BunnyFinder: Finding Incentive Flaws for Ethereum Consensus

Rujia Li[1], Mingfei Zhang[2], Xueqian Lu,
Wenbo Xu[3], Ying Yan[3], Sisi Duan[1]
[1]Tsinghua University
[2]Shandong University
[3]Ant Group


The 33rd NDSS Symposium — February 23–27, 2026, San Diego, California, US

Ethereum

- Ethereum is the second biggest blockchain system.
- In September 2022, Ethereum transitions from Proof-of-Work to a Proof-of-Stake consensus mechanism, Gasper.

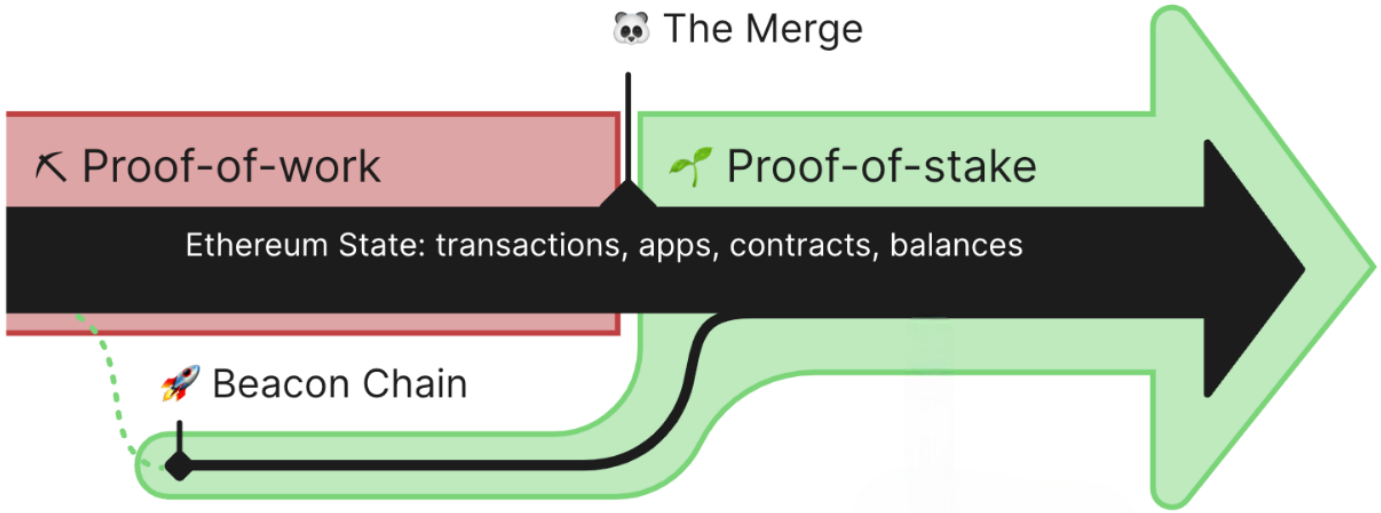


Figure source: https://ethereum.org/en/roadmap/merge/

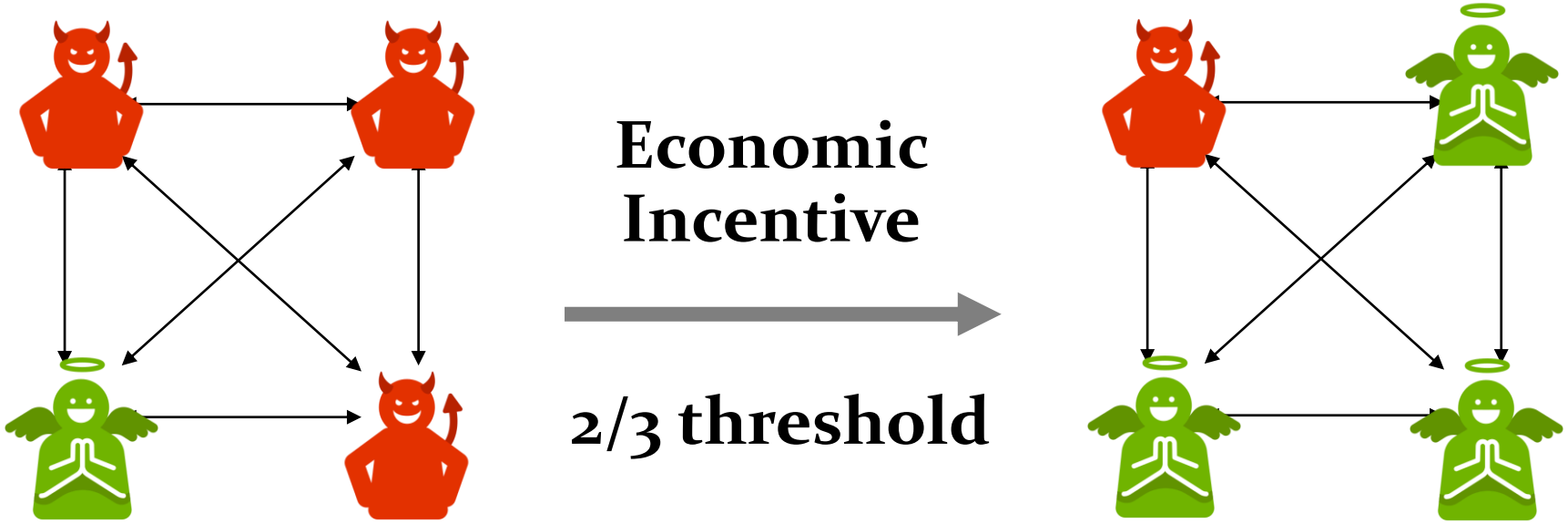
Assumption and Security Goal

Assumption: ≥ 2/3 stakes are controlled by honest validators.

- **Safety**: validators will agree on the same sequence of blocks.
- **Liveness**: transaction submitted by an honest party will eventually be accepted.

