A Systematic Study of Recent Smart Contract Security Vulnerabilities

Zhuo Zhang[1,4], *Brian Zhang*[2], Wen Xu[3,4], Zhiqiang Lin[5]

[1]Purdue University [2]Harrison High School (Tippecanoe)

[3]Georgia Institute of Technology [4]PNM Labs [5]Ohio State University

Introduction

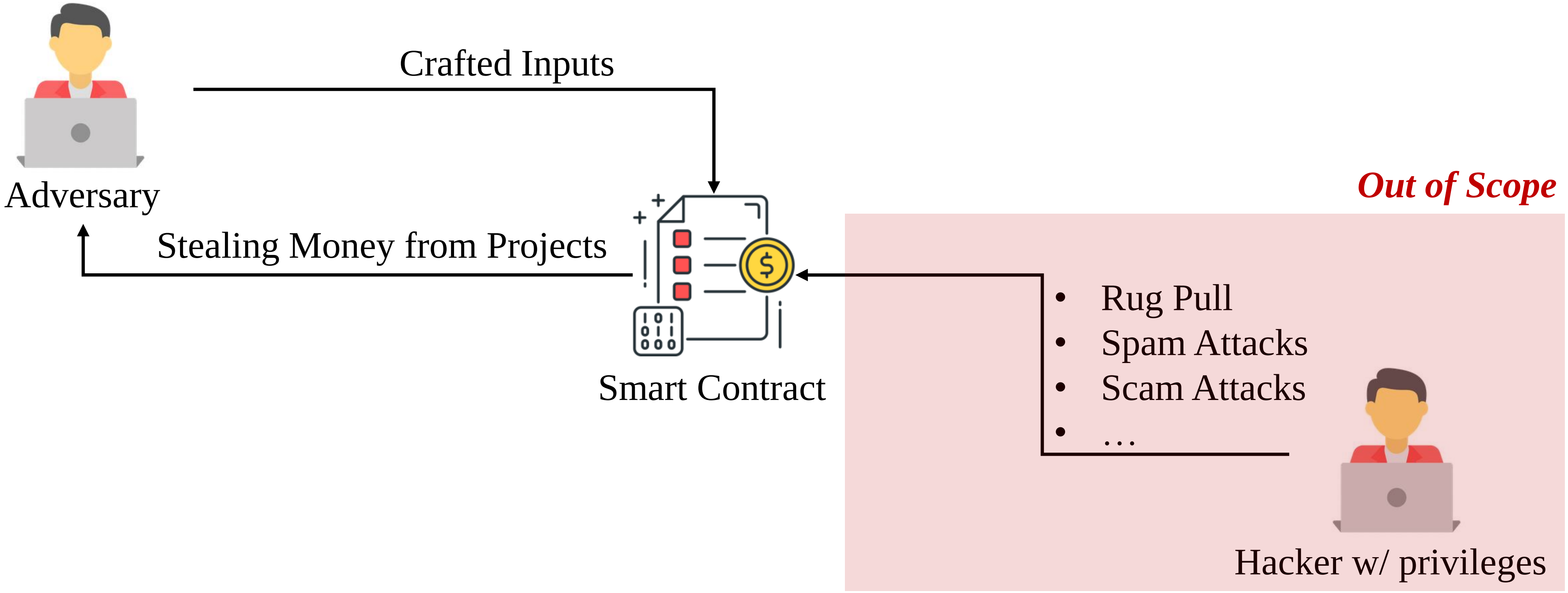
Exploitable bugs in smart contracts have caused *significant monetary loss ($1.57 billion were exploited from various smart contracts as of May 1st, 2022)*, despite the substantial advances in smart contract bug finding.

It is hence interesting to understand

- The effectiveness of existing techniques to detect real-world vulnerabilities
- The categories and distributions of bugs that cannot be detected by existing techniques (i.e., machine unauditable bugs)
- *How we can further improve existing techniques*

