What are the Actual Flaws in Important Smart Contracts (and How Can We Find Them)?

FC 2020

Authors

- Alex Groce (agroce@gmail.com) [NAU]
- Josselin Feist (josselin@trailofbits.com) [TOB]
- Gustavo Grieco (gustavo.grieco@trailofbits.com) [TOB]
- Michael Colburn (michael.colburn@trailofbits.com) [TOB]

[NAU] Northern Arizona University

[TOB] Trail of Bits

The problem

- **What are the vulnerabilities in production-ready code?**
  - Necessary to orient future researches
  - Few public datasets
  - Contracts deployed on blockchain are mostly unused and buggy
  - Projects rely on multiple contracts, difficult to identify all the onchain components
- **How many could be found without human-assistance?**
  - Help prioritizing program analysis research

Our approach

- Summary of 23 professional audits
- Categorization of 246 bugs
- Estimate the efficacy of a *perfect* automated vulnerability detector

Results

- Distribution of bugs is *similar* to traditional software
  - Reentrancy bugs are rarely seen
- Large % can be found automatically
- Many key issues cannot be found automatically
- No relation between high-quality unit tests and absence of serious vulnerabilities

The Dataset

Dataset

- **23 audits performed by Trail of Bits**
  - 17 public https://github.com/trailofbits/publications
- **Per codebase:**
  - 1 to a few dozen of contracts
  - 2 to 22 findings, median of 10
  - 1 - 12 person-weeks of effort, median 4
- **24 different auditors**
  - Mean of 2.6 per audit

Code review

- **Mix of manual analysis and automated tools**
  - Almost all used Static analysis (Slither)
  - ≈ 50% Symbolic Execution (Manticore) or Fuzzing (Echidna)

The Vulnerabilities

Vulnerabilities

Category	%	High-Low	Severity High	Severity Med.	Severity Low	Severity Info.	Severity Und.	Difficulty High	Difficulty Med.	Difficulty Low	Difficulty Und.
data validation	36%	11%	21%	36%	24%	13%	6%	27%	16%	55%	2%
access controls	10%	25%	42%	25%	12%	21%	0%	33%	12%	54%	0%
race condition	7%	0%	41%	41%	6%	12%	0%	100%	0%	0%	0%
numerics	5%	23%	31%	23%	38%	8%	0%	31%	8%	62%	0%
undefined behavior	5%	23%	31%	15%	31%	8%	15%	15%	8%	77%	0%
patching	7%	11%	17%	11%	39%	28%	6%	6%	11%	61%	22%
denial of service	4%	10%	20%	30%	30%	20%	0%	50%	0%	40%	10%
authentication	2%	25%	50%	25%	25%	0%	0%	50%	0%	50%	0%
reentrancy	2%	0%	50%	25%	25%	0%	0%	50%	25%	0%	25%
error reporting	3%	0%	29%	14%	0%	57%	0%	43%	29%	29%	0%
configuration	2%	0%	40%	0%	20%	20%	20%	60%	20%	20%	0%
logic	1%	0%	33%	33%	33%	0%	0%	100%	0%	0%	0%
data exposure	1%	0%	33%	33%	0%	33%	0%	33%	33%	33%	0%
timing	2%	25%	25%	0%	75%	0%	0%	75%	0%	25%	0%
coding-bug	2%	0%	0%	67%	33%	0%	0%	17%	0%	83%	0%
front-running	2%	0%	0%	80%	0%	20%	0%	100%	0%	0%	0%
auditing and logging	4%	0%	0%	0%	33%	44%	22%	33%	0%	56%	11%
missing-logic	1%	0%	0%	0%	67%	33%	0%	0%	0%	100%	0%
cryptography	0%	0%	0%	0%	100%	0%	0%	100%	0%	0%	0%
documentation	2%	0%	0%	0%	25%	50%	25%	0%	0%	75%	25%
API inconsistency	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%
code-quality	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%

