Report for Ethereum Bounty

Summary

TLDR: We propose a novel DoS (denial-of-service) attack that can exacerbate the resources consumed of state modification on Ethereum, causing the block throughput to decrease by at least 54% per block.

Hi! Ethereum team, our research team proposes a new type of DoS attack on Ethereum based on the observation that the gas mechanism [1] can not accurately reflect the actual resource consumption of state modification on Ethereum's state storage structure (i.e., Merkle Patricia Trie (MPT)). The DoS attack can exacerbate the resources consumed by state modification, and decrease the block gas throughput of Ethereum, by manipulating the topology of the MPT. Under a proof of concept (PoC) attack, we confirm that using limited computing resources (such as RTX 3080 GPU and a 10-core R5218 CPU), an adversary can mount the DoS attack to persistently increase the time of state modification by 112% and constantly reduce the block gas throughput (we define it as gas consumed in a block divided by block processing time) by about 54% per block on the Ethereum platform.

Furthermore, we reveal the reason why the gas mechanism can not accurately reflect the actual resource consumption of state modification (Section III). Besides, we elaborate on the design of the DoS attack and details of the PoC attack (Section IV). Looking forward to your feedback.

I. Introduction

The state modification is a series of time-consuming tasks, including maintaining and verifying Ethereum's state data in its storage structure (i.e., Merkle Patricia Trie (MPT)), which involves the consumption of CPU computation, memory operation, and loading and saving with disks. Inspired by our observation that the gas mechanism [1] is unable to accurately reflect the actual resource consumption of state modification, we propose a new form of DoS attack by manipulating the topology of the MPT to exacerbate the consumed resources involved in state modification.

The core idea of our DoS attack is to deepen the depth of the MPT by inserting leaf nodes in the MPT, i.e., nodes located at the ends of branches of the MPT. Hence, our DoS attack will lead Ethereum to maintain and verify more state data in its MPT, and we give an example of how a leaf node causes more consumed resources in Fig. 4. Specifically, each node in the MPT is located by the keccak256 [2] value (defined as $h_{keccak256}$) of its preimage. In detail, the preimage [3] of a node is the data for specifying the node, e.g., a slot of a contract's storage, and an account's address. As the node of the MPT locates by the prefix of $h_{keccak256}$, the DoS attack will search the preimage to collide $h_{keccak256}$ with the specific prefix. As a result, attackers can insert a leaf node by constructing its preimage with a specific prefix.

However, a dilemma arises when constructing leaf nodes in the DoS attack. Specifically, constructing deeper leaf nodes with a longer keccak256 perimage can lead to increased resource consumption, but, it also requires more computing resources for attackers. Conversely, constructing shallower leaf nodes with a shorter keccak256 perimage will cause limited resource consumption. Although, it also needs fewer computing resources for attackers. In our proof-of-concept evaluation, with computing resources such as an RTX 3080 GPU and a 10 core R5218 CPU, we have confirmed that an adversary can persistently increase the time of state modification by 112% and constantly reduce the block gas throughput by about 54% per block on the Ethereum blockchain. Please note that we do not try to find the best trade-off to maximize the impact of our DoS attack, and we believe that the attack impact can be further exacerbated.