QFuzz: Quantitative Fuzzing for Side Channels

Yannic Noller

National University of Singapore

Saeid Tizpaz-Niari

UTEP

yannic.noller@acm.org
saeid@utep.edu

Detection / Quantification of Side-Channel Vulnerabilities

stringEquals (Original Jetty, v1)

```
boolean stringEquals(String s1, String s2) {
  if (s1 == s2)
    return true;
  if (s1 == null || s2 == null ||
      s1.length() != s2.length())
    return false;
  for (int i = 0; i < s1.length(); ++i)
    if (s1.charAt(i) != s2.charAt(i))
      return false;
  return true;
}
```

conditional early return
causes leakage

Side Channel Vulnerability

- **leackage** of **secret** data
- **software** side-channels
- **observables** (e.g., execution time)

Detection vs Quantification

Is there a **vulnerability**?

⇔

How much information can be leaked?

yannic.noller@acm.org
saeid@utep.edu

State of the Art

Challenges

How to go **beyond** non-interference?

How to avoid **expensive** symbolic execution?

How to **scale** to larger programs?

How to provide **guarantees** for vulnerability?

Blazer

Decomposition Instead of Self-Composition for k-Safety

Timos Antopoulos, Paul Gazzillo, Michael Hicks†, Eric Koskinen, Tachio Terauchi‡, and Shiyi Wei†
Yale University † University of Maryland ‡ JAIST

Themis

Precise Detection of Side-Channel Vulnerabilities using Quantitative Cartesian Hoare Logic

Jia Chen, University of Texas at Austin, Austin, Texas, jchen@cs.utexas.edu
Yu Feng, University of Texas at Austin, Austin, Texas, yufeng@cs.utexas.edu
Isil Dillig, University of Texas at Austin, Austin, Texas, isil@cs.utexas.edu

DiffFuzz

DIFFUZZ: Differential Fuzzing for Side-Channel Analysis

Shirin Nilizadeh*, *University of Texas at Arlington*, Arlington, TX, USA, shirin.nilizadeh@uta.edu
Yannic Noller*, *Humboldt-Universität zu Berlin*, Berlin, Germany, yannic.noller@hu-berlin.de
Corina S. Păsăreanu, *Carnegie Mellon University Silicon Valley, NASA Ames Research Center*, Moffett Field, CA, USA, corina.s.pasareanu@nasa.gov

MaxLeak

Multi-run side-channel analysis using Symbolic Execution and Max-SMT

Corina S. Păsăreanu, Carnegie Mellon University, NASA Ames, Moffett Field, CA, USA, corina.s.pasareanu@nasa.gov
Quoc-Sang Phan, Carnegie Mellon University, Moffett Field, CA, USA, sang.phan@sv.cmu.edu
Pasquale Malacaria, Queen Mary University of London, London, United Kingdom, p.malacaria@qmul.ac.uk

Quantification

Threat Model

Attacker can pick an **ideal public input** to compromise the **secret value** or some properties of it in **one try**.

Information Leakage: min-entropy [Smith2009]

Assuming that the program P is deterministic and the distribution over secret input Σ is uniform, then the information leakage can be characterized $\boldsymbol{log_2 k^*}$ (ε=0).

maximum number of classes in the cost observations

$\boldsymbol{log_2 |\Sigma_{Y=y^*}|}$

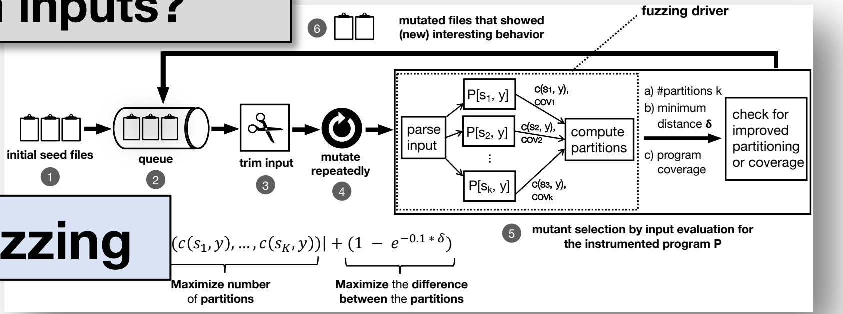
$\varepsilon \geq 0$

Problem Statement

Find set of secret values Σ and public value y* that characterize the maximum number of observation classes with the highest distance δ.