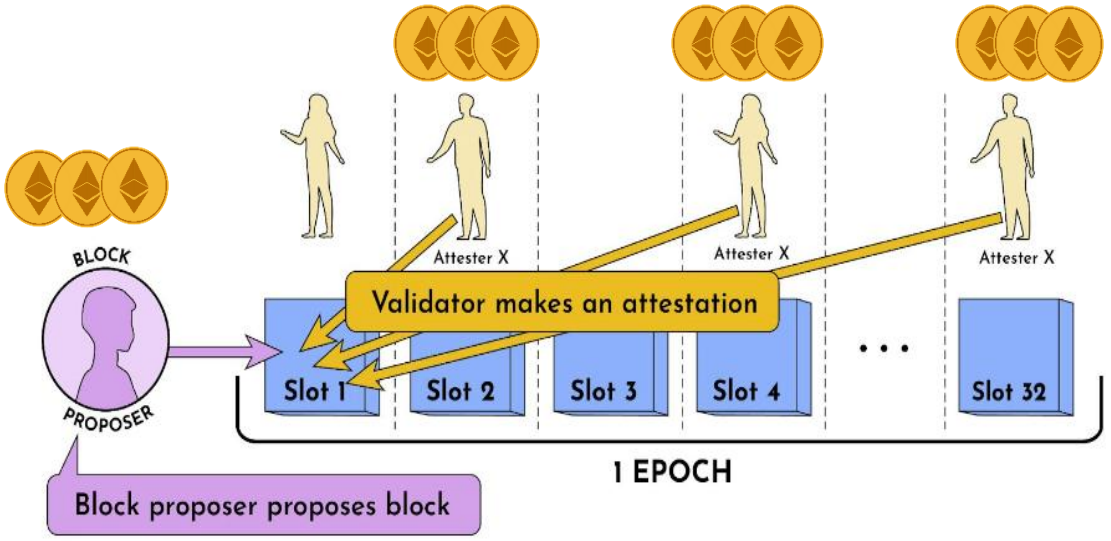
How to keep validators honest 

Assumption Is Incentive-Driven



Using incentives to keep the majority honest

Incentives in Ethereum



Positive Incentives (Rewards)

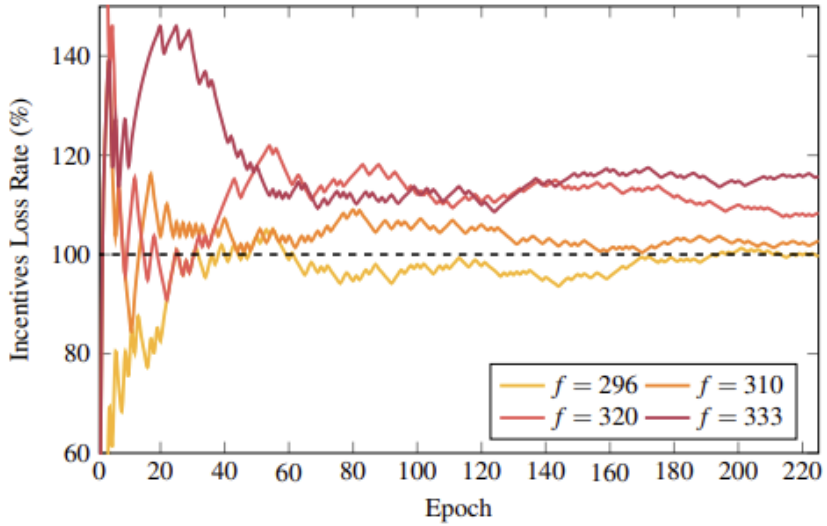
- Attestation rewards
- Block rewards

Negative Incentives (Penalties)

- Slashing for equivocation
- Inactivity penalties

The Incentive Mechanism is Not Perfect

Example: Staircase Attack （USENIX Security 2024）



"Honest validators suffer from penalties, even if they strictly follow the specifications of the protocol"

Zhang, Mingfei, Rujia Li, and Sisi Duan. "Max attestation matters: Making honest parties lose their incentives in ethereum PoS." In 33rd USENIX Security Symposium (USENIX Security 24), pp. 6255-6272. 2024.

