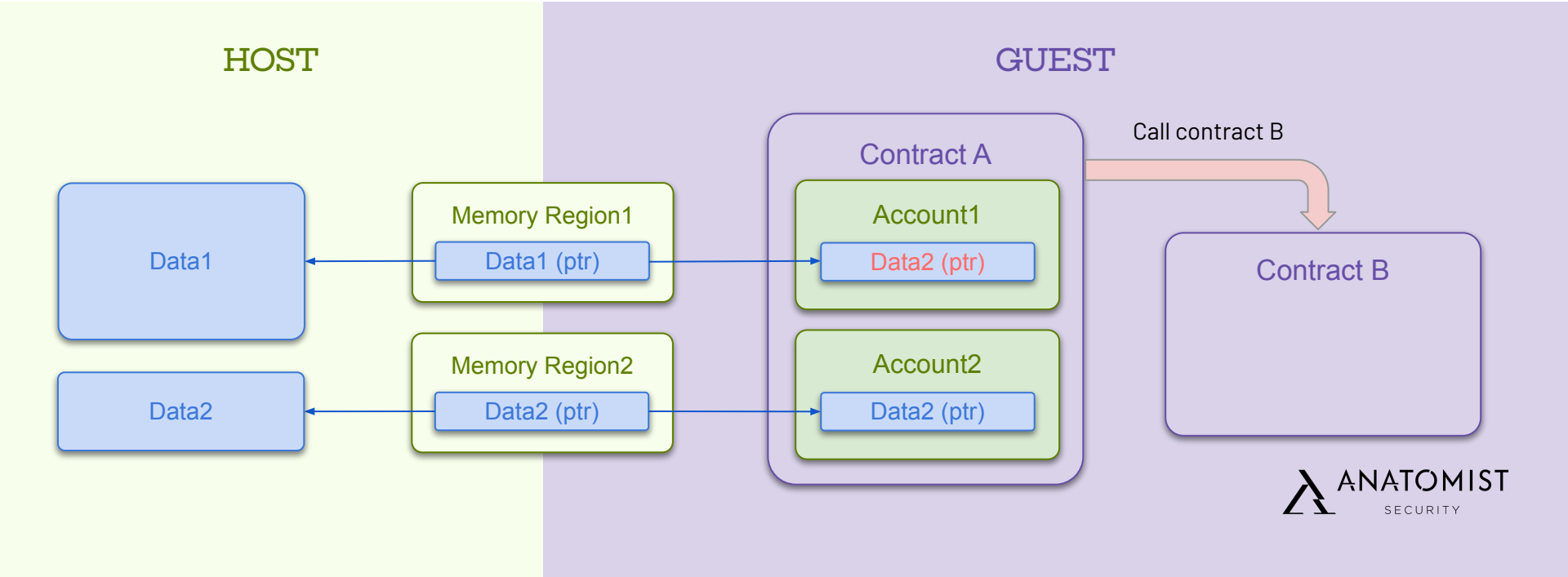
THE PROBLEM

- When returning from contract B, we can mess up the memory mapping



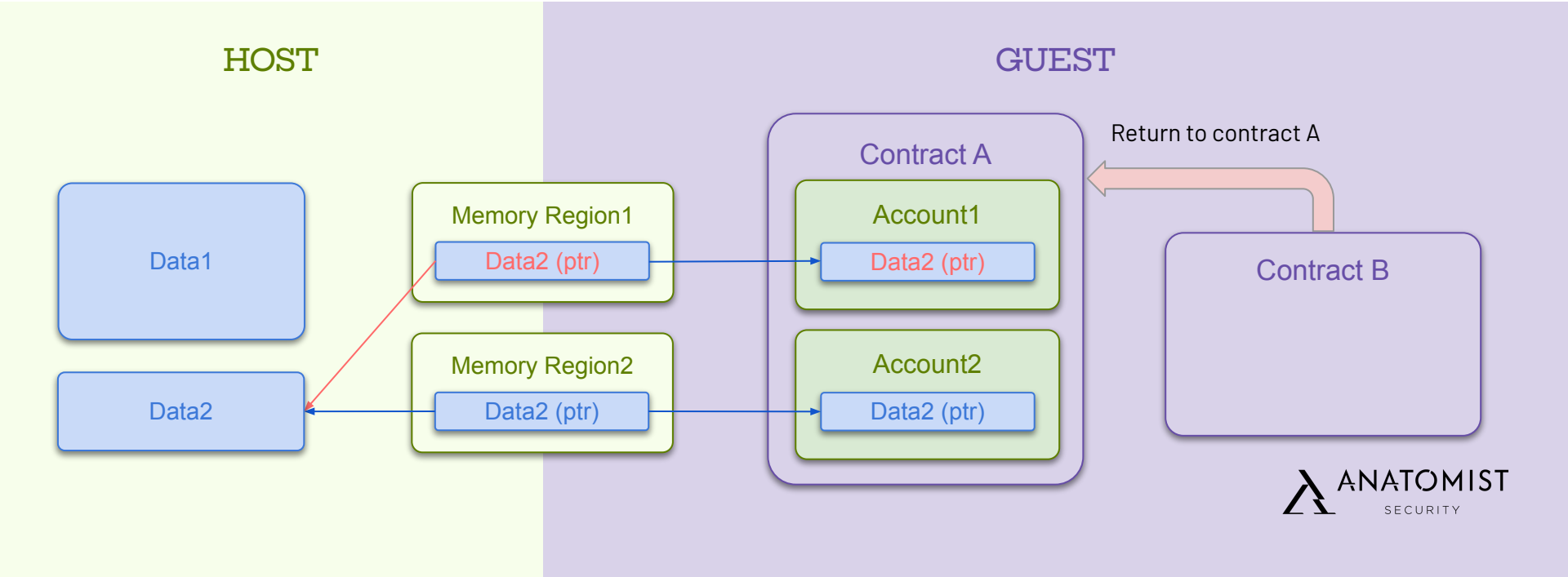
THE PROBLEM