1 **How to identify such inputs?**

Fuzzing



2 **How to characterize observation classes?**

Partitioning Algorithm

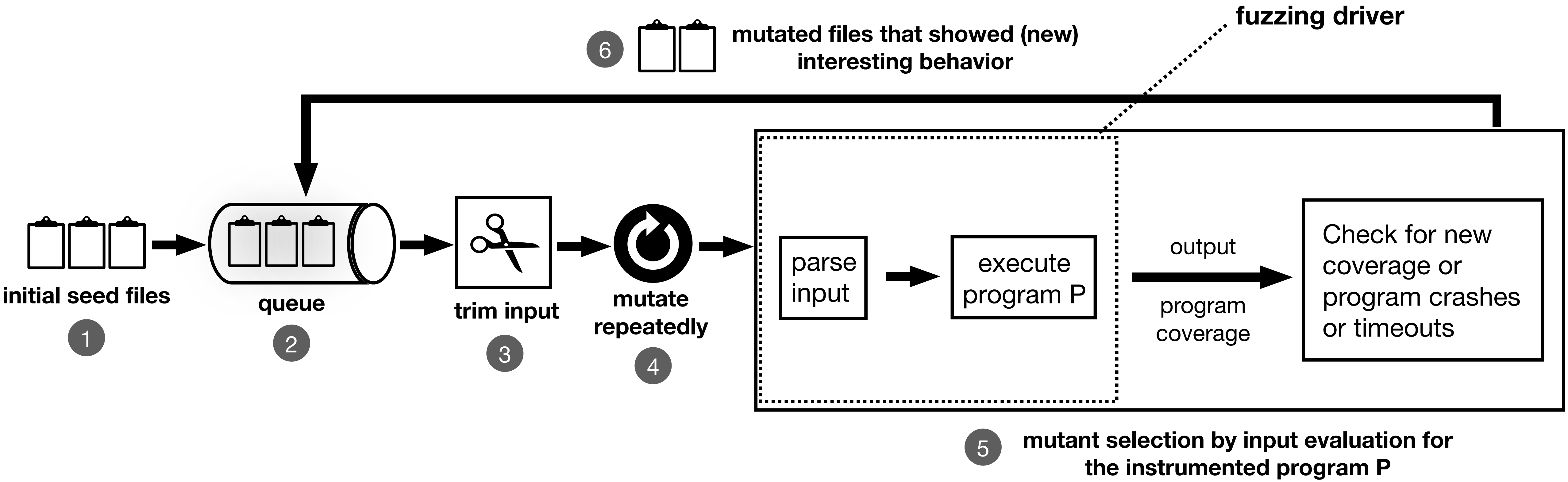
KDynamic & Greedy

$c(s_1, y)$, $c(s_2, y)$ → p_1

$c(s_3, y)$, $c(s_4, y)$ → p_2

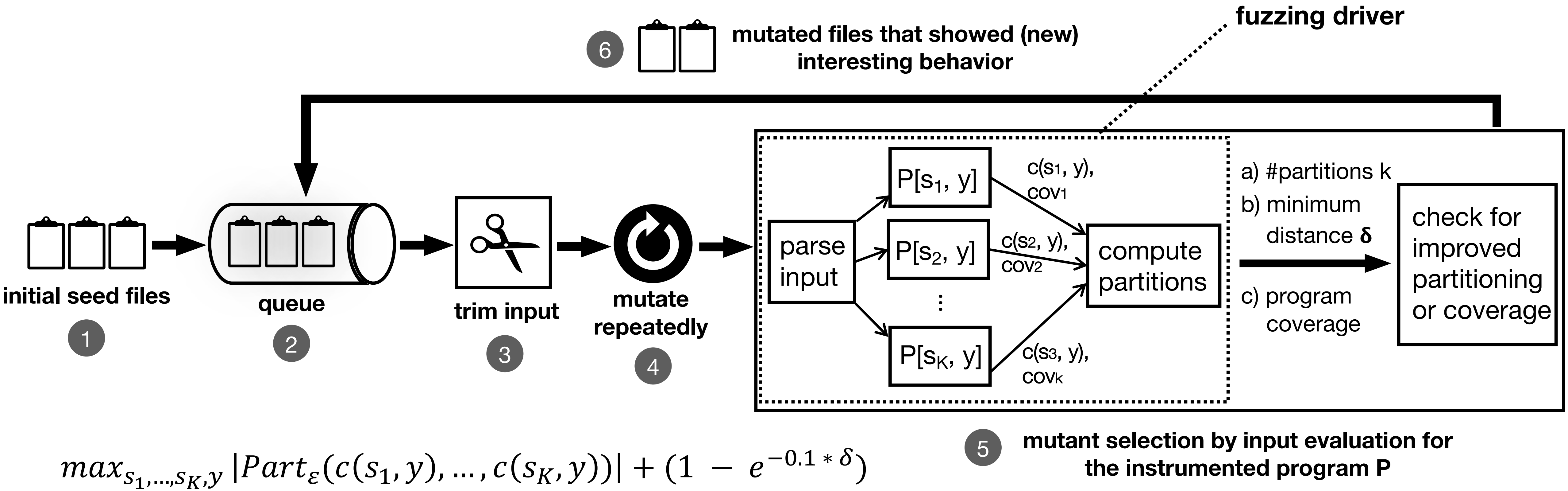
yannic.noller@acm.org
saeid@utep.edu

Background: Greybox Fuzzing