Incentive Flaws can Break Consensus

1. Honest validators lose rewards

2. Adversarial validators' stakes grow

3. Threshold weakens

4. Safety/liveness risk
   a) *double spending becomes possible*
   b) *transactions fail to finalize*

Existing Flaw Finding Approaches are Manual

Limitations of existing approaches

- Existing discoveries are manual
- Each attack is crafted for a specific scenario

Scheme	Flaw Type	Attack Strategy		Attack Result		Identified Ethereum Implementations
		Content Manipulation	Order Manipulation	Honest	Byzantine	
Ex-ante reorg attack [5]	I	block	head (short for h)	less reward	reward	Prysm 5.2.0; Teku 25.6.0
Sandwich reorg attack [10]	I	block	head, modify parent	less reward	reward	Prysm 5.2.0; Teku 25.6.0
Unrealized justification attack [11]	III	-	parent	penalty	reward	Prysm 4.0.5
Justification withholding attack [12]	III	block	head	penalty	reward	Prysm 4.0.5
Staircase attack [6]	III	block	source, h target, attestations	penalty	reward	Prysm 4.0.5
Selfish mining attack (this work)	I	block	head, modify parent	less reward	reward	Prysm 4.0.5
Staircase attack-II (this work)	III	block	source, h, target, attestations	penalty	reward	Prysm 5.2.0
Pyrrhic victory attack (this work)	II, IV	blocks, attestations	all	less reward&penalty	penalty	Prysm 5.2.0; Teku 25.6.0

Traditional Testing Approaches Fall Short

Focus on implementation vulnerabilities and bugs.

- **Penetration Testing**
- **Fuzz Testing**
- **Chaos Engineering**

Incentive flaws are:

1. Not code bugs
2. Not protocol violations
3. But rational strategy exploits

Our Research Question

Incentive Flaws Are Dangerous

- Undermine fairness
- Cause safety or liveness failures

Existing Discovery Methods

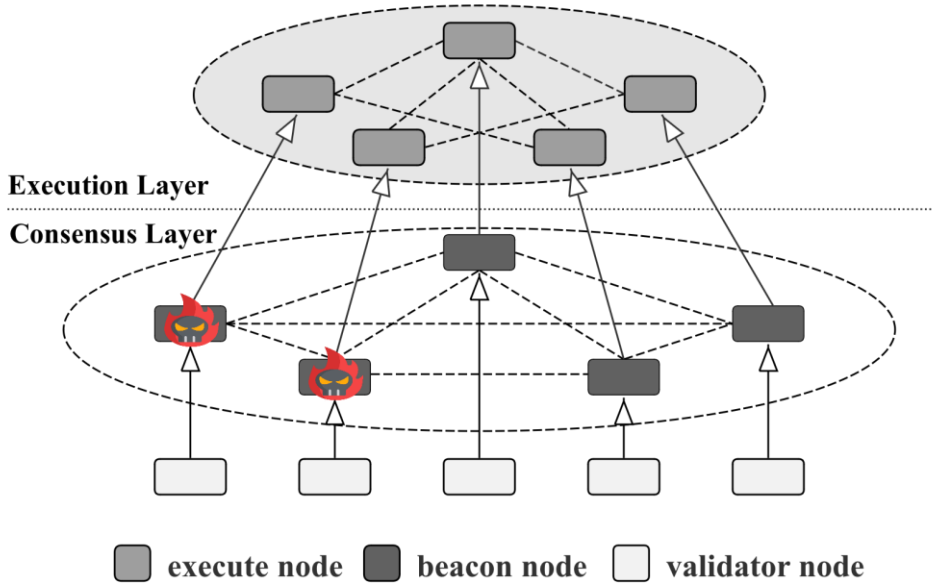
- Rely heavily on expert intuition
- Require manual effort

Can we find the incentive flaws in Ethereum while involving less manual effort?

