Static Exploration of Taint-Style Vulnerabilities Found by Fuzzing

Bhargava Shastry, Federico Maggi, Fabian Yamaguchi, Konrad Rieck, and Jean-Pierre Seifert

TU berlin

TREND MICRO™

Technische Universität Braunschweig

How It Started

- Spun afl-fuzz on Open vSwitch
  - Found **8** vulnerabilities
  - Responsibly disclosed and now patched
  - 1 **RCE**
    - Crashing input tweetable

ffffffffffff0000000000008847

Bottleneck

- OvS has over **100** *functional* test cases
    - Only **3-4** fuzzable
    - Test coverage **≤ 3%**

Duh, extensively write fuzzable test cases!

Problem

- Not faulting **OvS**, problem deep-rooted
- Writing fuzzable tests **challenging**
  - Applicability **limited**
  - Does not **scale**
  - Requires domain **expertise**

Fuzzing may not exercise every single LoC

Pitch

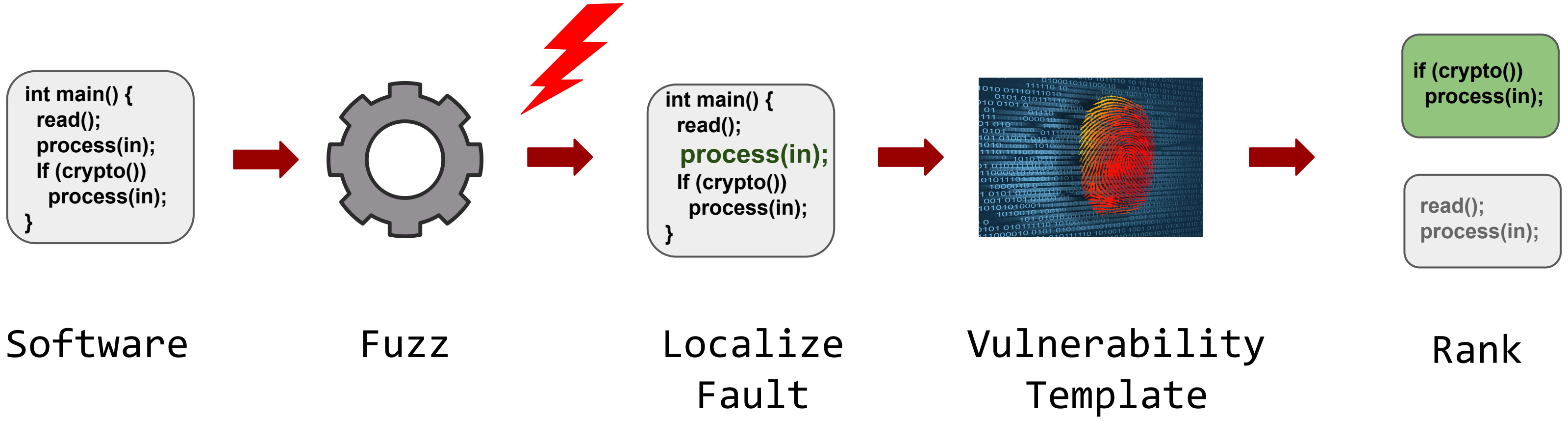
Fuzzer-directed **static analysis**

Proposal

Leverage hard data to ask the compiler specific questions

Fuzzer crash ⇒ Stack trace ⇒ **Vulnerability Template** ⇒ Recurrences

Design



```
int main() {
  read();
  process(in);
  If (crypto())
    process(in);
}
```

Software

Fuzz

```
int main() {
  read();
  process(in);
  If (crypto())
    process(in);
}
```

Localize Fault

Vulnerability Template

```
if (crypto())
  process(in);
```

```
read();
process(in);
```

Rank

Image: https://www.laserfiche.com/content/uploads/2015/02/shutterstock_137894381.jpg

Implementation

- Fault localization & Ranking ⇒ custom python script
- Template matching engine ⇒ **Clang libASTMatcher**

https://github.com/test-pipeline

Results: Effectiveness

Vulnerability	Num. matches	Num. issues
CVE-2016-10377	5	0
CVE-2017-9264 (TCP)	10	0
CVE-2017-9264 (UDP)	**2**	**1**
CVE-2017-9264 (IPv6)	3	0
CVE-2017-9214	41	0
CVE-2017-9263	34	0
CVE-2017-9265	1	0