- Before calling contract B, change Account1 data ptr to Account2



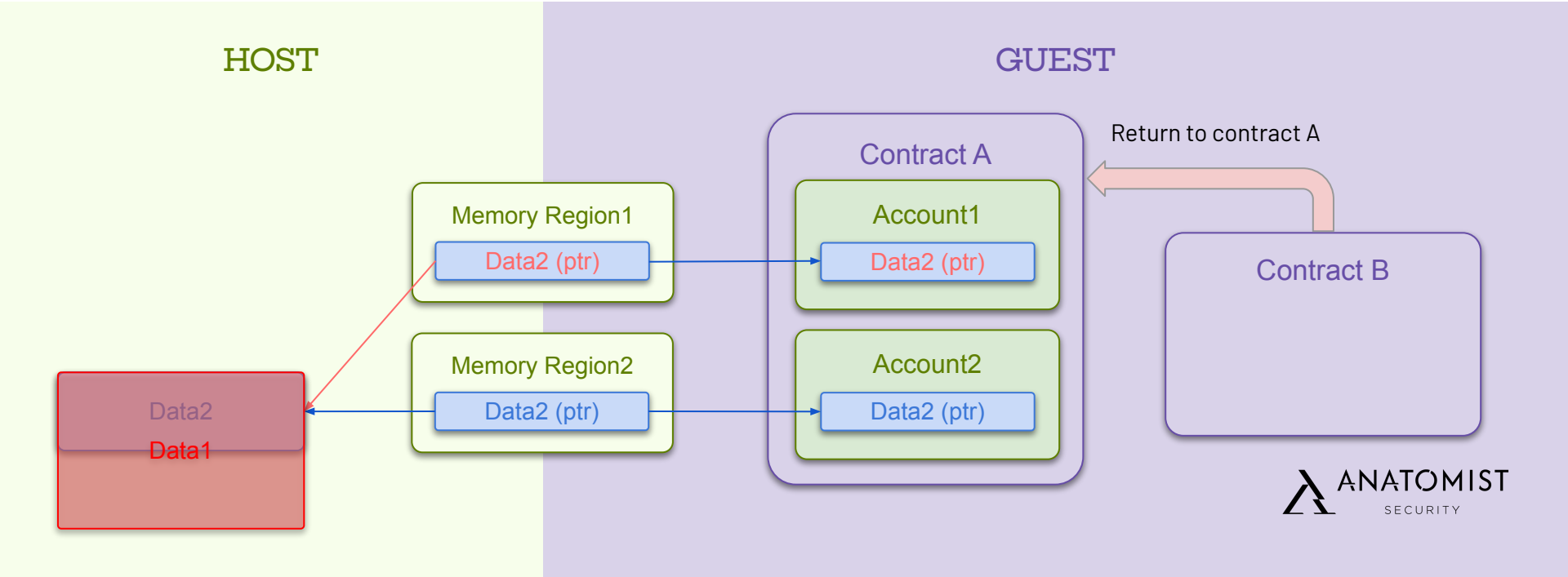
THE PROBLEM

- When returning to contract A, Memory Region1 thinks Account1 updated data → wrongly updates ptr