Our Approach: BunnyFinder

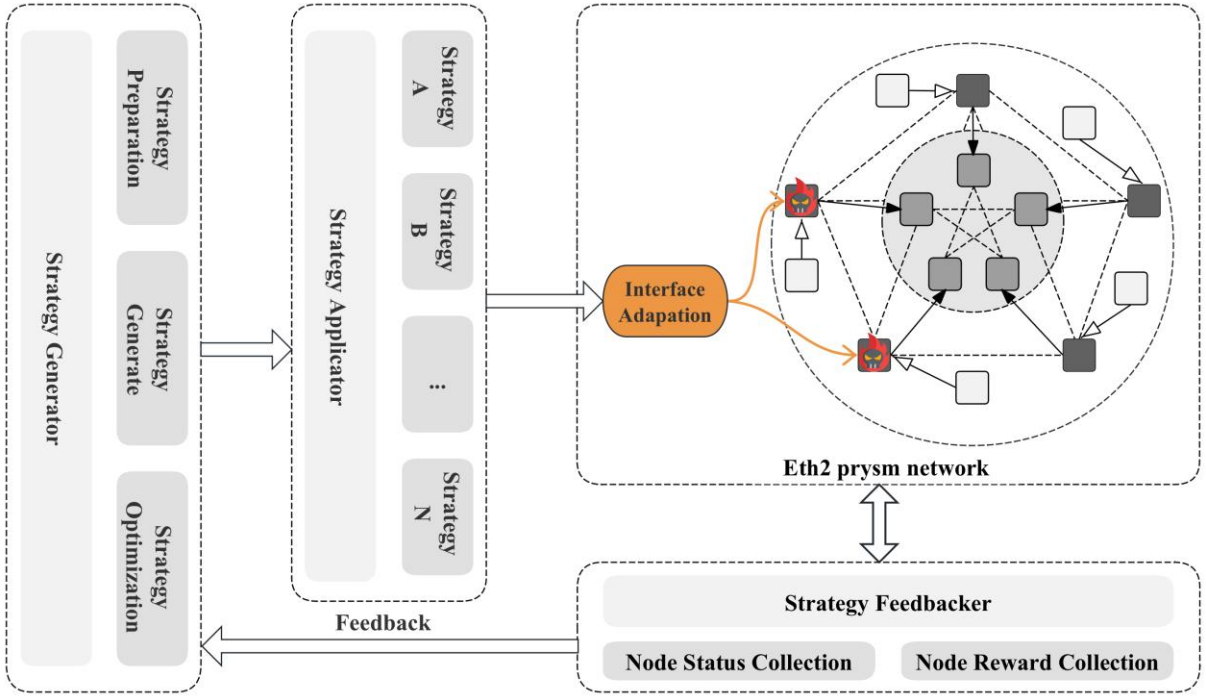
The first framework for finding incentive flaws in Ethereum PoS

1. **Semi-automated** discovery
2. Focuses on **incentive flaws**
3. Feedback-driven optimization for **attack refinement**



Design Philosophy

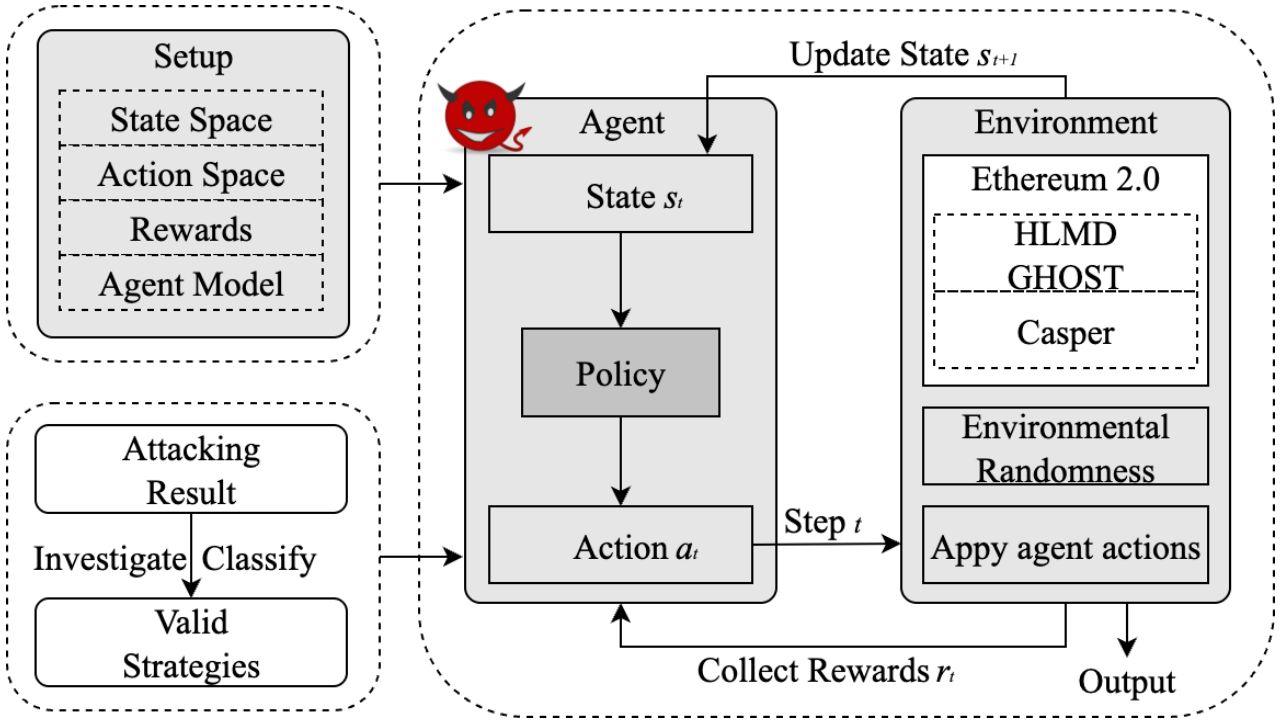
Inspired by failure injection in software testing, instead of injecting faults into code, we **inject strategic behaviors into the protocol**.



