Research Statement

Ph.D. candidate: Yibo Wang

Software security and software engineering are critical for building systems that are reliable, efficient, and secure. In software security, the focus is on identifying and mitigating vulnerabilities that can compromise system integrity and availability. Meanwhile, software engineering emphasizes the systematic development of software that is scalable, maintainable, and secure. My research integrates these two areas by tackling security challenges in distributed systems. I work to identify vulnerabilities that threaten system security while addressing the high costs associated with transaction fees and operational inefficiencies. By focusing on both security and cost optimization, my research aims to build more secure, scalable, and cost-effective distributed systems that encourage broader adoption.

Theme 1: Automatic Design-flaw Discovery via Fuzzing

1a. Preliminary research: Mempool security and fuzzing

This project tackles broader security challenges in decentralized system, with a particular focus on blockchain mempool security. The mempool, where pending transactions are held before confirmation and inclusion in the blockchain, is crucial for system functionality. If the mempool service is denied, normal transactions cannot be included in the blockchain, which negatively impacts both the clients sending the transactions and the downstream services.

This research explores vulnerabilities in the blockchain mempool, focusing on how admission and eviction policies can be misused. It introduces a zero-Ether-cost denial-of-service (DoS) attack targeting the Ethereum mempool, exploiting weaknesses in transaction handling, such as issues with future and overdraft transactions. By targeting critical nodes in the Ethereum network, like major miners or RPC services, the attack can cause widespread disruption. My research identifies two vulnerabilities in Ethereum's transaction handling and develops strategies to address these risks, improving network security and resilience. This work "DETER: Denial of Ethereum Txpool sERvices" [3] has been published in the top-tier conference ACM CCS 2021. The DoS attack design (DETER) is extended into an active measurement method, revealing Ethereum network topology for the first time in the literature. This method systematically measures both the full-network topology in major testnets and the critical sub-network topology in the Ethereum mainnet. The work, "TopoShot: Uncovering Ethereum's Network Topology Leveraging Replacement Transactions" [2], is published in ACM IMC 2021.

To investigate whether other mempool DoS attacks exist beyond the DETER attack, we adopt a systematic and semi-automated approach. This exploration focuses on vulnerabilities in the blockchain mempool, specifically examining the misuse of admission and eviction policies rather than traditional system crash bugs. Given that existing blockchain fuzzers [1, 4, 9] are ineffective at detecting mempool-related DoS vulnerabilities, we propose MPFUZZ, a symbolized-stateful mempool fuzzer. MPFUZZ identifies potential vulnerabilities by defining the search space using symbolized transaction sequences. It efficiently explores this space using feedback from state coverage and state promissingness to trigger bug oracles. Running MPFUZZ on Ethereum clients uncovers new mempool DoS vulnerabilities, including stealthy mempool eviction and locking, in addition to known vulnerabilities like DETER and MemPurge [8]. Results are detailed in the publication, "Understanding Ethereum Mempool Security under Asymmetric DoS by Symbolized Stateful Fuzzing" [6], presented at the top-tier USENIX Security 2024 conference.

1b. Next research: Understand decentralized system security using LLM assisted fuzzing

Fuzzing gossip protocols beyond the mempool: The next step in blockchain fuzzing research will expand beyond the mempool to target gossip protocols, which are essential for fast and reliable transaction and block propagation among nodes. In Ethereum, transactions utilize multi-round propagation, where nodes first send an announcement, wait for a request, and then transmit the full transaction. This approach aims to reduce network congestion and ensure efficient dissemination of information. However, this method's complexity introduces exploitable vulnerabilities. An attacker could trigger a mempool reorganization and evict old propagation records, causing the same transaction to propagate twice. This can lead to the victim node being dropped by its peers and disrupting network reliability.

To address these vulnerabilities, the research will implement fuzzing techniques for transaction propagation, focusing on the mutation of symbolized transactions to explore various propagation scenarios. State coverage feedback will guide the fuzzing process. An oracle will be employed to identify key outcomes, such as instances where the same transaction propagates twice, valid transactions that experience no propagation, and delays in transaction propagation. This approach aims to uncover vulnerabilities in gossip protocols and enhance the robustness of blockchain networks.

Automated symbol extraction for fuzzing block validation: In blockchain clients, block validation ensures that proposed blocks meet consensus rules and contain only valid transactions. However, attackers can exploit this complex process through DoS attacks by crafting malicious transactions that consume a validator's computational resources without passing validation. These attacks can exhaust resources, slow down validation, and disrupt block-building while incurring no transaction fees, posing a significant threat to network stability.