yannic.noller@acm.org
saeid@utep.edu

QFuzz: Workflow

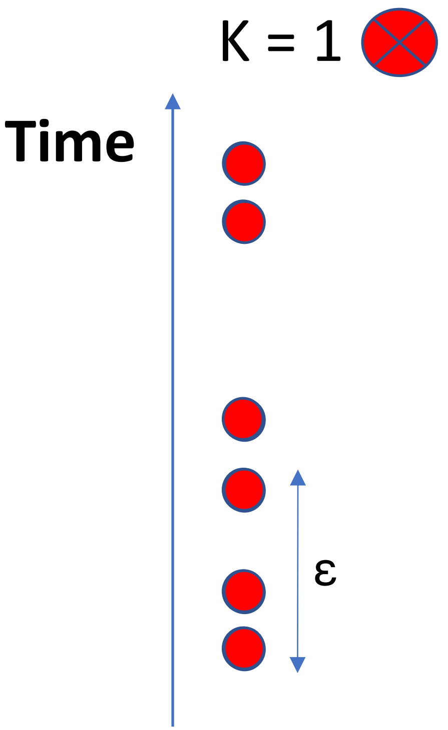


$$max_{s_1,\ldots,s_K,y} \, |Part_{\varepsilon}(c(s_1,y),\ldots,c(s_K,y))| + (1 - e^{-0.1 * \delta})$$

Maximize number of **partitions**

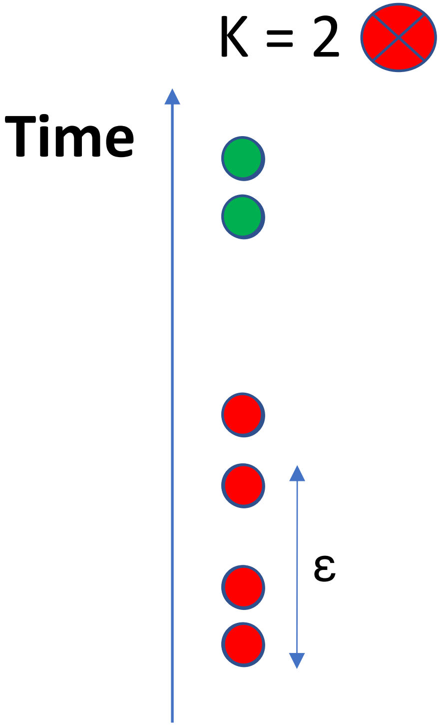
Maximize the **difference between** the **partitions**