- Simulate adversarial behaviors
- Compile into injectable instructions
- Inject instructions into the execution
- Monitor and analyze system responses

Design Philosophy

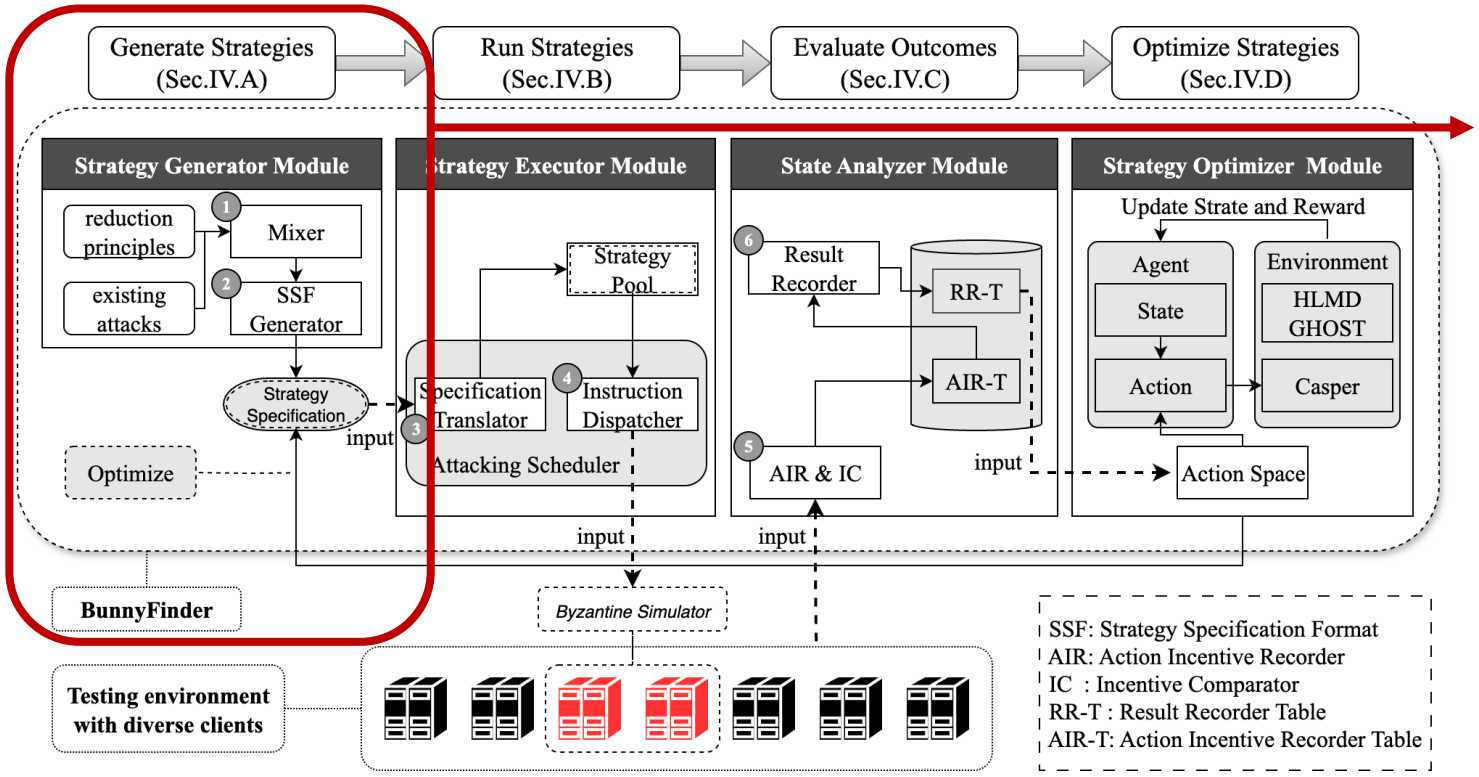
Use Reinforcement Learning to adaptively refine attacks



- Uses Reinforcement Learning to adjust attack timing and parameters (e.g., block withholding) to understand the full exploitability of the flaw.