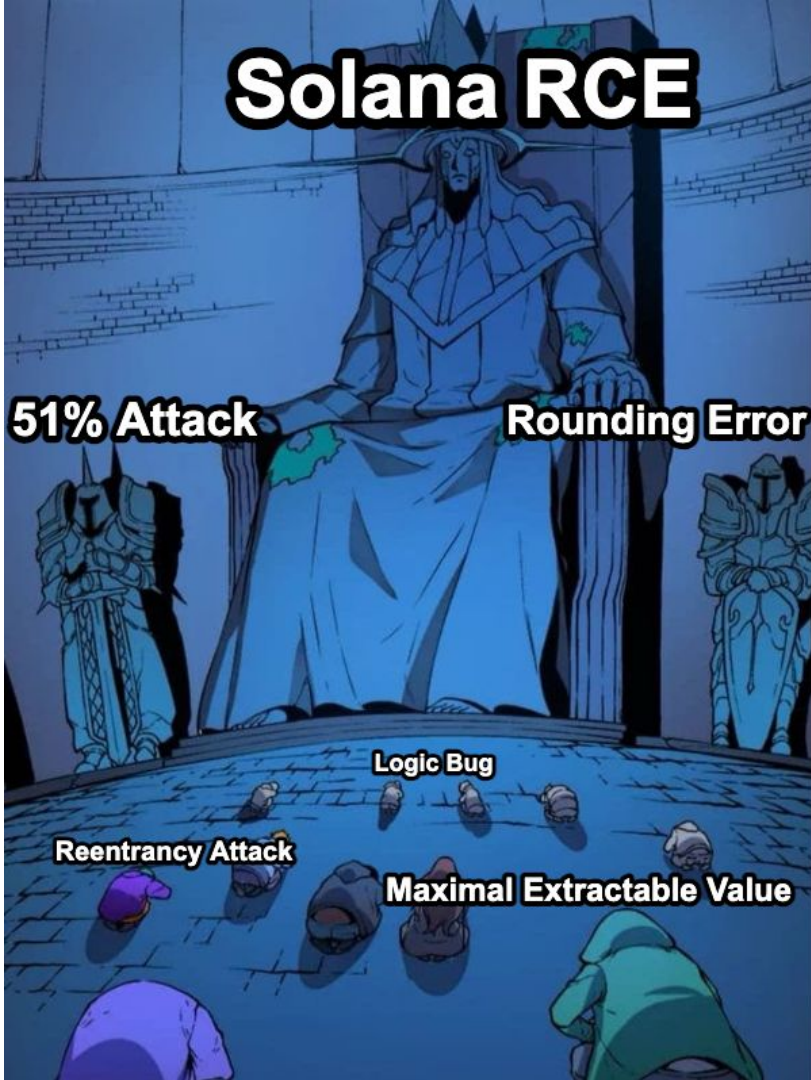
THE PROBLEM

- But Memory Region1 still thinks the length of Data1 is larger → Out of bounds read & write on the HOST



THE IMPACT (that didn't happen)

- Remote Code Execution (RCE) on Solana
- Exploitable with NO preconditions
- Arbitrary manipulation of Solana's $6 Billion TVL
    - Unlimited money printing
    - Balance fabrication



SOLANA

- Swift response & mitigation
- Postponed launching
- Multiple fix & follow-up

→ NO FINANCIAL LOSS

SWAY

- Considered out of scope
- Downgraded the severity
- Didn't patch it

→ SERVICE SHUTDOWN

MORE DETAILS

4/17 (Thu.) 10:10 - 10:40

SHARE

Pwning Blockchain for Fun and Profit: Exploiting an RCE Vulnerability in the Solana validator

Premiere: 4/17 10:10 - 10:40

Replays: 4/17 16:10 - 16:40, 4/17 22:10 - 22:40

While extensive research has been conducted on all kinds of smart contracts, analysis of the underlying infrastructure powering blockchains remains relatively rare, despite its far greater impact. This talk explores a RCE vulnerability in Solana's validator, discovered during its transition to a new runtime optimization in version 1.16.

SPEAKER

Ginoah

Anatomist Security

Co-Founder

TOPIC / TRACK
CYBERSEC GLOBAL 2025: United as One

LEVEL
Advanced

SESSION TYPE
Live Stream Session

LANGUAGE
English

SUBTOPIC
Blockchain
Exploit of Vulnerability
Open Source Security

ANATOMIST SECURITY

05

Consensus

SECURITY CRITERIA

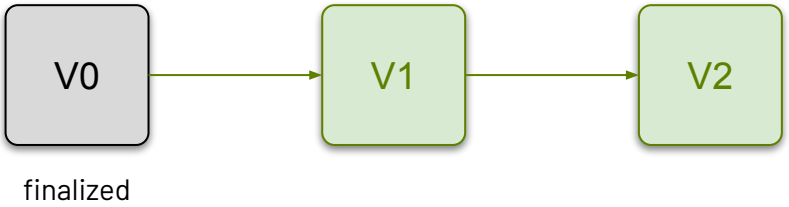
Liveness	• The system always makes progress • Something "GOOD" eventually happens	Ex: Transactions are eventually finalized in a blockchain
Safety	• The system never produces incorrect results • Nothing "BAD" ever happens	Ex: No two nodes commit different blocks at the same height

SECURITY CRITERIA

Liveness	• The system always makes progress • Something "GOOD" eventually happens	Ex: Transactions are eventually finalized in a blockchain
Safety	• The system never produces incorrect results • Nothing "BAD" ever happens	Ex: No two nodes commit different blocks at the same height