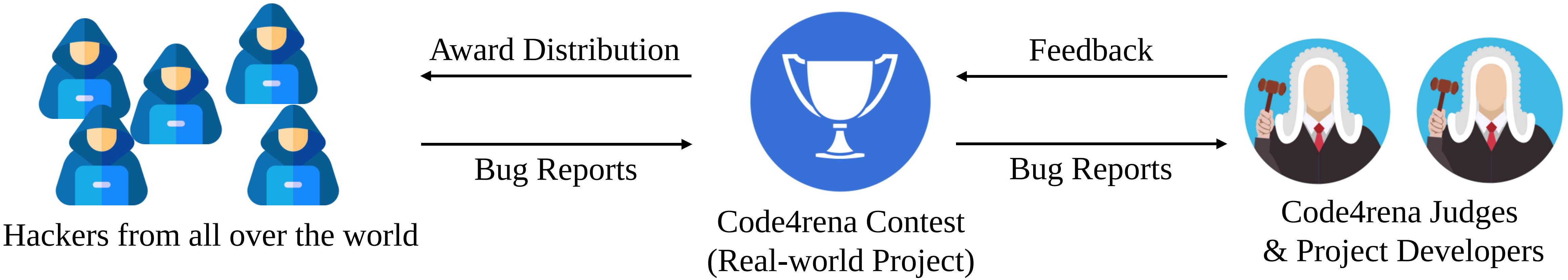
Threat Model

- In our threat model, an *adversary* is a contract user who crafts special inputs to exploit the on-chain contract and further cause monetary loss.



Data Collection

- Code4rena[1] is a highly reputable audit contest platform, specificized for Web 3.0 auditing.



[1] https://code4rena.com/

Data Collection

- Code4rena[1] is a highly reputable audit contest platform, specificized for Web 3.0 auditing.

113 Code4rena contests

$2.8B Fund protected

358 Hackers/teams participated

$6.7M Bounty paid out

462 Bugs analyzed, among which 341 are in-scope

[1] https://code4rena.com/

Data Collection

- Code4rena[1] is a highly reputable audit contest platform, specificized for Web 3.0 auditing.

- We also studied 54 real-world exploits happened from January 2022 to June 2022 (Details can be found in our paper).

Research Questions

- How many real-world exploitable bugs are $\boxed{\text{machine auditable}}$?

  Bugs can be detected by existing techniques

- What are the categories and distributions of machine unauditable bugs?

Research Questions

- How many real-world exploitable bugs are machine auditable?

  Bugs can be detected by existing techniques

- What are the categories and distributions of machine unauditable bugs?
- How ***difficult*** is it to audit exploitable bugs?
- What are the ***symptoms*** and fixes of machine unauditable bugs?
- Can machine unauditable be properly abstracted such that automated oracles can be devised?

Details can be found in our paper

RQ1: How many real-world exploitable bugs are machine auditable?

- What kinds of Bugs are Machine-auditable?

Existing Techniques (38)

Fuzzing	(11)
Static Analysis	(12)
Formal Verification	(5)
Symbolic Execution	(10)

Transaction-ordering Dependency
Integer Bug
Transaction Origin Use
Control-flow Hijack
Suicidal Contract
Ether Leak
Arbitrary Write
Weak PRNG
Gas-related Issue
Block-state Dependency
Reentrancy
Assertion Failure
Ether Freezing
Precision Loss
Compiler Error
Mishandled Exception
Uninitialized Variables

Machine-Auditable Bugs (17):
Their oracles are ***general*** and ***sufficiently simple*** to support instantiations in a wide range of projects.

RQ1: How many real-world exploitable bugs are machine auditable?

- What kinds of Bugs are Machine-auditable?

Alarm once any integer variable is overflow or underflow.