KDynamic vs Greedy Partitioning



yannic.noller@acm.org
saeid@utep.edu

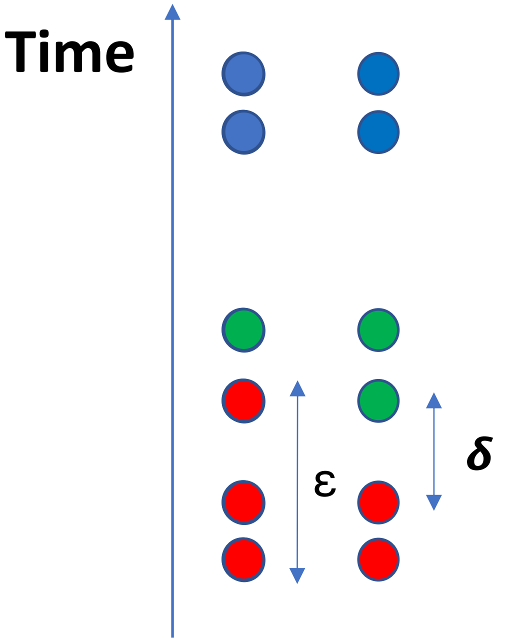
KDynamic vs Greedy Partitioning



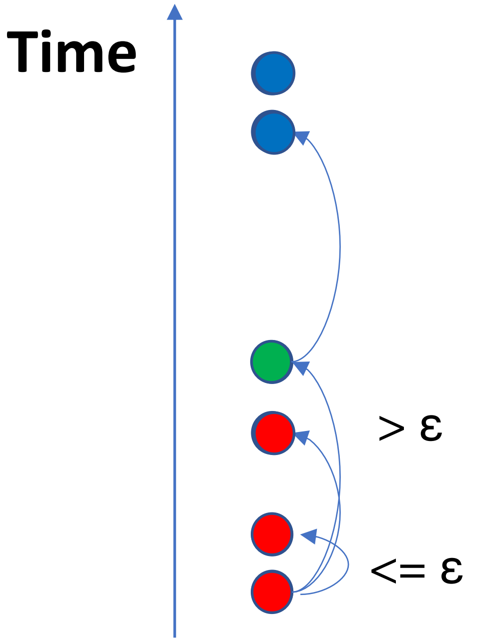
yannic.noller@acm.org
saeid@utep.edu

KDynamic vs Greedy Partitioning

K = 3



Find valid partitions with max. delta



Find valid partitions with guarantees; simple and fast

yannic.noller@acm.org
saeid@utep.edu

Example (K=100, ε=1, length=16, count=bytecode-instruction)

K=17
δ=3

stringEquals (Original Jetty, v1)

```
boolean stringEquals(String s1, String s2) {
  if (s1 == s2)
    return true;
  if (s1 == null || s2 == null ||
      s1.length() != s2.length())
    return false;
  for (int i = 0; i < s1.length(); ++i)
    if (s1.charAt(i) != s2.charAt(i))
      return false;
  return true;
}
```

stringEquals (Current Jetty, v4)

```
boolean stringEquals(String s1, String s2) {
  if (s1 == s2) return true;
  if (s1 == null || s2 == null)
    return false;
  boolean result = true;
  int l1 = s1.length();
  int l2 = s2.length();
  for (int i = 0; i < l2; ++i)
    result &= s1.charAt(i%l1) == s2.charAt(i));
  return result && l1 == l2;
}
```

K=9
δ =1

K=1

stringEquals (Safe Jetty, v5)

```
boolean stringEquals(String s1, String s2) {
  if (s1 == s2) return true;
  if (s1 == null || s2 == null)
    return false;
  int l1 = s1.length();
  int l2 = s2.length();
  if(l2 == 0){return l1 == 0}
  int result |= l1 - l2;
  for (int i = 0; i < l2; ++i){
    int r = ((i - l1) >>> 31) * i;
    result |= s1.charAt(r) ^ s2.charAt(i);
  }
  return result == 0;
}
```