This research aims to automate the extraction of symbols from block validation code paths, reusing the concept of transaction symbolization. Each code path, from transaction argument checks to execution results, will be abstracted into symbols representing specific execution paths. This method allows for more efficient and targeted fuzzing, focusing on detecting weaknesses where resource-intensive transactions can trigger DoS attacks during block validation. Such transactions may pass initial checks but consume excessive computational resources before failing validation. Symbolized fuzzing will enable a deeper exploration of these scenarios, helping to uncover DoS vulnerabilities in block validation.

Fuzzing blockchain network stacks beyond Ethereum: The next research direction focuses on the security of non-Ethereum blockchains, particularly those using novel programming models like Move. These blockchains enhance smart contract security by employing static analysis techniques, such as control flow graph (CFG) generation and type checking, before execution. This process helps identify vulnerabilities and ensures code adherence to expected behaviors. However, as these frameworks evolve, they introduce unique security challenges. For instance, Move-based blockchains like Sui are vulnerable to DoS attacks due to flaws in transaction validation and pre-execution verification. This research will analyze how these systems handle complex payloads and verification tasks, specifically investigating flaws that may cause verifiers to enter endless loops during semantic processing, potentially leading to system shutdowns. The goal is to develop targeted security measures to improve the resilience and stability of these non-Ethereum blockchains against sophisticated attacks.

Fuzzing with LLM-extracted specifications and models: The development of Large Language Models (LLMs) has significantly improved performance across various tasks, offering a promising enhancement to fuzzing techniques for uncovering non-crash bugs. The key insight is using LLMs to identify inconsistencies between specifications and implementations, a challenge that traditional methods often miss.

LLMs automate and accelerate the extraction of information from blockchain client specifications, a task traditionally done manually. They can parse complex protocols and create machine-readable specifications, providing a precise ground truth for spotting implementation bugs in blockchain clients. This method streamlines bug detection and reduces human error and bias. Unlike traditional differential fuzzing [9], which is cumbersome and imprecise, LLMs offer a systematic approach to identifying discrepancies. This enables thorough analysis of inconsistencies across implementations, such as Ethereum clients, enhancing the efficiency of bug detection and contributing to more robust and secure blockchain systems.

This research targets diverse blockchain implementations in different programming languages, including Ethereum, Bitcoin, Cosmos, and the Account Abstraction (bundler) model. LLMs are used to automate specification analysis, extracting critical input information such as message formats and parameter types. This enables the generation of valid input arguments tailored for fuzzing. Additionally, LLMs extract expected output information, which facilitates precise verification of execution results against the specified behavior. This approach ensures comprehensive testing of blockchain clients, improving security and conformance across multiple implementations.

Theme 2: Workload Analysis and Cost Optimization in Decentralized Systems

2a. Preliminary research: Transaction batching

This project tackles the challenge of high transaction fees limiting blockchain's broader adoption. Rising costs for transactions and application maintenance hinder its applicability. Our research focuses on reducing the costs associated with blockchain-based decentralized applications (DApps) while maintaining security and minimizing latency. A key innovation of this project is the design of a method to aggregate multiple smart contract calls, to spread out transaction fees more efficiently. We developed iBatch, a middleware system on Ethereum, which securely batches smart contract invocations and protects against transaction manipulation. iBatch reduces Gas costs per invocation by 14.6% to 59.1%, with only a modest 2-minute delay. This research, titled “iBatch: Saving Ethereum Fees via Secure and Cost-Effective Batching of Smart-Contract Invocations” [7], was presented at the top tier ESEC/FSE 2021 software engineering conference.

To enable iBatch to work with existing contract code, techniques are developed to rewrite smart contracts at both the Solidity and EVM bytecode levels. iBatch is also adapted for other platforms, like TRON and EOS.IO, achieving similar cost reductions. This extended research, titled “Towards Saving Blockchain Fees via Secure and Cost-Effective Batching of Smart-Contract Invocations” [5], was published in the IEEE Transactions on Software Engineering (TSE) in 2023.

