Tutorial: Analyzing, Exploiting, and Patching Smart Contracts in Ethereum

Jens-Rene Giesen* Sebastien Andreina† Michael Rodler* Ghassan O. Karame†‡ Lucas Davi*

*University of Duisburg-Essen †NEC Laboratories Europe ‡Ruhr-Universität Bochum

***Abstract*—Smart contracts are programs which encode business logic and execute on the blockchain. While Ethereum is the most popular blockchain platform for smart contracts, an increasing number of new blockchain platforms are also able to support smart contract execution (e.g., Solana or Cardano). Security vulnerabilities in Ethereum smart contracts have demonstrated that writing secure smart contracts is highly challenging. This is exacerbated by the fact that the exploitation of buggy smart contracts seems disproportionately easier compared to exploiting classic PC software.**

In this tutorial, we overview a number of smart contract vulnerabilities focusing on the Ethereum ecosystem. We also provide an introduction to the de-facto smart contract programming language Solidity and provide a comprehensive hands-on lab tutorial that involves analyzing vulnerable smart contracts, developing proof-of-concept exploits as well as introducing security analysis tools for testing smart contracts.

I. Introduction

Most modern blockchain technologies support smart contracts–that is, the execution of decentralized programs which encode business logic within the blockchain. This has been fueled by the increasing popularity of Ethereum smart contracts, that often handle cryptocurrency worth millions of US-Dollars. Unfortunately, the supporting tooling around smart contract development neglects important safety and security features prevalent in modern day development cycles, which are either not integrated within smart contracts or not sufficiently usable [1]. This is evidenced in Ethereum smart contracts being plagued by several types of bugs that can lead to security vulnerabilities [2].

Some of these vulnerabilities resulted in high-profile attacks against the Ethereum blockchain, such as the notorious *TheDAO* attack, the *Parity Multisig Wallet* attack, and the more recent *Lendf.me* and *Uniswap* attacks. Although these attacks and their root causes have fueled research to strengthen the security of smart contracts for several years, one root cause, namely reentrancy, continues to regularly cause millions of US-Dollars in losses today [3], [4], [5].

Our goal in this tutorial is to introduce the participants to the Ethereum platform including the Solidity smart contract programming language, and explore common smart contract vulnerabilities, exploits, and possible defensive measures. Based on a hands-on lab, participants will experience smart contract vulnerabilities from the perspective of both developers and attackers.

Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2092 CASA – 390781972 and SFB 1119 – 236615297 project T1.

II. Background

A. Ethereum

Ethereum is a blockchain platform that combines its native cryptocurrency *Ether* with a decentralized virtual machine called Ethereum Virtual Machine (EVM). The EVM executes smart contracts, which effectively consist of programs operating on the Ethereum blockchain. Ethereum smart contracts are commonly written in the Solidity programming language, and compiled to the EVM bytecode format. After its deployment, the bytecode of a smart contract is stored in the same address space as account data of externally owned accounts (EOAs)–that is, accounts operated by a human and not by smart contract code. Thus, addresses of smart contracts and common users are relatively indistinguishable in Ethereum.

Every interaction with the Ethereum platform, such as calling smart contract functions, requires users to create and send transactions, which are stored on the blockchain ledger once executed. When calling smart contracts, the user has to account for the costs of their execution through the gas mechanism. Gas costs vary starkly between different EVM instructions. Moreover, a smart contract, once called, may invoke other smart contracts at will.

B. Reentrancy Vulnerabilities and Arithemetic Integer Bugs

Reentrancy vulnerabilities allow attackers to leverage state inconsistencies by reentering the contract in unexpected or unintended ways. A basic form of malicious reentrancy is possible when an attacker can *trick* a victim contract into calling another (usually attacker controlled) contract *after* transferring value, but *before* updating its state.

Due to cross-contract interactions, reentrancy cannot always be entirely avoided, and can be exploited if not carefully implemented. Several attacks against Ethereum smart contracts have demonstrated how vulnerable implementations can lead to a tremendous loss of value [3], [4], [5], [6].

Arithmetic integer bugs occur when the result of an arithmetic operation grows either greater (integer overflow) or lower (integer underflow) than the size of the target type allows. Although arithmetic integer bugs exist long before the Ethereum platform, such bugs were actively exploited in popular smart contracts [7].

C. Tools

During the hands-on session of this tutorial, participants will be introduced to several tools. When writing smart contract

code, participants will learn how to use the beginner-friendly smart contract IDE Remix. To analyze smart contracts for security vulnerabilities, participants will further use Slither and Osiris to find bugs in their target smart contract. Finally, to assist in finding transaction sequences for exploitation, participants can use Mythril. Note that this list is in no way exclusive, i.e., participants are free to use any other tool they prefer.

III. Tutorial Format

We design this tutorial for a total length of 3 hours, consisting of two main parts, namely 1) an interactive presentation part (approx. 1 hour) providing background on Ethereum smart contracts and smart contract vulnerabilities, and 2) a hands-on session for participants (approx. 2 hours) to solve exercises in analyzing, exploiting, and fixing smart contracts.