BunnyFinder Overview

A framework for finding incentive flaws in Ethereum PoS

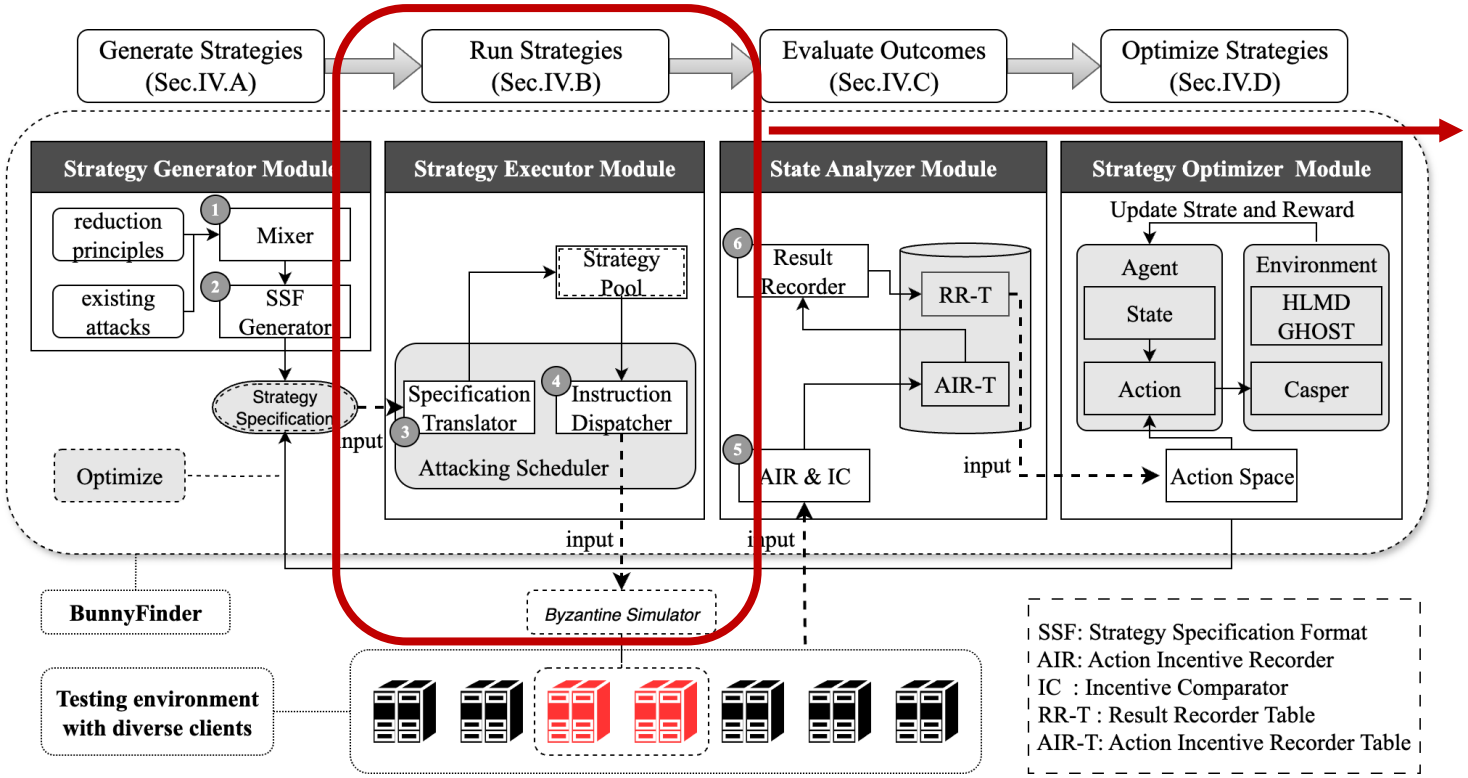


Strategy Generator (SG)

Creates a vast pool of potential strategies by varying parameters such as delay duration.