2b. Next research: RL-based workload analysis and adaptive state placement

The cost of storage on the blockchain is a significant concern, as both data uploading and storage are expensive. To optimize costs and enable more viable decentralized applications (DApps), I will design an Ethereum middleware system that supports flexible transaction execution by switching between on-chain and off-chain environments. This system will store receiver states on an off-chain server and dynamically determine when to update on-chain states through batching. My research will include analyzing blockchain workloads and implementing adaptive mechanisms for optimal timing of state updates, aiming to reduce transaction fees and overall operation costs without compromising security and integrity. By optimizing the cost structure, this approach will make DApps more affordable and foster wider adoption of blockchain technology.

Furthermore, I will employ reinforcement learning (RL) to enhance cost and performance optimization in blockchain systems. Integrating RL into the iBatch framework will improve decision-making for transaction batching and delay durations, allowing us to identify the most effective transactions to batch and the optimal timing for delays. Additionally, RL will guide the adaptive execution of workloads, determining which transactions to store off-chain and when to upload them on-chain. By dynamically adjusting the placement of transactions, we aim to improve overall cost efficiency.

Theme 3: Software Supply-chain Security by Decentralization

3a. Preliminary research: Decentralized bug reporting for smart contracts

This research focuses on developing a decentralized bug-reporting system for smart contracts to address the limitations of centralized systems, such as CVE. In current centralized bug databases, privileged users can manipulate reports, delay disclosures, and lack transparency in bug verification. The proposed decentralized system eliminates these issues by enabling anyone to submit bug reports directly to the blockchain, with transparency and immutability. A decentralized group of verifiers is responsible for ensuring the accuracy and legitimacy of these reports, preventing manipulation, censorship, or reliance on any single entity.

The system workflow begins with a bug reporter submitting an encrypted Proof of Evidence (PoE) to the blockchain, containing both exploit transactions and their associated pre-conditions and post-conditions. A back-end server decrypts the PoE and passes it to a decentralized voting group, which determines whether the reported issue qualifies as a valid attack. Upon reaching consensus, the PoE is forwarded to verifiers who perform validation within a Trusted Execution Environment (TEE). By setting the blockchain to the pre-exploit state and verifying the outcome against the post-conditions, the system ensures secure and accurate validation, safeguarding the decentralized nature of the bug reporting process. All submissions and results are recorded on the blockchain, ensuring transparency and immutability.

Building on this, the next phase of research aims to enhance the system by incorporating LLMs as bug oracles. Currently, the validity of bug reports is determined by a group of voters, which can be time-consuming and subjective. LLM oracles can automate and improve this process by providing more accurate insights into whether a transaction triggers a bug or constitutes an attack. This integration will reduce reliance on manual voting, increase precision in bug classification, and streamline the overall verification process, making smart contract security assessments more efficient and reliable.

3b. Next research: Decentralized bug reporting for software applications

Building upon the advancements in smart contract bug reporting, the next research phase aims to extend the decentralized bug reporting framework to software applications more broadly. Traditional bug reporting systems for software applications often suffer from similar issues as centralized systems for smart contracts, including manipulation, delayed disclosures, and lack of transparent verification. By applying the decentralized approach used in smart contract reporting, we propose developing a system where bug reports for general software applications are submitted to a blockchain, ensuring transparency and immutability. A decentralized network of verifiers will validate these reports, utilizing similar methods to those in the smart contract system. This expansion will enhance security across various software applications, creating a more resilient and scalable bug reporting ecosystem that ensures the integrity and reliability of vulnerability detection and confirmation.

Theme 4: DoS Security and Cost Optimization in LLM-Service Markets

4a. Next research: Understanding cost model in LLM services

As Large Language Models (LLMs) become increasingly prevalent, ensuring robust security and cost efficiency is crucial. This research focuses on understanding the cost model within LLM service markets, particularly how it impacts resource allocation and potential vulnerabilities. Current models often disconnect service fees from actual resource usage, allowing attackers to exploit the system for minimal costs. For instance, they can overwhelm CPUs, memory, and GPUs by sending complex queries or making numerous requests, all while paying very little. This research aims to analyze existing cost models to identify their weaknesses and propose adjustments that better reflect resource consumption. By optimizing the cost model, the goal is to prevent exploitation and improve overall system efficiency. Techniques such as automated input mutation and test case generation will be employed to simulate potential attacks and pinpoint vulnerabilities within the system.