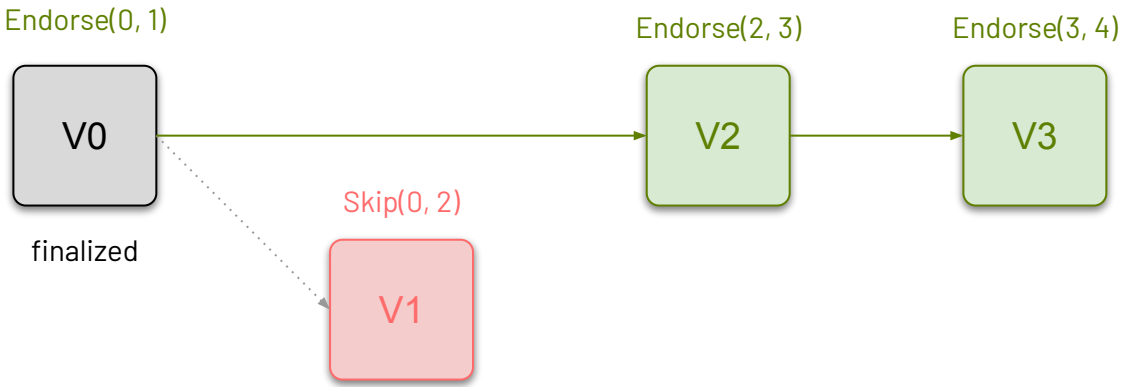
CONSENSUS A

- Consensus reached if more than ⅔ validators agree
- Implicit finalization after 2 consecutive blocks



CONSENSUS A

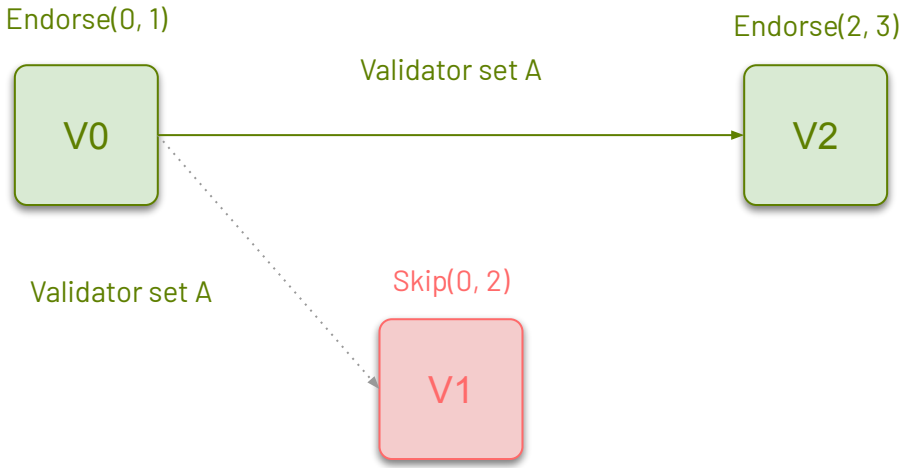
- A validator has 2 actions
  - Endorse(X, X+1)
    - Received block X ⟶ It's safe to build a new block on top of it
  - Skip(X, X+2)
    - Haven't seen X+1 for a while ⟶ Suspect the leader of X+1 has failed



ANATOMIST SECURITY

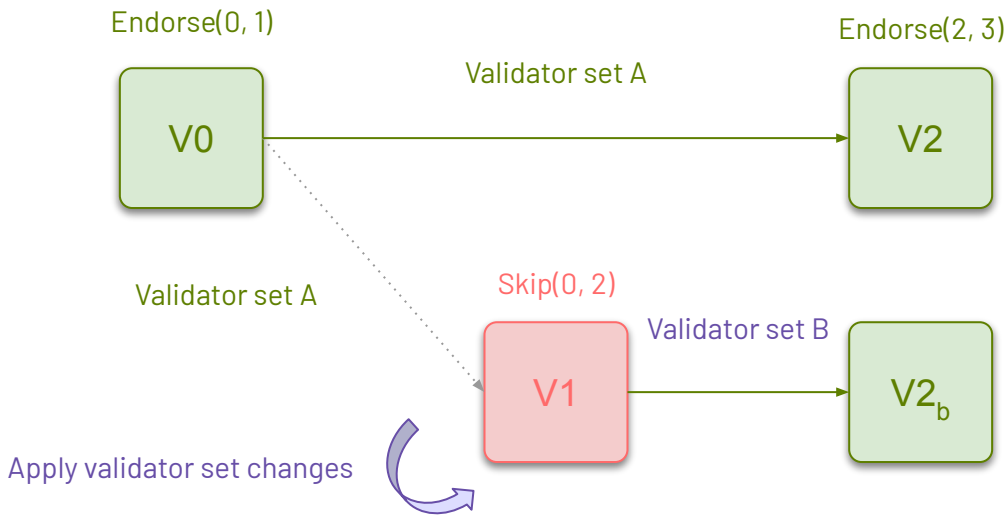
CONSENSUS A

- What if validator set changes?