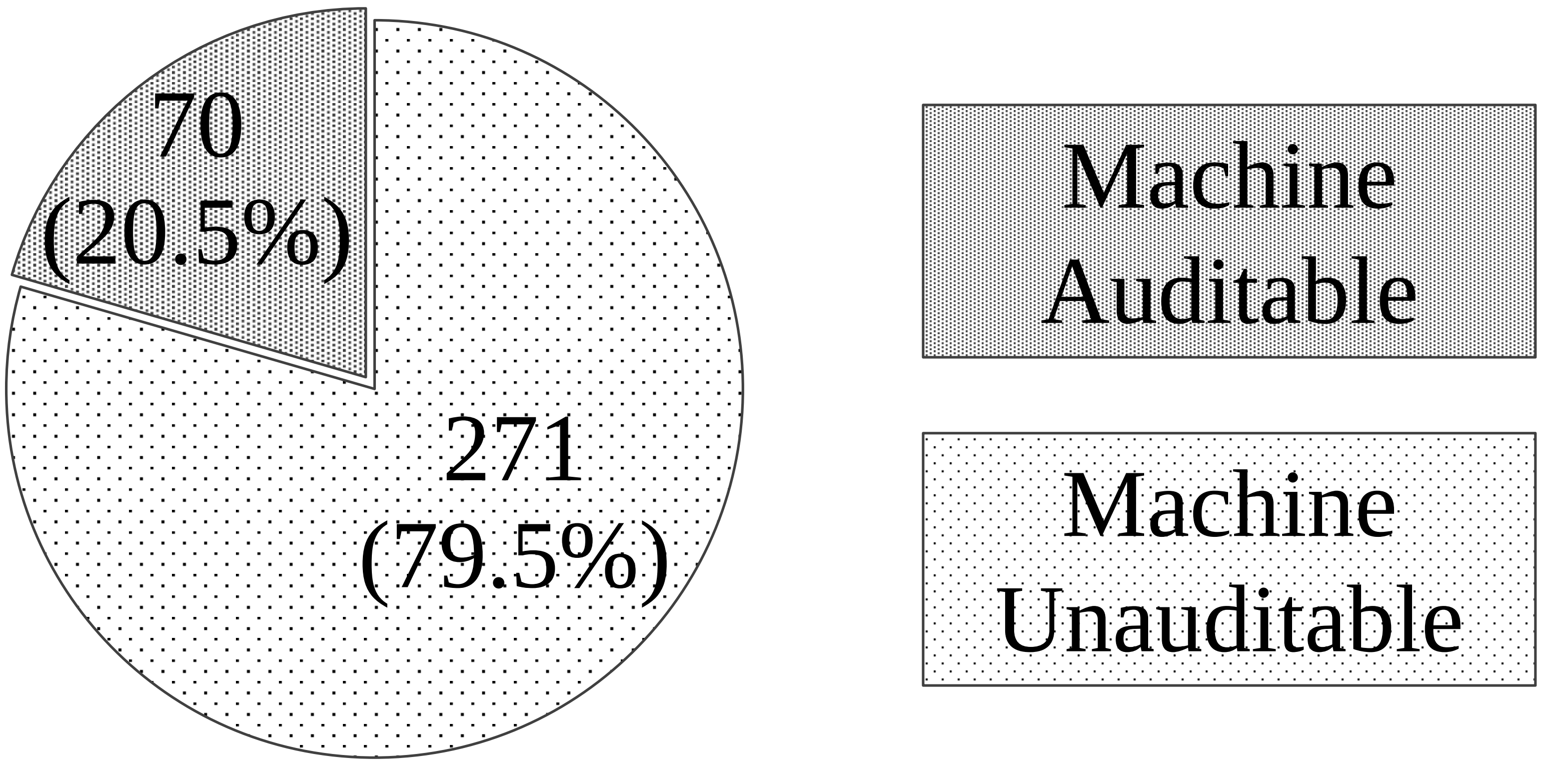
Existing Techniques (38)

- Fuzzing (11)
- Static Analysis (12)
- Formal Verification (5)
- Symbolic Execution (10)

Transaction-ordering Dependency
Integer Bug
Transaction Origin Use
Control-flow Hijack
Suicidal Contract
Ether Leak
Arbitrary Write
Weak PRNG
Gas-related Issue
Block-state Dependency
Reentrancy
Assertion Failure
Ether Freezing
Precision Loss
Compiler Error
Mishandled Exception
Uninitialized Variables

Machine-Auditable Bugs (17):
Their oracles are ***general*** and ***sufficiently simple*** to support instantiations in a wide range of projects.