During the presentation part, we introduce basic blockchain technologies and smart contracts in Ethereum, including an overview of common vulnerabilities in the Ethereum ecosystem as well as a brief introduction to Solidity, the de-facto smart contract programming language for Ethereum smart contracts. After this introduction, we move to an interactive hands-on lab during which we guide the participants in analyzing and exploiting a basic smart contract in our virtual environment.

The hands-on session consists of several exercises. Each exercise provides participants with the opportunity to analyze the smart contract with state-of-the-art smart contract analysis tools that are pre-installed on a virtual machine we provide for this tutorial (see Section IV-A for requirements). Moreover, participants leverage the insights from their analysis of the target smart contract to exploit the analyzed contract. This includes developing simple attacker contracts and crafting transactions that trigger faulty behavior, such as reentrancy and integer overflow vulnerabilities. Finally, participants will develop patches that prevent exploitation of the smart contract vulnerability. Note that each exercise of smaller tasks providing guidance for participants towards the overall goal of the exercise.

IV. Audience & Learning Goals

A. Expected Audience

This tutorial is intended for those interested in smart contracts and blockchain technologies in general. As the tutorials covers an introduction to all relevant concepts of the Ethereum platform, we do not assume any prior knowledge of blockchain or smart contract technologies. However, we consider it beneficial for the participants of this tutorial to have a basic understanding of software security as well as entry-level experience with basic Linux shell commands. We further require the participants to bring laptops capable of running x86-64 VirtualBox virtual machine images, either in VirtualBox or any compatible virtualization solution.

B. Learning Goals

This tutorial introduces the participants to the basics of smart contracts in the Ethereum ecosystem and common smart contract vulnerabilities. The hands-on session on smart contract analysis of this tutorial facilitates a deep understanding of one of the most notorious vulnerabilities on this platform, namely reentrancy, and how to exploit such vulnerabilities. By introducing the participants to state-of-the-art analysis tools, we also expect this tutorial to strengthen their sensitivity for smart contract security within development processes. Finally, the last part of our hand-on session provides useful hints on defensive smart contract development.

V. Previous Tutorials

Ghassan Karame is professor at the Ruhr Universität Bochum and held tutorials on Bitcoin and blockchain security at *ACM CCS 2016* [8] and at *ASIACCS 2017*. Lucas Davi and Ghassan Karame are principal investigators in a joint project on smart contract security. Lucas Davi is professor at the University of Duisburg-Essen and held tutorials on software exploits and defenses at the *ACM/IEEE Design Automation Conference (DAC) 2016* and ESWEEK 2015. Michael Rodler is a doctoral student at the University of Duisburg-Essen with research experience in the field of smart contract security [9], [10]. Sebastien Andreina is a research scientist at NEC Laboratories Europe and has a large industry experience on blockchain technology. Jens-Rene Giesen is a doctoral student at the University of Duisburg-Essen collaborating with Sebastien Andreina on smart contract security and compilers [11].

References

[1] T. Sharma, Z. Zhou, A. Miller, and Y. Wang, "Exploring security practices of smart contract developers," apr 2022. [Online]. Available: http://arxiv.org/abs/2204.11193

[2] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *International Conference on Principles of Security and Trust*. Springer, 2017, pp. 164–186.

[3] C. Williams, "$8.2M Lost as Visor Finance Suffers Latest DeFi Hack," Dec. 2021. [Online]. Available: https://cryptobriefing.com/8-2m-lost-visor-finance-suffers-latest-defi-hack/

[4] J. Benson, "Grim Finance Hacked for $30 Million in Fantom Tokens," Dec. 2021. [Online]. Available: https://decrypt.co/88727/grim-finance-hacked-30-million-fantom-tokens

[5] M. White, ""NFT mortgage lender" Bacon Protocol is hacked for $1 million." Mar. 2022. [Online]. Available: https://web3isgoinggreat.com/?id=bacon-protocol-hacked-for-1-million

[6] C. Jentzsch. (2017) The History of the DAO and Lessons Learned. [Online]. Available: https://blog.slock.it/the-history-of-the-dao-and-lessons-learned-d06740f8cfa5

[7] p0n1. (2018) A disastrous vulnerability found in smart contracts of BeautyChain (BEC). [Online]. Available: https://medium.com/secbit-media/a-disastrous-vulnerability-found-in-smart-contracts-of-beautychain-bec-dbf24ddbc30e

[8] G. Karame, "On the security and scalability of bitcoin's blockchain," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. Association for Computing Machinery, pp. 1861–1862. [Online]. Available: https://doi.org/10.1145/2976749.2976756

[9] M. Rodler, W. Li, G. Karame, and L. Davi, "Sereum: Protecting existing smart contracts against re-entrancy attacks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2019.

[10] M. Rodler, W. Li, G. O. Karame, and L. Davi, "EVMPatch: Timely and automated patching of ethereum smart contracts," in *USENIX Security Symposium (USENIX Security 2021)*, 2021.

[11] J.-R. Giesen, S. Andreina, M. Rodler, G. O. Karame, and L. Davi, "Practical mitigation of smart contract bugs," 2022. [Online]. Available: https://arxiv.org/abs/2203.00364