POSTER: As Strong As Its Weakest Link: How to Break Blockchain DApps at RPC Service

Kai Li* Jiaqi Chen* Xianghong Liu* Yuzhe Tang* ✉ XiaoFeng Wang† Xiapu Luo‡
* Syracuse University. Emails: {kli111,jchen217,xliu167,ytang100}@syr.edu
† Indiana University Bloomington. Email: xw7@indiana.edu
‡ Hong Kong Polytechnic University. Email: csxluo@comp.polyu.edu.hk

***Abstract*—Modern blockchains have evolved from cryptocurrency substrates to trust-decentralization platforms, supporting a wider variety of decentralized applications known as DApps. Blockchain remote procedure call (RPC) services emerge as an intermediary connecting the DApps to a blockchain network. In this work, we identify the free contract-execution capabilities that widely exist in blockchain RPCs as a vulnerability of denial of service (DoS) and present the DoERS attack, a Denial of Ethereum RPC service that incurs zero Ether cost to the attacker.**

To understand the DoERS exploitability in the wild, we conduct a systematic measurement study on nine real-world RPC services which control most DApp clients' connection to the Ethereum mainnet. In particular, we propose a novel measurement technique based on orphan transactions to discover the previously unknown behaviors inside the blackbox RPC services, including load balancing and gas limiting. Further DoERS strategies are proposed to evade the protection intended by these behaviors.

We evaluate the effectiveness of DoERS attacks on deployed RPC services with minimal service interruption. The result shows that all the nine services tested (as of Apr. 2020) are vulnerable to DoERS attacks that can result in the service latency increased by $2.1X \sim 50X$. Some of these attacks require only a single request. In addition, on a local Ethereum node protected by a very restrictive limit of 0.65 block gas, sending 150 DoERS requests per second can slow down the block synchronization of the victim node by 91%.

We propose mitigation techniques against DoERS without dropping service usability, via unpredictable load balancing, performance anomaly detection, and others. These techniques can be integrated into a RPC service transparently to its clients.

I. Introduction

With the advent of operational blockchains, decentralized applications (*DApps*) running atop these systems are gaining popularity, providing decentralized finances (DeFi), online gaming, information-security infrastructures, etc. A typical DApp is architectured in three layers: *DApp clients* running inside web browsers send requests to a *Remote Procedure Call* (*RPC*) service that translates the clients' requests to cryptocurrency transactions or queries to a *blockchain P2P network*. Such a RPC service operates on a blockchain full node maintained directly by the DApp owner (i.e., an in-house RPC node) or a set of nodes hosted by a third party (i.e., a third-party RPC service) intended to ease DApp deployment. Given the ever-growing blockchain states (e.g., 130 GB and 1.8 TB for a fully synced and an archived Ethereum node, respectively, as of 2018), the RPC service plays an increasingly important role in the DApp ecosystem, scaling DApp clients to low-end mobile devices and web browsers. Major blockchains today flock to roll out RPC supports, which spawn a good number of services in practice, including nine service providers (as is evaluated in this work) supporting the Ethereum's JSON-RPC interface [7], blockchain.info [2] with Bitcoin's JSON-RPC [1], dfuse.io [6] and greymass.com [14] with EOSIO's Chain API [3], stellar.org [16] with Stellar Horizon [10], etc. These services host the majority of DApps; for instance, at least 63% of Ethereum based DApps use one RPC service [5].

Despite its importance, the RPC service is less decentralized (one to hundreds of nodes) than the blockchain network (of tens to hundreds of thousands of nodes) and therefore could become a single point of failure should a denial of service (DoS) attack happen, which could lead to the collapse of the whole DApp ecosystem. It is important to note that DoS is known to pose a significant threat to the blockchain ecosystem, particularly in Bitcoin exchanges and mining pools [21], [19]. We believe such an attack can also be launched against a victim RPC service, allowing a service competitor to steal customers from the victim. Besides, the perpetrator who denies a RPC service can illicitly manipulate the transaction order of a financial DApp [18], [17] and gain profit. For instance, in an auction for registering an Ethereum domain [8] or purchasing a CryptoKitty [4], a bidder can delay others' bidding transactions through denying the RPC service they use to win the auction at an unfairly low price. As another example, a client depositing to a hash-time-lock contract (HTLC), as widely used in blockchain applications (e.g., atomic intra-chain or cross-chain swaps [20] and payment channels [13], [22]), can defer the withdrawal of the deposit after the expiration of the lock (so-called grieving [18]) by denying the RPC service used by the withdrawal, thus retaining the deposit. With such significant implications, this security risk, however, has never been studied before.