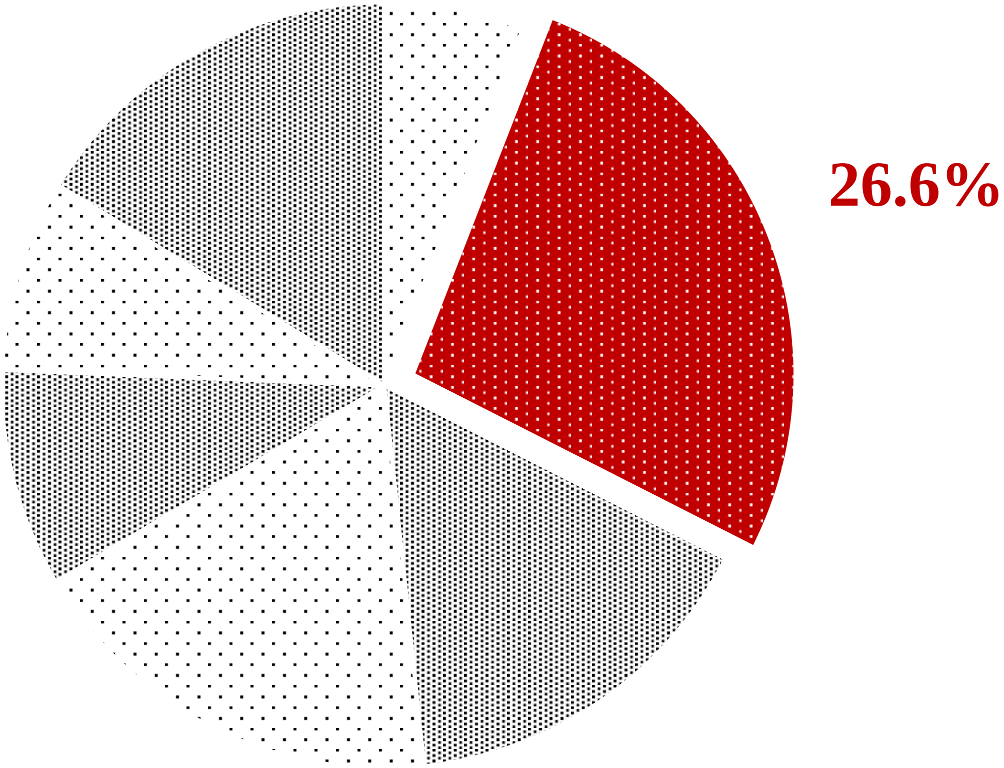
RQ1: How many real-world exploitable bugs are machine auditable?



<u>Finding</u>: A large portion of exploitable bugs in the wild (i.e., 79.5%) are not machine auditable.

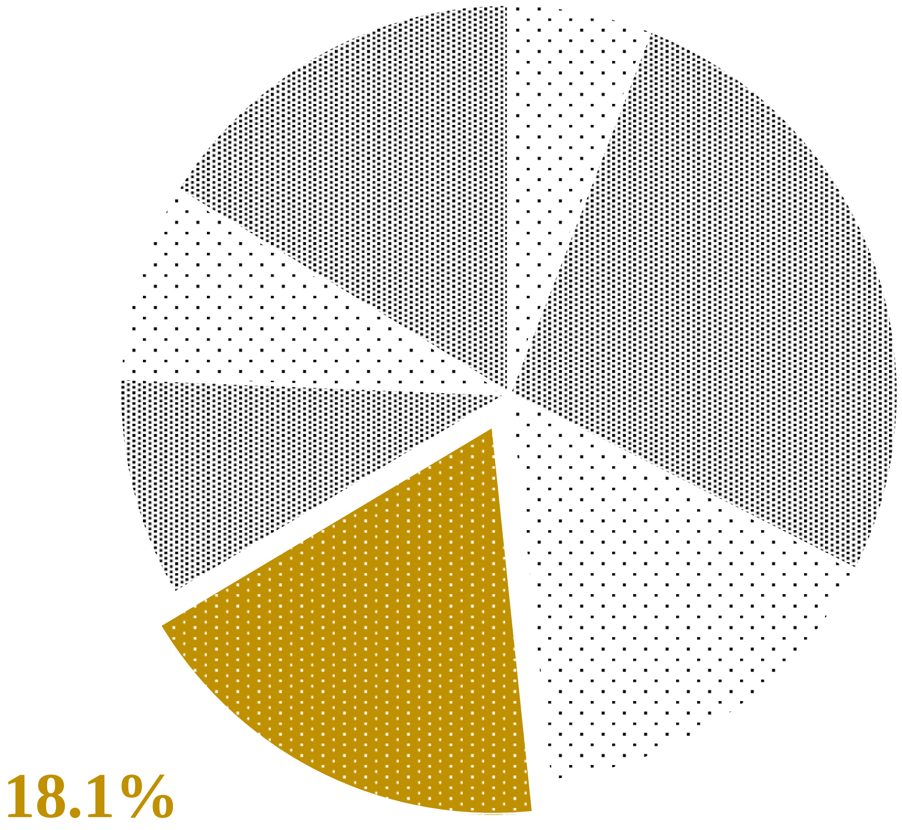
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6% of 462)**
  - Incorrect implementation of existing domain-specific financial models
  - Most popular amongst audit contests because contests bring in very broad domain expertise on various business models