Vulnerabilities

Category	%	High-Low	Severity High	Severity Med.	Severity Low	Severity Info.	Severity Und.	Difficulty High	Difficulty Med.	Difficulty Low	Difficulty Und.
data validation	36%	11%	21%	36%	24%	13%	6%	27%	16%	55%	2%
access controls	10%	25%	42%	25%	12%	21%	0%	33%	12%	54%	0%
race condition	7%	0%	41%	41%	6%	12%	0%	100%	0%	0%	0%
numerics	5%	23%	31%	23%	38%	8%	0%	31%	8%	62%	0%
undefined behavior	5%	23%	31%	15%	31%	8%	15%	15%	8%	77%	0%
patching	7%	11%	17%	11%	39%	28%	6%	6%	11%	61%	22%
denial of service	4%	10%	20%	30%	30%	20%	0%	50%	0%	40%	10%
authentication	2%	25%	50%	25%	25%	0%	0%	50%	0%	50%	0%
reentrancy	2%	0%	50%	25%	25%	0%	0%	50%	25%	0%	25%
error reporting	3%	0%	29%	14%	0%	57%	0%	43%	29%	29%	0%
configuration	2%	0%	40%	0%	20%	20%	20%	60%	20%	20%	0%
logic	1%	0%	33%	33%	33%	0%	0%	100%	0%	0%	0%
data exposure	1%	0%	33%	33%	0%	33%	0%	33%	33%	33%	0%
timing	2%	25%	25%	0%	75%	0%	0%	75%	0%	25%	0%
coding-bug	2%	0%	0%	67%	33%	0%	0%	17%	0%	83%	0%
front-running	2%	0%	0%	80%	0%	20%	0%	100%	0%	0%	0%
auditing and logging	4%	0%	0%	0%	33%	44%	22%	33%	0%	56%	11%
missing-logic	1%	0%	0%	0%	67%	33%	0%	0%	0%	100%	0%
cryptography	0%	0%	0%	0%	100%	0%	0%	100%	0%	0%	0%
documentation	2%	0%	0%	0%	25%	50%	25%	0%	0%	75%	25%
API inconsistency	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%
code-quality	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%

Vulnerabilities

Category	%	High-Low	Severity High	Severity Med.	Severity Low	Severity Info.	Severity Und.	Difficulty High	Difficulty Med.	Difficulty Low	Difficulty Und.
data validation	36%	11%	21%	36%	24%	13%	6%	27%	16%	55%	2%
access controls	10%	25%	42%	25%	12%	21%	0%	33%	12%	54%	0%
race condition	7%	0%	41%	41%	6%	12%	0%	100%	0%	0%	0%
numerics	5%	23%	31%	23%	38%	8%	0%	31%	8%	62%	0%
undefined behavior	5%	23%	31%	15%	31%	8%	15%	15%	8%	77%	0%
patching	7%	11%	17%	11%	39%	28%	6%	6%	11%	61%	22%
denial of service	4%	10%	20%	30%	30%	20%	0%	50%	0%	40%	10%
authentication	2%	25%	50%	25%	25%	0%	0%	50%	0%	50%	0%
reentrancy	2%	0%	50%	25%	25%	0%	0%	50%	25%	0%	25%
error reporting	3%	0%	29%	14%	0%	57%	0%	43%	29%	29%	0%
configuration	2%	0%	40%	0%	20%	20%	20%	60%	20%	20%	0%
logic	1%	0%	33%	33%	33%	0%	0%	100%	0%	0%	0%
data exposure	1%	0%	33%	33%	0%	33%	0%	33%	33%	33%	0%
timing	2%	25%	25%	0%	75%	0%	0%	75%	0%	25%	0%
coding-bug	2%	0%	0%	67%	33%	0%	0%	17%	0%	83%	0%
front-running	2%	0%	0%	80%	0%	20%	0%	100%	0%	0%	0%
auditing and logging	4%	0%	0%	0%	33%	44%	22%	33%	0%	56%	11%
missing-logic	1%	0%	0%	0%	67%	33%	0%	0%	0%	100%	0%
cryptography	0%	0%	0%	0%	100%	0%	0%	100%	0%	0%	0%
documentation	2%	0%	0%	0%	25%	50%	25%	0%	0%	75%	25%
API inconsistency	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%
code-quality	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%