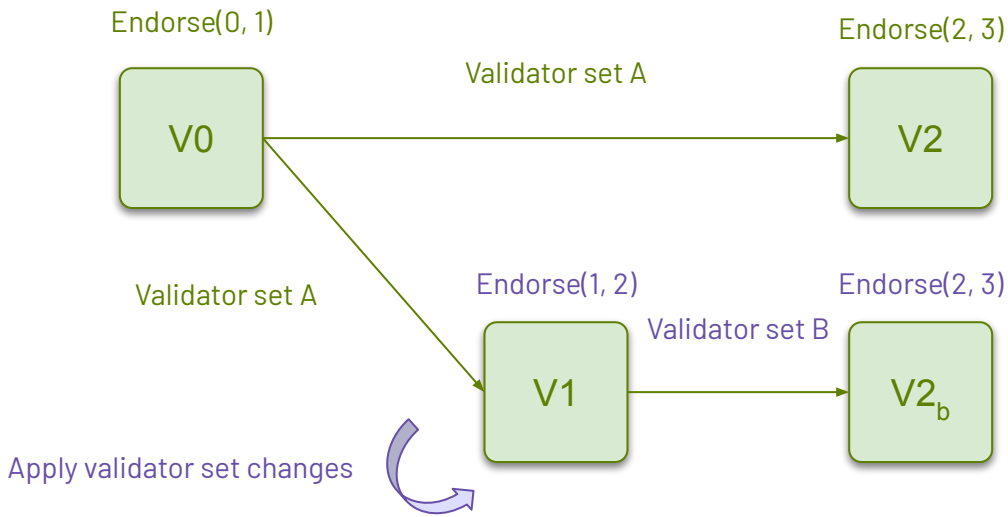
CONSENSUS A

- What if validator set changes?



CONSENSUS A

- What if validator set changes?

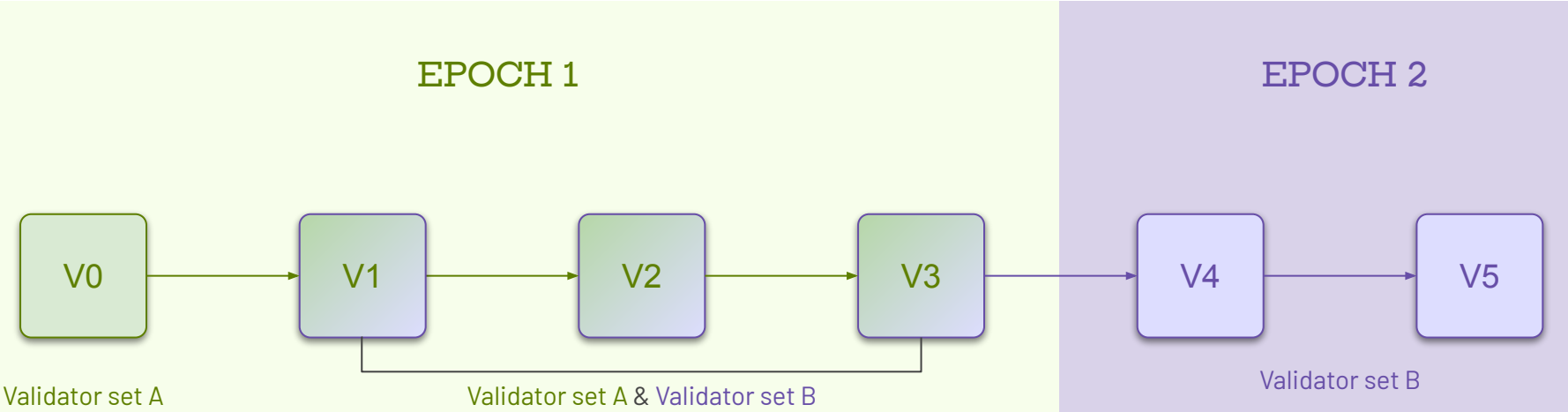


Safety BREAK!

ANATOMIST SECURITY

THE REAL CONSENSUS A

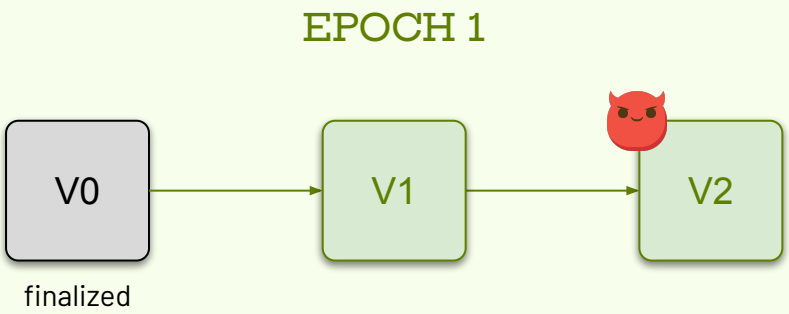
- Switching validators by splitting blocks into epochs
- Both validator sets must vote 3 consecutive blocks before switching



ANATOMIST SECURITY

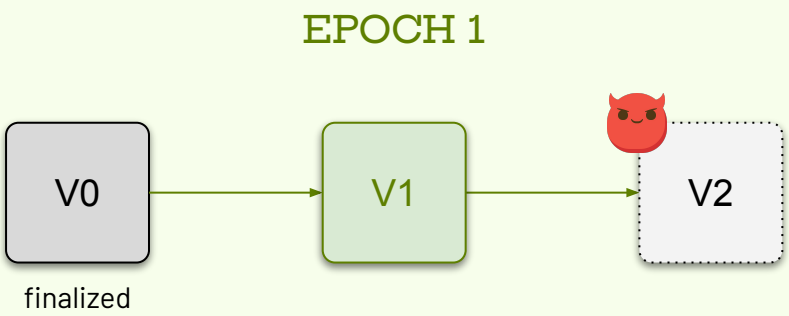
THE PROBLEM

- What if some leaders are evil?