BunnyFinder Overview

A framework for finding incentive flaws in Ethereum PoS

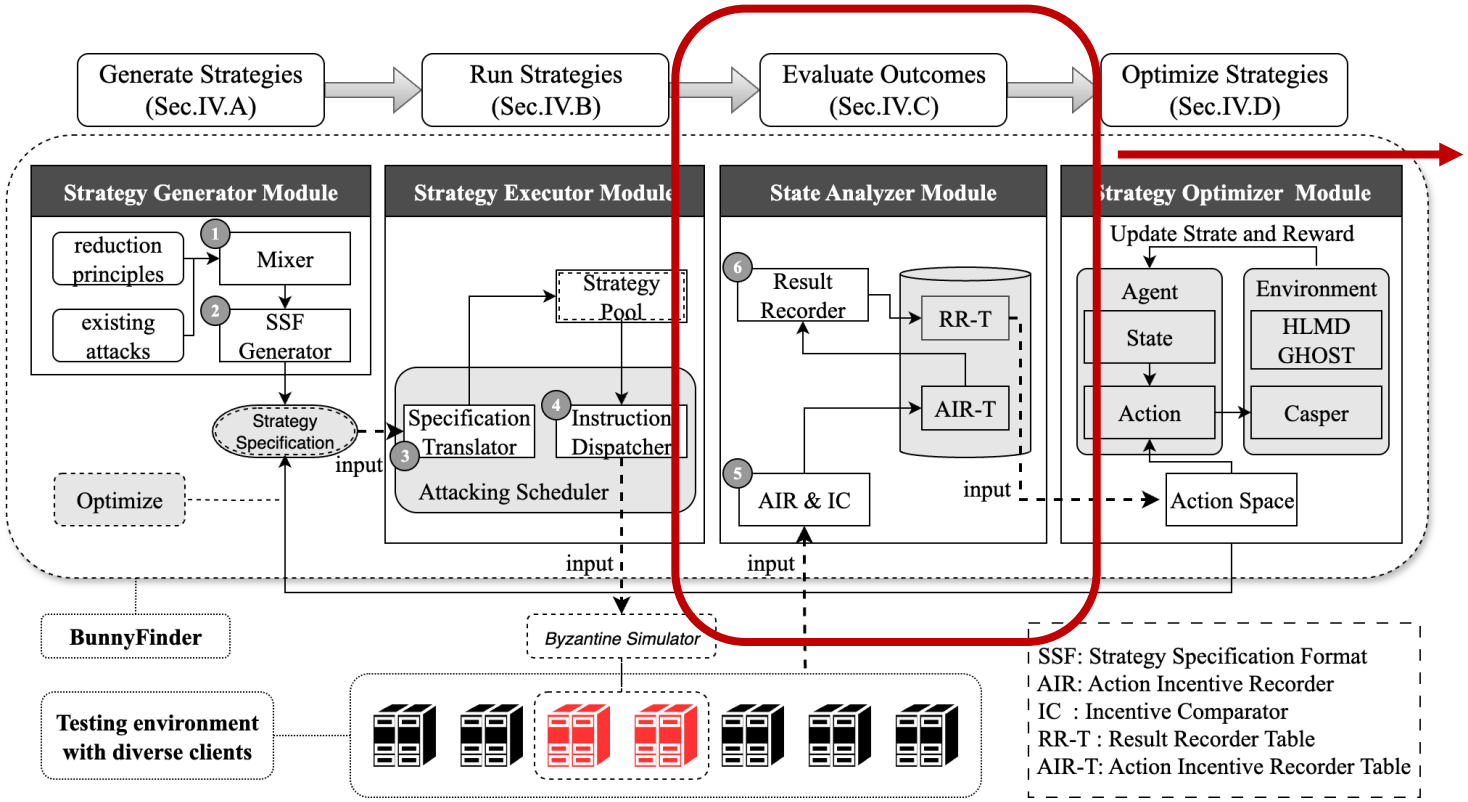


Strategy Executor (SE)

Integrates with Ethereum clients (i.e., Prysm, Teku) to simulate a network with honest and malicious validators, mimicking real-world conditions.