Ranking Matches

- Reports provides insufficient context
- We rank matches based on fuzzer coverage
- Matches containing uncovered code **interesting**

Only **36** out of 96 matches ranked **high**

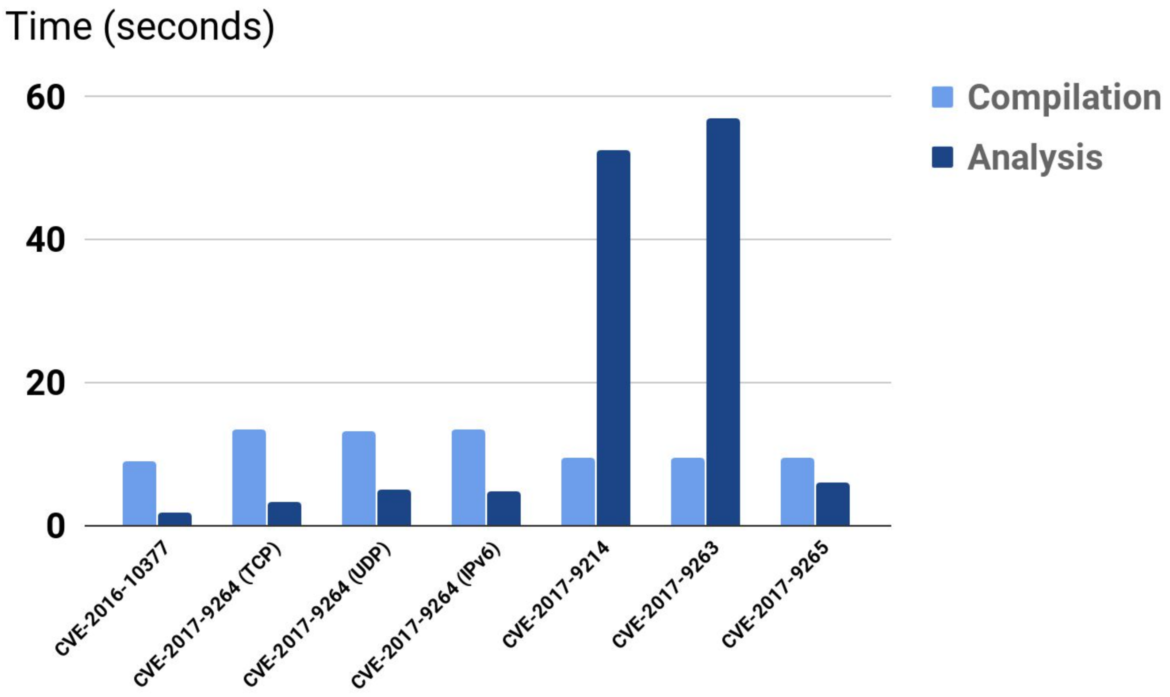
Insight

Developers want **contextual** information

“I would like to hear about other similar problem(s) you find in the code. Whether they are exploitable or not, it is better for the code to be careful.”

– Ben Pfaff, OvS lead developer

Results: Run time



Insight

- Structural (AST) analysis is relatively **fast**
- Semantic analysis is relatively **slow**
- Tension between analysis precision and speed
- Run time suitable for **continuous integration**

Summary

- Going beyond fuzzing is **necessary**
- **Static analysis** well-suited, results **promising**
- Evaluated on OvS, drew attention to **1 real issue** and **several corner cases**
- **Fast** enough for continuous integration

Future Work

- Reducing false positives
  - Formulating more **precise** vulnerability templates
- Easing manual review further
  - Use **Angr** for path reachability queries
  - Greetz to Dominic Maier

Acknowledgements

Thank OvS Security/Dev team for timely fixes

Questions?

Related Work

When vulnerable code pattern known

- **Code mining**
  - ○ Rely on security patches ⇒ **Reactive**

When vulnerable code pattern unknown

- **Machine learning**
  - ○ As good as training set ⇒ **Insufficient guarantees**