THE PROBLEM

- The leader of V2 choose to WITHHOLD its block, causing other validators to timeout and skip(1, 3)

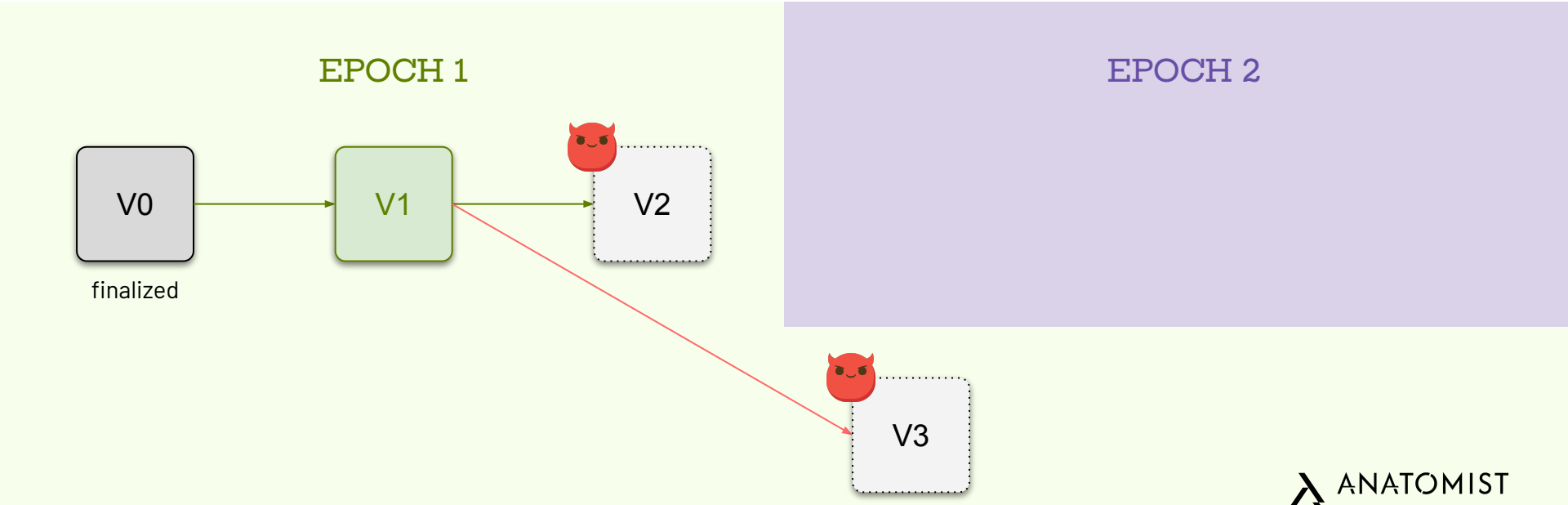


EPOCH 2

ANATOMIST
SECURITY

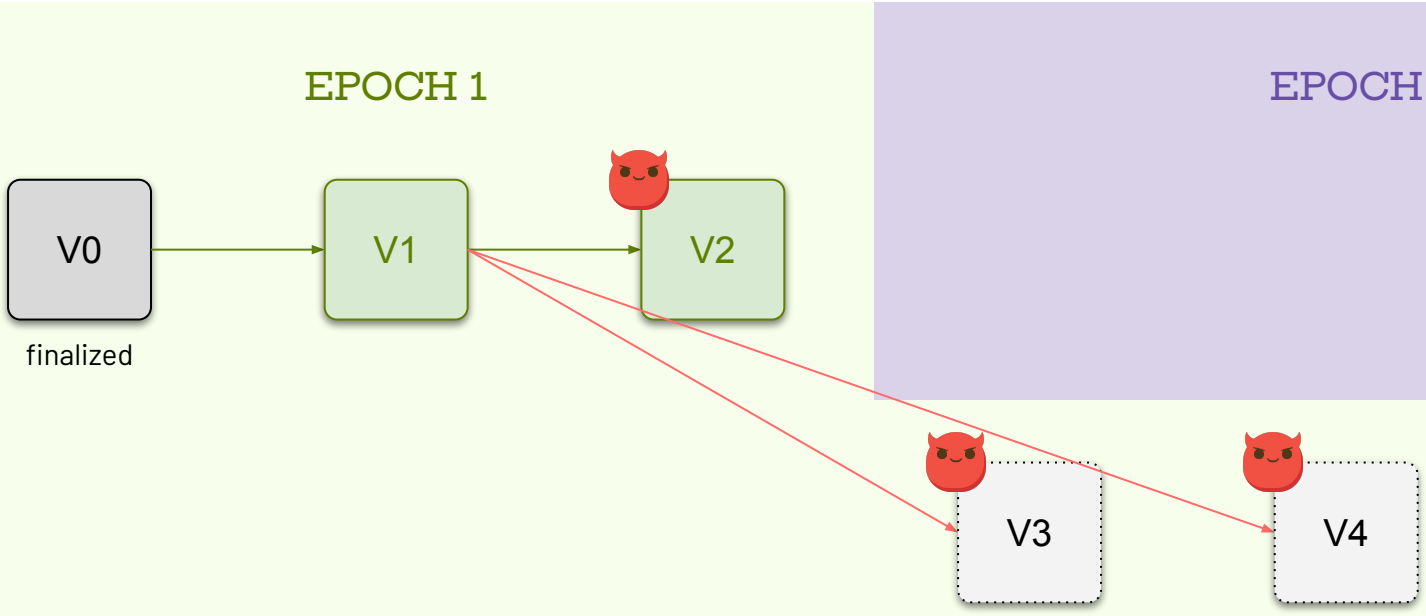
THE PROBLEM

- The leader of V3 also choose to NOT produce a block, causing other validators to timeout and skip(1, 4)