BunnyFinder Overview

A framework for finding incentive flaws in Ethereum PoS

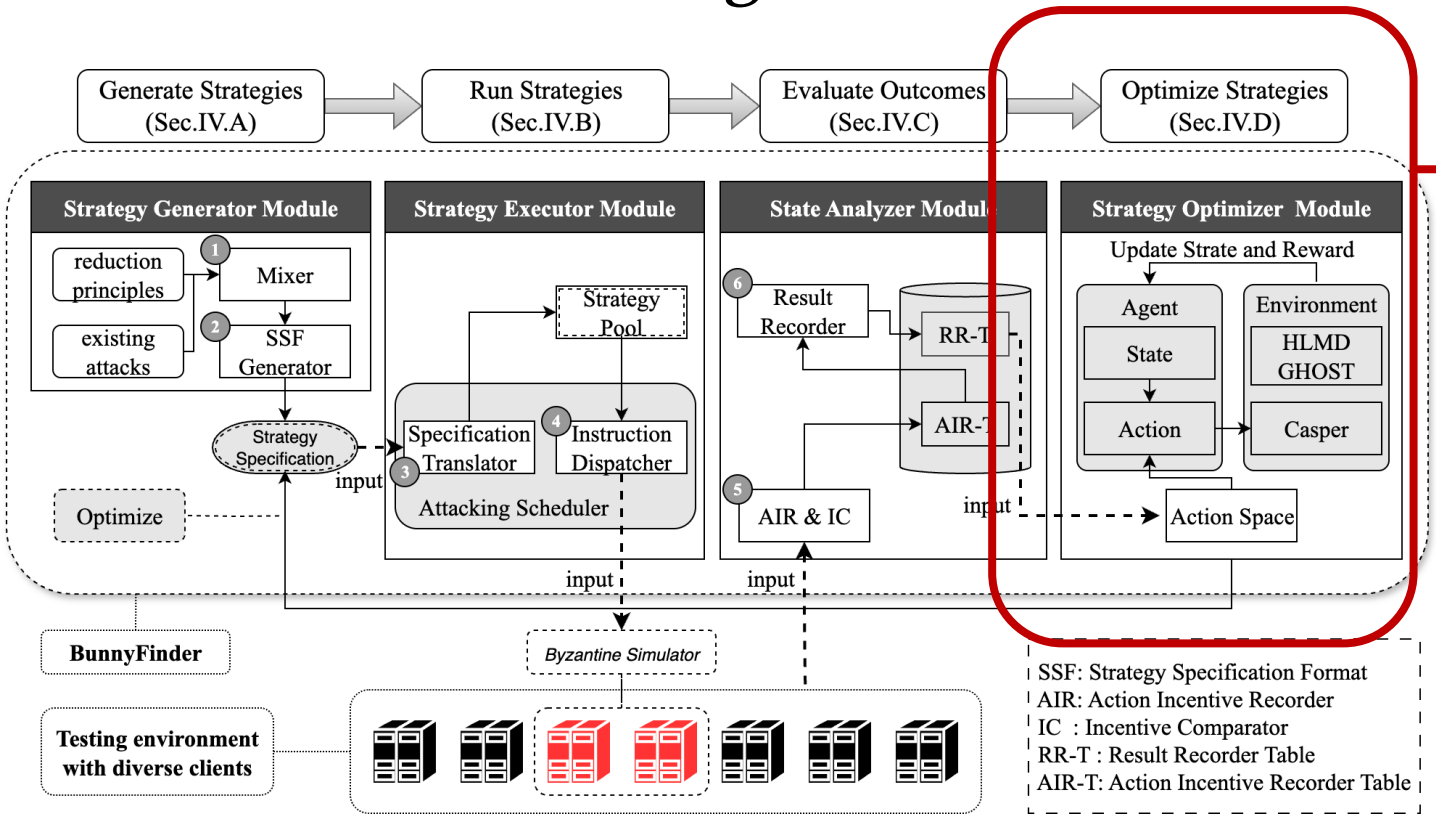


State Analyzer (SA)

Determines if an attack successfully exploited an incentive flaw by examining post-attack state.

BunnyFinder Overview

A framework for finding incentive flaws in Ethereum PoS



Strategy Optimizer (SO)

Takes successful attacks and fine-tunes parameters to maximize their effectiveness and profit potential.

Implementation and Evaluation

1. Deployed BunnyFinder on real Ethereum implementations (Prysm, Teku)
2. Conducted large-scale adversarial experiments in a controlled testnet
3. Explored the attack space with **9,354** injected strategies

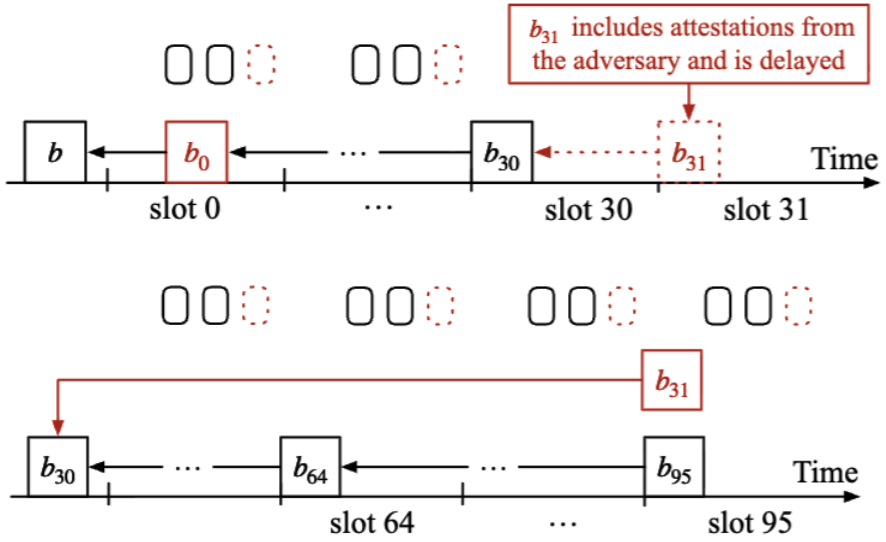
- Project homepage
- User documentation
- Open-source implementation
- Evaluation datasets

9,354	3,121	5	3
Attack Instances Simulated	Incentive Flaws 32.9% identified	Known Attacks Reproduced	New Attacks Discovered

Case Study: Staircase Attack-II

Attack Requirement

Requires the proposers of the **first slot of two consecutive epochs** to be Byzantine.



Key Results

33.3%	1/9
Stake Required	Launch Probability

Impact

All honest validators suffer **penalties** while Byzantine validators continue receiving **rewards**

Delay → Break justification → Trigger reorg → Suffer penalties

Case Study: Pyrrhic Victory Attack

A **pyrrhic victory** means a win that comes at such great cost to the victor that it is nearly equivalent to defeat.

Figure source: https://en.wikipedia.org/wiki/Pyrrhic_victory

In the basic attack, the adversary suffers higher loss than honest validators (not effective).

Basic Attack Results

Honest Validators	Adversary Loss
~67% of fair share	**~180% loss rate**

After optimization, the most effective instance: **adversary loses 5.1%** to make **honest validators lose 19.9%**.

Key Takeaways

- First semi-automated framework for finding incentive flaws in Ethereum PoS
- New attacks discovered with responsible disclosure to Ethereum Foundation
- RL-based optimization significantly improves attack effectiveness

BunnyFinder: Finding Incentive Flaws for Ethereum Consensus

Rujia Li[1], Mingfei Zhang[2], Xueqian Lu,
Wenbo Xu[3], Ying Yan[3], Sisi Duan[1]
[1]Tsinghua University
[2]Shandong University
[3]Ant Group

Thank You