Menace of DoERS. Our research shows that indeed this risk is realistic and serious: today's RPC services are vulnerable and can be easily disabled by a new type of DoS attacks (which we call *DoERS* or Denial of Ethereum Rpc Service) that exploit the free execution capability they expose. More specifically, blockchain systems support the Gas-free execution of a smart contract on an individual RPC node, such as Ethereum's

`eth_call` RPC [11] (and `eth_estimateGas` [12] running the same code path). An `eth_call` can be triggered to run any smart contracts including those reading and/or updating their states. Unlike the transaction-triggered smart contract execution, the `eth_call`-triggered execution occurs locally on the recipient RPC node, and its state update, if any, will not be propagated to or reflected in the global blockchain state. The purpose of such a capability is to enable a variety of real applications in pre-production contract testing (e.g., estimating the smart-contract cost before DApp deployment by `estimateGas`), "stored-procedure" like database analytics on blockchain and decentralized financial analysis, and others. Most importantly, `eth_call` is often free. Therefore, the adversary can deploy on the blockchain an attack smart-contract involving a resource-consuming procedure (e.g., an infinite loop of hashing computations) and then trigger it through `eth_call`. This attack is shown in our research to effectively stop a node from performing critical operations for all DApps it hosts, including block/transaction synchronization, serving RPC requests, etc.

Notably, DoERS is different from other DoS attacks on the blockchain network, as studied in the prior research [24], [15], [9], [23]. First, it aims at disrupting the communication channel between a blockchain and its DApps by blocking third-party RPC services, not taking down the blockchain itself as the other attacks do. Second, our attack exploits a unique weakness – Gas-free contract execution on RPC-enabled Ethereum nodes, while existing DoS attacks seek underpriced instructions for attacking replicated smart-contract execution [24], [15], [9] or misusing mining mechanisms [23].

The attack is non-trivial to carry out. We need to overcome the protection already in place on each Ethereum node, such as limiting each call's Gas and time, through strategically delivering continuous queries at an alarmingly low rate below the victim's rate limit. Also it is less clear how extensively RPC interfaces are open to the public on the nodes operated by the DApp owners. Even more challenging is the use of third-party RPC services, which typically run a load balancer in front of RPC nodes. Such a balancer hides the node(s) serving a specific DApp and spread out its clients' requests using undisclosed strategies. Understanding how it works is critical to the success of an attack targeting a specific DApp or a specific client of the DApp. For this purpose, we performed an analysis and measurement study on Ethereum.

Contributions. The contributions of the paper are outlined as follows:

• *New attack*. We identify a new denial of service weakness in today's blockchain, showing that the widely existing free query calls enable a potential resource depletion attacks on RPC services, a weakest link of the DApp ecosystem. Also we implemented the attack on Ethereum incurring zero Ether costs and demonstrated the real-world impact of the threat across leading RPC services.

• *New understanding*. We performed a systematic measurement study on the nine leading RPC services that control the connection between most DApp clients and the Ethereum network. Our measurement on leading RPC services' load balancers, using a novel orphan-transaction based prober, has brought to light the hidden strategies they take, which enables targeted attacks on DApps and clients they serve.

• *Mitigation*. We also studied the potential mitigation on the new threat, identifying a few promising solutions, including the ones that selectively penalize the DApps or clients consuming a large amount of resources on a node.

REFERENCES

[1] Api reference (json-rpc). https://en.bitcoin.it/wiki/API_reference_(JSON-RPC).

[2] Blockchain explorer. https://www.blockchain.com/explorer.

[3] Chain api — eosio developer docs. https://developers.eos.io/manuals/eos/latest/nodeos/plugins/chain_api_plugin/api-reference/index#operation/get_account.

[4] Cryptokitties: Collect and breed digital cats! https://www.cryptokitties.co/.

[5] Dapp survey results 2019. https://medium.com/fluence-network/dapp-survey-results-2019-a04373db6452.

[6] Eosio on dfuse. https://docs.dfuse.io/guides/eosio/.

[7] Ethereum json rpc. https://ethereumbuilders.gitbooks.io/guide/content/en/ethereum_json_rpc.html.

[8] Ethereum name service.

[9] Geth nodes under attack again (reddit). https://www.reddit.com/r/ethereum/comments/55s085/geth_nodes_under_attack_again_we_are_actively/.

[10] Go sdk — stellar developer — rest api. https://www.stellar.org/developers/horizon/reference/index.html.

[11] Json-rpc in ethereum wiki (eth_call). https://github.com/ethereum/wiki/wiki/json-rpc#eth_call.

[12] Json-rpc in ethereum wiki (eth_estimategas). https://github.com/ethereum/wiki/wiki/json-rpc#eth_estimateGas.

[13] Ligntning network, scalable, instant bitcoin/blockchain transactions.

[14] Public apis on greymass, an eosio block producer. https://greymass.com/en/apis.

[15] Transaction spam attack: Next steps. https://blog.ethereum.org/2016/09/22/transaction-spam-attack-next-steps/.

[16] Raphael Lefbvre Renaud Larsen Cathy Pill, Sarah Levin Weinberg. Stellar: Smart influencer marketing platform. stellar.io.

[17] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. CoRR, abs/1904.05234, 2019.