Vulnerabilities

Category	%	High-Low	Severity					Difficulty			
			High	Med.	Low	Info.	Und.	High	Med.	Low	Und.
data validation	36%	11%	21%	36%	24%	13%	6%	27%	16%	55%	2%
access controls	10%	25%	42%	25%	12%	21%	0%	33%	12%	54%	0%
race condition	7%	0%	41%	41%	6%	12%	0%	100%	0%	0%	0%
numerics	5%	23%	31%	23%	38%	8%	0%	31%	8%	62%	0%
undefined behavior	5%	23%	31%	15%	31%	8%	15%	15%	8%	77%	0%
patching	7%	11%	17%	11%	39%	28%	6%	6%	11%	61%	22%
denial of service	4%	10%	20%	30%	30%	20%	0%	50%	0%	40%	10%
authentication	2%	25%	50%	25%	25%	0%	0%	50%	0%	50%	0%
reentrancy	2%	0%	50%	25%	25%	0%	0%	50%	25%	0%	25%
error reporting	3%	0%	29%	14%	0%	57%	0%	43%	29%	29%	0%
configuration	2%	0%	40%	0%	20%	20%	20%	60%	20%	20%	0%
logic	1%	0%	33%	33%	33%	0%	0%	100%	0%	0%	0%
data exposure	1%	0%	33%	33%	0%	33%	0%	33%	33%	33%	0%
timing	2%	25%	25%	0%	75%	0%	0%	75%	0%	25%	0%
coding-bug	2%	0%	0%	67%	33%	0%	0%	17%	0%	83%	0%
front-running	2%	0%	0%	80%	0%	20%	0%	100%	0%	0%	0%
auditing and logging	4%	0%	0%	0%	33%	44%	22%	33%	0%	56%	11%
missing-logic	1%	0%	0%	0%	67%	33%	0%	0%	0%	100%	0%
cryptography	0%	0%	0%	0%	100%	0%	0%	100%	0%	0%	0%
documentation	2%	0%	0%	0%	25%	50%	25%	0%	0%	75%	25%
API inconsistency	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%
code-quality	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%

Example: Data Validation

```
function forwardCall(address destination, bytes memory data)
public {
    (bool success, ) = destination.call(data);
    require(success);
```

Example: Access Control

```
function withdrawFromOwner() public // isOwner
{
    msg.sender.transfer(address(this).balance);
}
```

Vulnerabilities

Category	%	High-Low	Severity High	Severity Med.	Severity Low	Severity Info.	Severity Und.	Difficulty High	Difficulty Med.	Difficulty Low	Difficulty Und.
data validation	36%	11%	21%	36%	24%	13%	6%	27%	16%	55%	2%
access controls	10%	25%	42%	25%	12%	21%	0%	33%	12%	54%	0%
race condition	7%	0%	41%	41%	6%	12%	0%	100%	0%	0%	0%
numerics	5%	23%	31%	23%	38%	8%	0%	31%	8%	62%	0%
undefined behavior	5%	23%	31%	15%	31%	8%	15%	15%	8%	77%	0%
patching	7%	11%	17%	11%	39%	28%	6%	6%	11%	61%	22%
denial of service	4%	10%	20%	30%	30%	20%	0%	50%	0%	40%	10%
authentication	2%	25%	50%	25%	25%	0%	0%	50%	0%	50%	0%
reentrancy	2%	0%	50%	25%	25%	0%	0%	50%	25%	0%	25%
error reporting	3%	0%	29%	14%	0%	57%	0%	43%	29%	29%	0%
configuration	2%	0%	40%	0%	20%	20%	20%	60%	20%	20%	0%
logic	1%	0%	33%	33%	33%	0%	0%	100%	0%	0%	0%
data exposure	1%	0%	33%	33%	0%	33%	0%	33%	33%	33%	0%
timing	2%	25%	25%	0%	75%	0%	0%	75%	0%	25%	0%
coding-bug	2%	0%	0%	67%	33%	0%	0%	17%	0%	83%	0%
front-running	2%	0%	0%	80%	0%	20%	0%	100%	0%	0%	0%
auditing and logging	4%	0%	0%	0%	33%	44%	22%	33%	0%	56%	11%
missing-logic	1%	0%	0%	0%	67%	33%	0%	0%	0%	100%	0%
cryptography	0%	0%	0%	0%	100%	0%	0%	100%	0%	0%	0%
documentation	2%	0%	0%	0%	25%	50%	25%	0%	0%	75%	25%
API inconsistency	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%
code-quality	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%

Vulnerabilities

Category	%	High-Low	Severity: High	Med.	Low	Info.	Und.	Difficulty: High	Med.	Low	Und.
data validation	36%	11%	21%	36%	24%	13%	6%	27%	16%	55%	2%
access controls	10%	25%	42%	25%	12%	21%	0%	33%	12%	54%	0%
race condition	7%	0%	41%	41%	6%	12%	0%	100%	0%	0%	0%
numerics	5%	23%	31%	23%	38%	8%	0%	31%	8%	62%	0%
undefined behavior	5%	23%	31%	15%	31%	8%	15%	15%	8%	77%	0%
patching	7%	11%	17%	11%	39%	28%	6%	6%	11%	61%	22%
denial of service	4%	10%	20%	30%	30%	20%	0%	50%	0%	40%	10%
authentication	2%	25%	50%	25%	25%	0%	0%	50%	0%	50%	0%
reentrancy	2%	0%	50%	25%	25%	0%	0%	50%	25%	0%	25%
error reporting	3%	0%	29%	14%	0%	57%	0%	43%	29%	29%	0%
configuration	2%	0%	40%	0%	20%	20%	20%	60%	20%	20%	0%
logic	1%	0%	33%	33%	33%	0%	0%	100%	0%	0%	0%
data exposure	1%	0%	33%	33%	0%	33%	0%	33%	33%	33%	0%
timing	2%	25%	25%	0%	75%	0%	0%	75%	0%	25%	0%
coding-bug	2%	0%	0%	67%	33%	0%	0%	17%	0%	83%	0%
front-running	2%	0%	0%	80%	0%	20%	0%	100%	0%	0%	0%
auditing and logging	4%	0%	0%	0%	33%	44%	22%	33%	0%	56%	11%
missing-logic	1%	0%	0%	0%	67%	33%	0%	0%	0%	100%	0%
cryptography	0%	0%	0%	0%	100%	0%	0%	100%	0%	0%	0%
documentation	2%	0%	0%	0%	25%	50%	25%	0%	0%	75%	25%
API inconsistency	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%
code-quality	1%	0%	0%	0%	0%	100%	0%	0%	0%	100%	0%