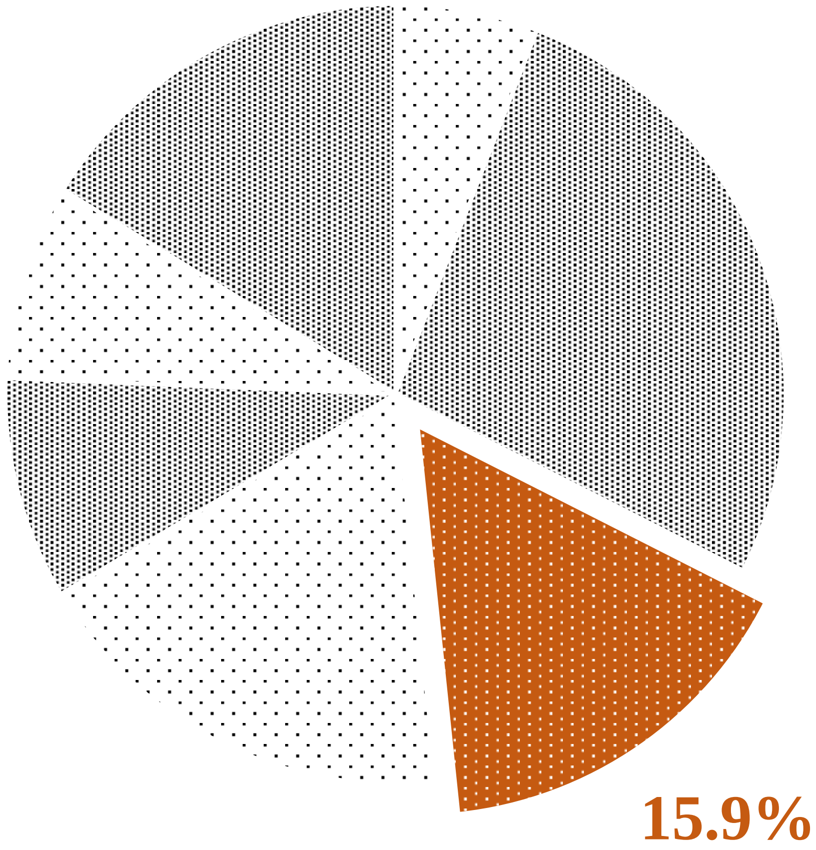
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6%)**
- **Inconsistent State Updates (18.1% of 462)**
  - Internal contract storage not updated completely after state changes
  - Usually small in impact, but can be accumulated for bigger effect



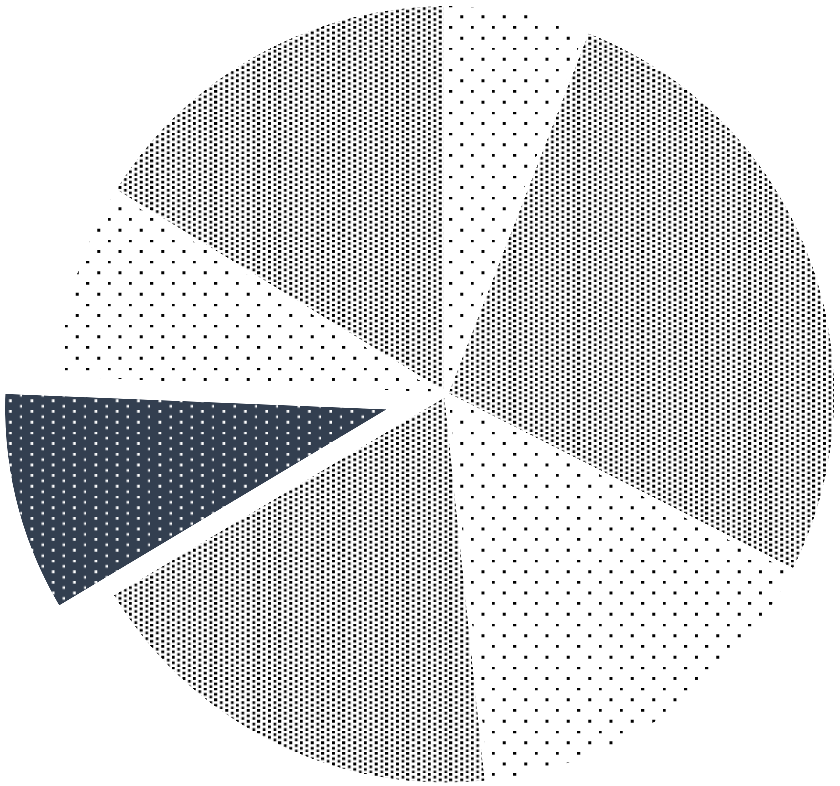
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6%)**
- **Inconsistent State Updates (18.1%)**
- **ID Uniqueness Violation (15.9%)**
    - Misuse/Lack of access control in ID-specific functionalities
    - Easiest to find