4b. Next research: Understanding DoS risks in open-source LLM services

Alongside the analysis of cost models, this research will examine the DoS risks related to open-source LLM services. While the open-source nature of these platforms encourages collaboration and innovation, it also presents unique security challenges. Attackers can exploit the accessibility of open-source LLMs to launch DoS attacks that overload system resources without incurring significant costs. Such attacks can disrupt service availability and degrade performance. The focus of this research will be on identifying specific vulnerabilities in open-source LLM services that could be exploited in DoS attacks. Techniques like fuzzing will generate crafted prompt inputs to stress-test the system and reveal weaknesses. Additionally, the research will explore mitigation strategies, including adaptive pricing models that adjust fees based on resource usage, rate-limiting measures to manage excessive requests, and resource-limiting techniques to prevent overuse. Load balancing will also be implemented to ensure even distribution of tasks and safeguard system integrity. By understanding the DoS risks in open-source LLM services, this research aims to enhance their resilience and maintain sustainable performance in a competitive market.

Conclusion

In conclusion, my research tackles key challenges at the intersection of blockchain technology and system security, focusing on enhancing security and cost-efficiency in decentralized systems. Projects like MPFUZZ and iBatch have led to significant advances in vulnerability detection and transaction cost optimization, supporting broader blockchain adoption. Future work will expand on these achievements by developing automated blockchain fuzzing techniques and utilizing LLMs for improved vulnerability detection and system resilience. In addition, I will develop a decentralized bug reporting system for smart contracts and general software applications, enhancing transparency and integrity. Furthermore, I will address security and cost optimization in LLM service markets to mitigate disproportionate resource exploitation.

Looking ahead, I am excited to further my research at your department. By collaborating with esteemed colleagues and students, I aim to drive innovation in blockchain and security technologies, contributing to their transformative impact and ensuring their robustness in the face of evolving threats.

References

[1] Y. Chen, F. Ma, Y. Zhou, Y. Jiang, T. Chen, and J. Sun. Tyr: Finding consensus failure bugs in blockchain system with behaviour divergent model. In 2023 2023 IEEE Symposium on Security and Privacy (SP) (SP), pages 2517–2532, Los Alamitos, CA, USA, may 2023. IEEE Computer Society.

[2] Kai Li, Yuzhe Tang, Jiaqi Chen, Yibo Wang, and Xianghong Liu. Toposhot: uncovering ethereum's network topology leveraging replacement transactions. In Dave Levin, Alan Mislove, Johanna Amann, and Matthew Luckie, editors, IMC '21: ACM Internet Measurement Conference, Virtual Event, USA, November 2-4, 2021, pages 302–319. ACM, 2021.

[3] Kai Li, Yibo Wang, and Yuzhe Tang. DETER: denial of ethereum txpool services. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021, pages 1645–1667. ACM, 2021.

[4] Fuchen Ma, Yuanliang Chen, Meng Ren, Yuanhang Zhou, Yu Jiang, Ting Chen, Huizhong Li, and Jiaguang Sun. LOKI: state-aware fuzzing framework for the implementation of blockchain consensus protocols. In 30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023. The Internet Society, 2023.

[5] Yibo Wang, Kai Li, Yuzhe Tang, Jiaqi Chen, Qi Zhang, Xiapu Luo, and Ting Chen. Towards saving blockchain fees via secure and cost-effective batching of smart-contract invocations. IEEE Trans. Software Eng., 49(4):2980–2995, 2023.

[6] Yibo Wang, Yuzhe Tang, Kai Li, Wanning Ding, and Zhihua Yang. Understanding ethereum mempool security under asymmetric dos by symbolized stateful fuzzing. In Davide Balzarotti and Wenyuan Xu, editors, 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024. USENIX Association, 2024.

[7] Yibo Wang, Qi Zhang, Kai Li, Yuzhe Tang, Jiaqi Chen, Xiapu Luo, and Ting Chen. ibatch: saving ethereum fees via secure and cost-effective batching of smart-contract invocations. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta, editors, ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021, pages 566–577. ACM, 2021.

[8] Aviv Yaish, Kaihua Qin, Liyi Zhou, Aviv Zohar, and Arthur Gervais. Speculative denial-of-service attacks in ethereum. In Davide Balzarotti and Wenyuan Xu, editors, 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024. USENIX Association, 2024.

[9] Youngseok Yang, Taesoo Kim, and Byung-Gon Chun. Finding consensus bugs in ethereum via multi-transaction differential fuzzing. In Angela Demke Brown and Jay R. Lorch, editors, 15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021, pages 349–365. USENIX Association, 2021.