Comparison with traditional software

Comparison to Non-Smart-Contract Audits

Category	#	%	Change	Category	#	%	Change
data validation	41	53%	-17%	patching	6	8%	-1%
denial of service	23	30%	-26%	authentication	5	6%	-4%
configuration	20	26%	-24%	timing	4	5%	-3%
data exposure	18	23%	-22%	numerics	2	3%	+3%
access controls	14	18%	-8%	auditing and logging	2	3%	+1%
cryptography	12	16%	-16%	race condition	1	1%	+6%
undefined behavior	7	9%	-4%	error reporting	1	1%	+2%

Comparison with 15 non-smart contracts audit from Trail of Bits

Comparison to Non-Smart-Contract Audits

Category	#	%	Change	Category	#	%	Change
data validation	41	53%	-17%	patching	6	8%	-1%
denial of service	23	30%	-26%	authentication	5	6%	-4%
configuration	20	26%	-24%	timing	4	5%	-3%
data exposure	18	23%	-22%	numerics	2	3%	+3%
access controls	14	18%	-8%	auditing and logging	2	3%	+1%
cryptography	12	16%	-16%	race condition	1	1%	+6%
undefined behavior	7	9%	-4%	error reporting	1	1%	+2%

- **Denial of service: mostly delegated to consensus**
- **Configuration: smaller configuration footprint**
- **Data exposure: data known to be public**

Comparison with 15 non-smart contracts audit from Trail of Bits

Automated detection

Optimistic Bug Finders

Category	% Dynamic	% Static	Category	% Dynamic	% Static
data validation	57%	22%	logic	0%	0%
access controls	50%	4%	data exposure	0%	0%
race condition	6%	59%	timing	50%	25%
numerics	46%	69%	coding-bug	67%	50%
undefined behavior	0%	31%	front-running	0%	0%
patching	17%	33%	auditing and logging	0%	38%
denial of service	40%	0%	missing-logic	67%	0%
authentication	25%	0%	cryptography	0%	100%
reentrancy	75%	100%	documentation	0%	0%
error reporting	29%	14%	API inconsistency	0%	0%
configuration	0%	0%	code-quality	0%	67%

- **Dynamic: ~36%**
  - 17 of the 27 High-Low plausibly detectable with properties testing
- **Static: ~26%**
  - Clients might have run Slither before the audit

Unit Tests

Unit Tests

- **No relation between unit tests presence and bugs found**
- **Intuition:**
  - Unit tests confirms *expectations* (i.e. the code works as expected in the normal context)
  - Vulnerabilities are edge-cases that the developers did not think about.

Threats to Validity

Threats to Validity

- **Only 23 reports**
- **Reports from one company**
    - Analyzed 19 and 18 reports from two other companies, similar results
- **Codebase varied in level of maturity**
    - But all were willing to pay for a professional audit

Conclusion

Conclusion

- **Problem**
  - Evaluate what bugs are present in production-ready contracts and how to find them
- **Our approach**
  - Analysis of 23 audits performed by professional security auditors
- **Our analysis**
  - Distribution of bugs is *similar* to traditional software
  - Large % can be found automatically, but not all
  - No relation between high-quality unit tests <> absence of security bugs

Trail of Bits: Crytic Prize

- **Build and maintain many open source tools**
  - Slither, Echidna, Manticore, evm-cfg-builder
  - https://github.com/crytic & https://github.com/trailofbits
- **Looking to support academic research**
  - Crytic Prize: $10k for best academic papers built on top of our tools
  - https://blog.trailofbits.com/2019/11/13/announcing-the-crytic-10k-research-prize/

TRAIL OF BITS