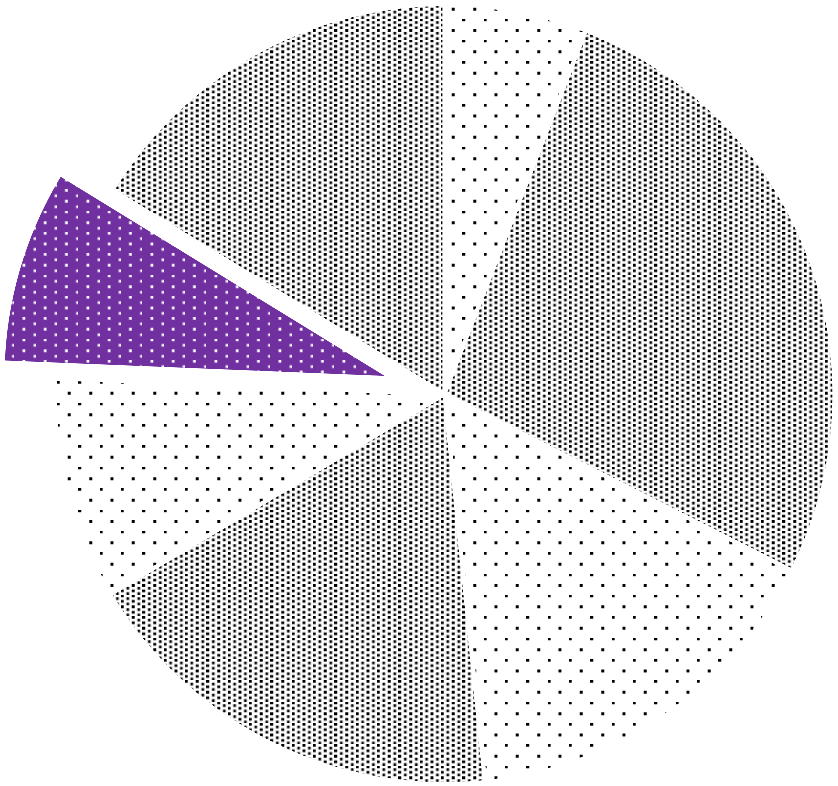
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6%)**
- **Inconsistent State Updates (18.1%)**
- **ID Uniqueness Violation (15.9%)**
- **Privilege Escalation (9.2%)**
  - Unexpected business flow that leads to weaker access control
  - Modification of existing program analysis tools may help prevent these bugs



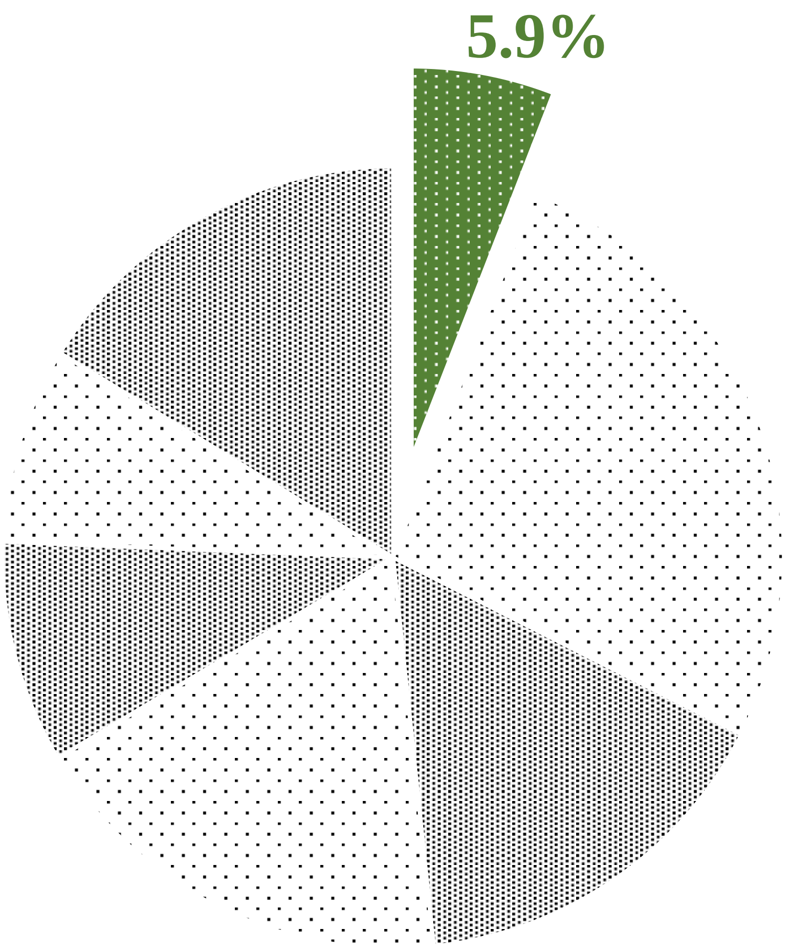
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6%)**
- **Inconsistent State Updates (18.1%)**
- **ID Uniqueness Violation (15.9%)**
- **Privilege Escalation (9.2%)**
- **Atomicity Violations (8.1%)**
  - Action sequences may modify values that are in use by other sequences
  - Second most difficult to find