[18] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. Sok: Transparent dishonesty: Front-running attacks on blockchain. In Financial Cryptography and Data Security - FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers, pages 170–189, 2019.

[19] Amir Feder, Neil Gandal, J. T. Hamrick, and Tyler Moore. The impact of ddos and other security shocks on bitcoin currency exchanges: evidence from mt. gox. J. Cybersecur., 3(2):137–144, 2017.

[20] Maurice Herlihy. Atomic cross-chain swaps. In ACM Symposium on PODC 2018, pages 245–254, 2018.

[21] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers, pages 72–86, 2014.

[22] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment channels that go faster than lightning. CoRR, abs/1702.05812, 2017.

[23] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial of service, 2019.

[24] Daniel Pérez and Benjamin Livshits. Broken metre: Attacking resource metering in EVM. CoRR, abs/1909.07220, 2019.

Syracuse University

NDSS SYMPOSIUM/2021

As Strong As Its Weakest Link: How to Break Blockchain DApps at RPC Service

Kai Li Syracuse University
Jiaqi Chen Syracuse University
Xianghong Liu Syracuse University
Yuzhe Tang Syracuse University
XiaoFeng Wang Indiana University Bloomington
Xiapu Luo Hong Kong Polytechnic University

Thread model

Ethereum-DApp ecosystem

Threat goal (DoERS)

Denial **o**f **E**thereum's **R**pc **S**ervice

Denied RPC service leads to real damage!

Denied RPC service is of interest to:

Competing services, frontrunning clients

Vulnerable service API

The RPC services privides API that allow clients call smartcontract functions which is free.

- `eth_call`
  - Allow client to call any smart-contract functions.
  - No Ether cost & local execution
- Benign uses:
  - DApp developer estimates Gas
  - DApp clients query BKC states
  - An essential business for RPC services
- Proposed misuse of `eth_call`:
  - A single `eth_call` to run contract of an infinite loop
  - Lead to a zero-cost DoS

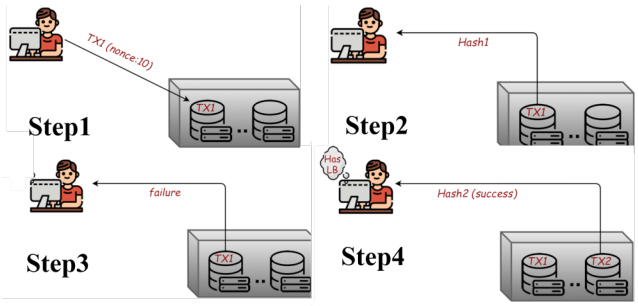
Deployed service exploitability

P:Protection in the wild M:Proposed misuse

P1: Gas limit
M1: multiple eth_calls

P2: Timeout (5-second default)
M2: eth_call to atomic instruction (CODECOPY)

P3: Load balancer in a RPC service
M3: Balancer-aware strategies



LB detected Case Process

Detail of M3

Challenging: Detect LB in a blackbox service from outside
Idea: Exploit orphan tx to detect LB.
Method: Send two orphan txs of same nonce to the target RPC service.

Both requests succeed ⇒ LB detected
One request fail ⇒ No LB detected

Type	RPC services	1IP-1key (LB0)	1IP-2key (LB1)	2IP-1key (LB2)	Gas limit
i	ServiceX1	✗	✗	✗	✗
	ServiceX2	✗	✗	✗	✗
	ServiceX3	✗	✗	✗	50
ii	ServiceX4	✗	✓	✗	✗
	ServiceX5	✗	✗	✓	✗
iii	ServiceX6	✓	✓	✓	10
	ServiceX7	✓	✓	✓	✗
	ServiceX9	✓	✓	✓	5
	ServiceX8	✓	✓	✓	1.5

Load balance result in 8 services

Evaluation

We designed an effective test on the target services, without attacking them.
The table below shows how DOERS attack work on different serveices. It caused an observable response-time increase by at least 3.8X.
The figures shows the effect of attack in two different services.

Services	⟨type,payload,rate⟩	Time	Gas*
ServiceX1	$\langle \texttt{CPU}, 2M, 10\rangle$	16×	13
ServiceX2	$\langle \texttt{CPU}, 0.15M, 30\rangle$	3.8×	0.2
ServiceX3	$\langle \texttt{CPU}, 3M, 0\rangle$	30×	19.5 (50)
ServiceX5	$\langle \texttt{Mem}, 50M, 0\rangle$	10×	5000
ServiceX4	$\langle \texttt{CPU}, 0.04M, 30\rangle$	4×	0.3
ServiceX6	$\langle \texttt{CPU}, 1.5M, 200\rangle$	5×	10 (10)
ServiceX7	$\langle \texttt{CPU}, 5M, 10\rangle$	15×	32.5
ServiceX9	$\langle \texttt{CPU}, 0.04M, 30\rangle$	2.1×	0.3 (5)
ServiceX8	$\langle \texttt{CPU}, 0.6M, 200\rangle$	110×	1.5 (1.5)