THE PROBLEM

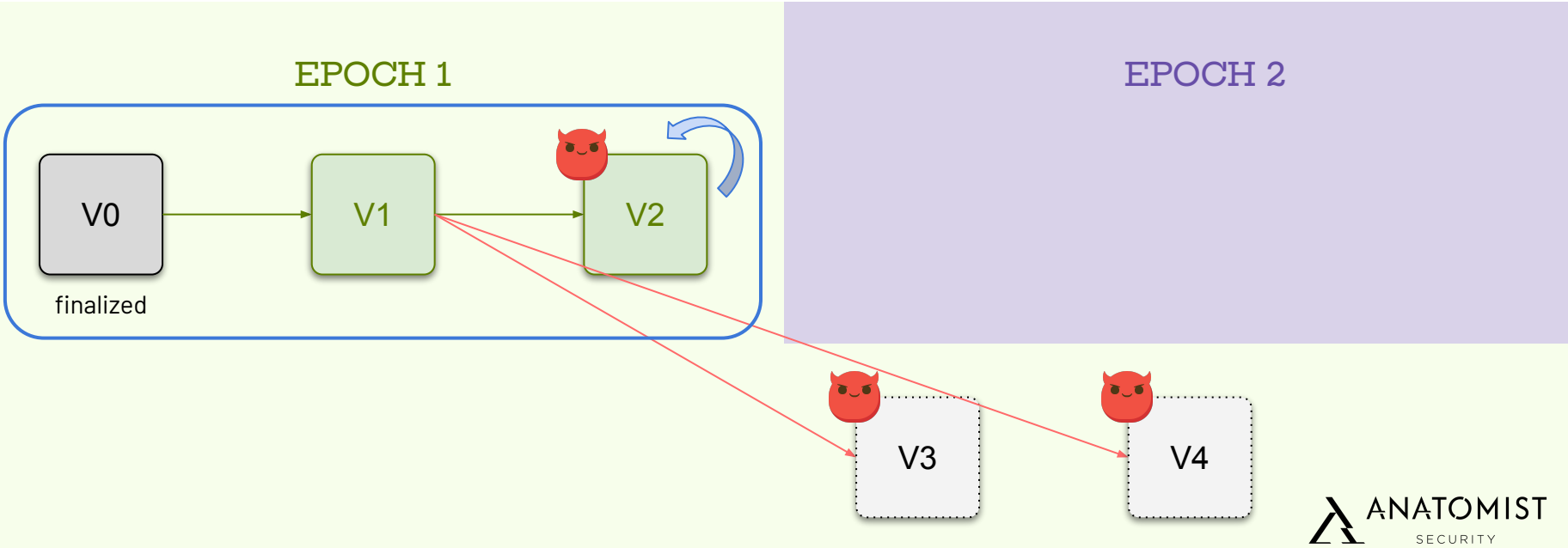
- The leader of V4 withhold its block, while V2 broadcasts its block and gets accepted