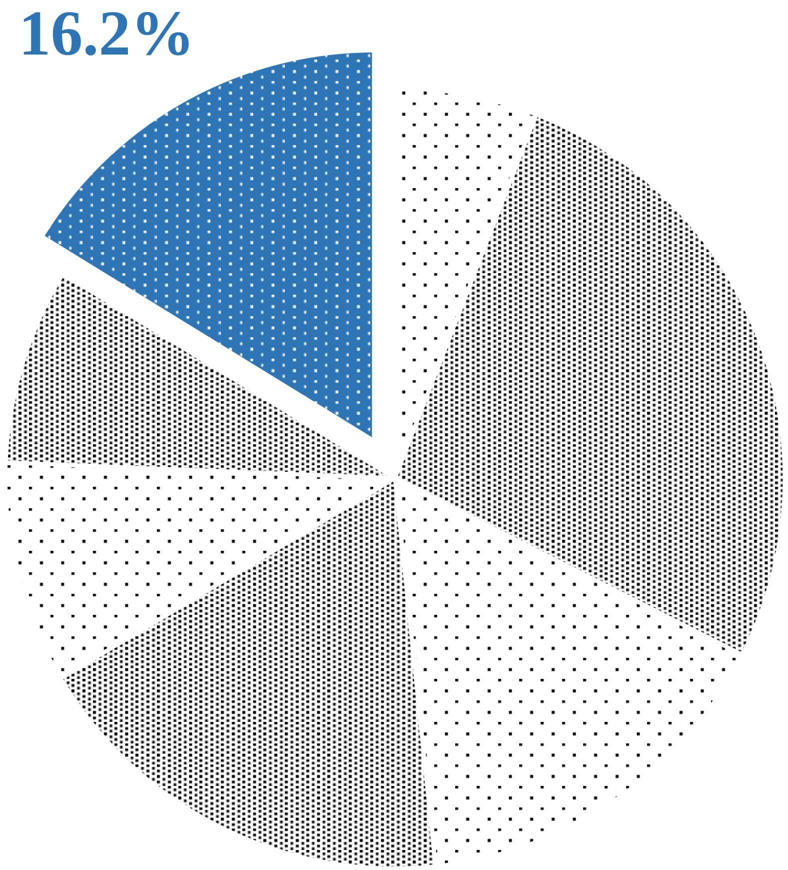
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6%)**
- **Inconsistent State Updates (18.1%)**
- **ID Uniqueness Violation (15.9%)**
- **Privilege Escalation (9.2%)**
- **Atomicity Violations (8.1%)**
- **Price Oracle Manipulation (5.8%)**
  - Manipulating external price authorities to exploit a contract's funds
  - Rank 1st regarding popularity in real-world
  - $44.8 million in first half of 2022



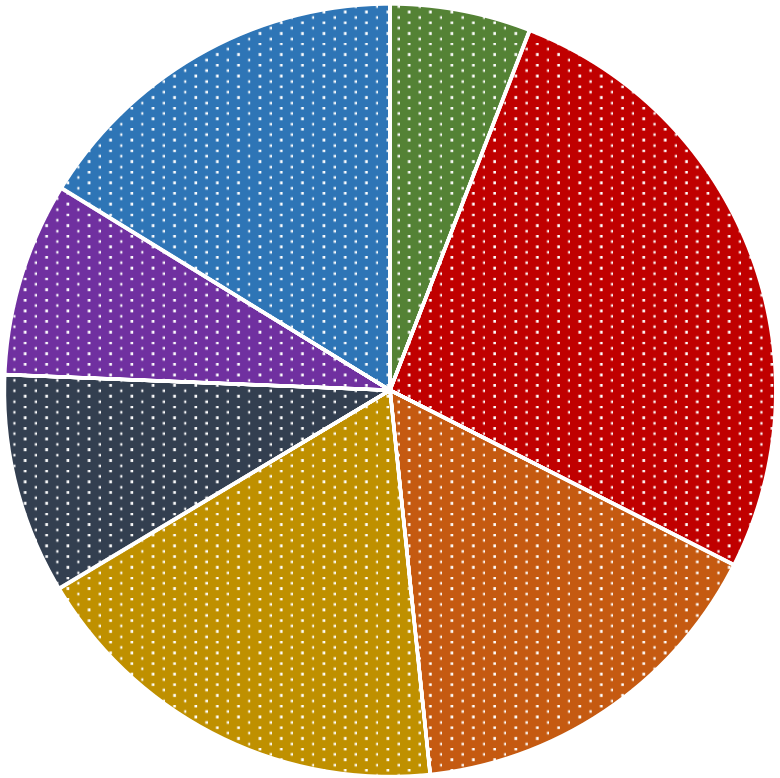
RQ2: What are the categories and distributions of machine unauditable bugs?

- **Erroneous accounting (26.6%)**
- **Inconsistent State Updates (18.1%)**
- **ID Uniqueness Violation (15.9%)**
- **Privilege Escalation (9.2%)**
- **Atomicity Violations (8.1%)**
- **Price Oracle Manipulation (5.8%)**
- **Contract-Specific Bugs (16.2%)**
  - Bugs and exploits that have a very low likelihood of appearing in other contracts



RQ2: What are the categories and distributions of machine unauditable bugs?

- **Price Oracle Manipulation (5.8%)**
- **Erroneous accounting (26.6%)**
- **ID Uniqueness Violation (15.9%)**
- **Inconsistent State Updates (18.1%)**
- **Privilege Escalation (9.2%)**
- **Atomicity Violations (8.1%)**
- **Contract-Specific Bugs (16.2%)**



Finding: Machine unauditable bugs can be classified to 7 categories, with around 85% are not project specific.

Take Away

- More than *80% of exploitable bugs are beyond existing tools*.
    - This is largely due to the lack in describing and checking the corresponding domain-specific properties (i.e., general testing *oracles*).
- The 80% of exploitable bugs that are beyond tools, called machine unauditable bugs (MUBs), can be classified into *7 categories*.
    - One of the categories (accounting for 16.2% of the MUBs) is project/implementation specific such that general oracles may not exist.
    - The remaining 6 categories have clear symptoms and can be properly abstracted such that automated oracles may be devised.

Our paper tries to raise the incentive of security researchers to develop automated oracles for machine unauditable bugs in smart contracts.

Other Findings in the Paper

- Majority of exploitable bugs in the wild are hard to find, including those within and beyond the scope of tools.
- Different types of MUBs have different distributions and different difficulty levels
  - Price oracle manipulation and privilege escalation are most popular in real-world exploits
  - Accounting errors are most popular in bugs found during audit contests
- MUBs are easy to fix, requiring 15 LoC on average.
- In our guided audit, we found ***15*** bugs, awarded around ***$150,000***

Related Works

- 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.
- W. Dingman, A. Cohen, N. Ferrara, A. Lynch, P. Jasinski, P. E. Black, and L. Deng, “Classification of smart contract bugs using the nist bugs framework,” in 2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA), 2019.
- P. Zhang, F. Xiao, and X. Luo, “A framework and dataset for bugs in ethereum smart contracts,” in 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2020.
- J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, “Defining smart contract defects on ethereum,” IEEE Transactions on Software Engineering, 2020.
- K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, “Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab,” in International conference on financial cryptography and data security. Springer, 2016.

Thanks!

Online Poster

zhan3299@purdue.edu
bzhangprogramming@gmail.com