ANATOMIST SECURITY

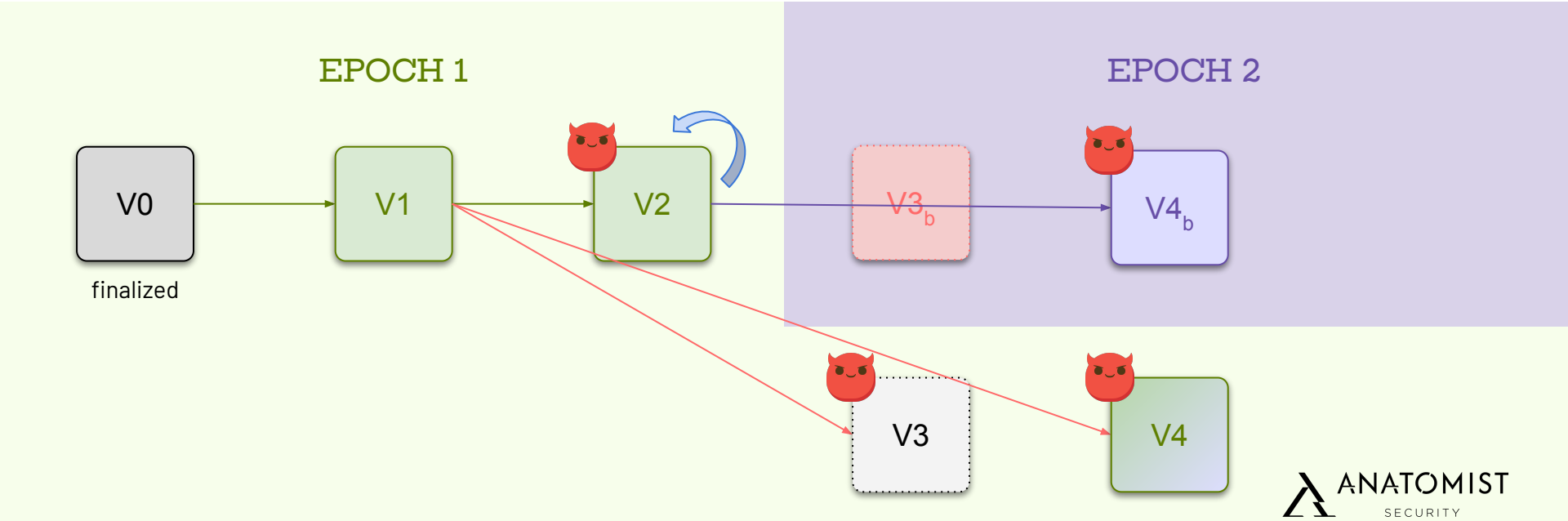
THE PROBLEM

- 3 consecutive blocks accepted, switch validator set!



THE PROBLEM

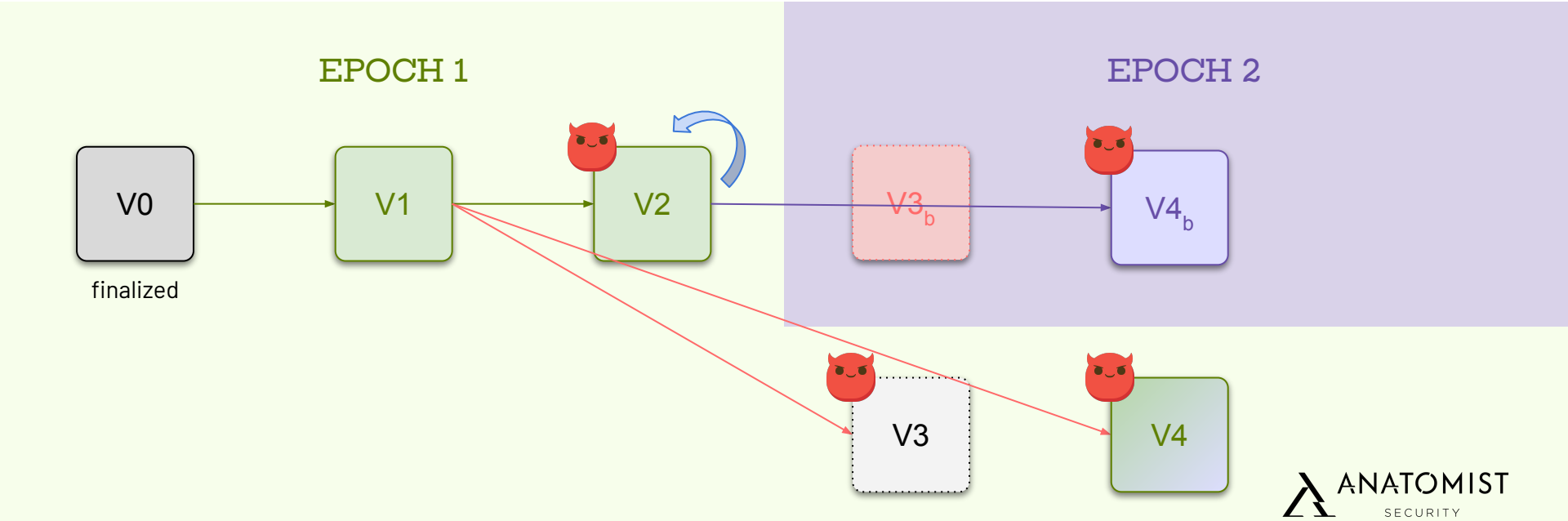
- $V3_b$ will be skipped, V4 and $V4_b$ release their blocks together → validators face a race condition



ANATOMIST SECURITY

THE PROBLEM

- If V4 and $V4_b$ both have ≥ ⅓ endorsement from validator set B, consensus can never be reached



ANATOMIST SECURITY

THE PROBLEM

Liveness	• The system always makes progress • Something "GOOD" eventually happens	Ex: Transactions are eventually finalized in a blockchain

- Validators at the same block height may disagree on the current epoch
  - Some think it's Epoch 1, others Epoch 2 → different voting targets
  - Neither side reaches ⅔ majority → Liveness BREAK

THE IMPACT

- If we have control over 4 leaders or 4 leaders collude
  - we can halt the entire chain

THE IMPACT

- If we have control over 4 leaders or 4 leaders collude
  - we can halt the entire chain

CONCLUSION

We went though:

- 4 major attack surfaces in the Web3 ecosystem
    - Including 3 commonly overlooked: Toolchain, VM, Consensus

For each surface, we explored:

- Key attack techniques
- Real-world case studies
- Impact on live deployments

ANATOMIST SECURITY

Q&A

 @th3anatomist

 kevinwang@anatomi.st

 https://anatomi.st

ANATOMIST SECURITY