Equals (Unsafe Spring-Security)

```
boolean Equals(String s1, String s2) {
  if (s1 == null || s2 == null)
    return false;
  byte[] s1B = s1.getBytes("UTF-8");
  byte[] s2B = s2.getBytes("UTF-8");
  int len1 = s1B.length;
  int len2 = s2B.length;
  if (len1 != len2)
    return false;
  int result = 0;
  for (int i = 0; i < len2; i++)
    result |= s1B[i] ^ s2B[i];
  return result == 0;
}
```

K=2
δ =149

DifFuzz

only leaks existence of special character

yannic.noller@acm.org
saeid@utep.edu

Evaluation

Research Questions

RQ1 Which partitioning algorithm (*KDynamic* or *Greedy*) performs better in terms of correct number of partitions and time for partition computation?

RQ2 How does QFuzz compare with state-of-the-art SC detection techniques like Blazer, Themis, and DiffFuzz?

RQ3 Can QFuzz be used for quantification of SC vulnerabilities in real-world Java applications and how does it compare with MaxLeak?

Subjects

- Micro-benchmark
- DARPA STAC
- GitHub projects

Tools/Techniques

- Blazer
- Themis
- DiffFuzz
- MaxLeak

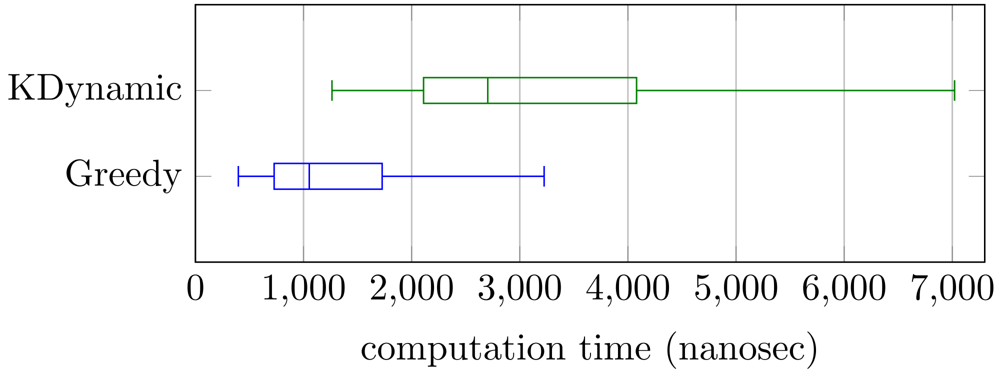
Our **open-source** tool **QFuzz** and all experimental subjects are **publicly available**:

https://github.com/yannicnoller/qfuzz

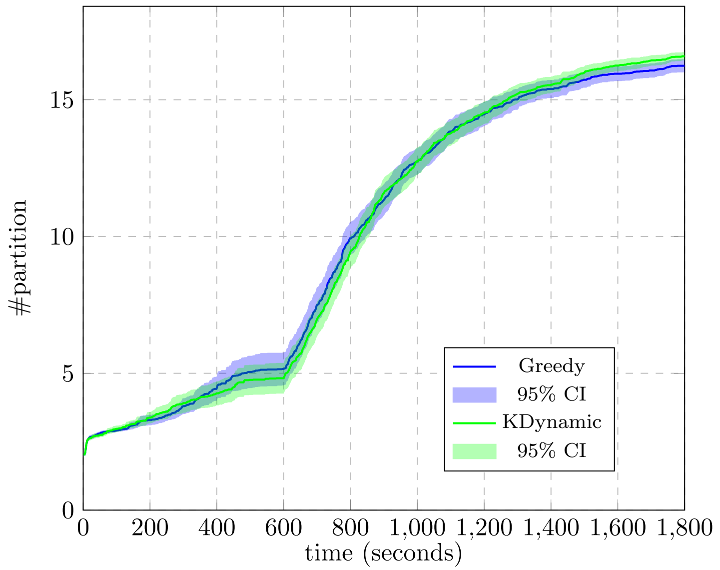
http://doi.org/10.5281/zenodo.4722965

yannic.noller@acm.org
saeid@utep.edu

RQ1 KDynamic vs. Greedy



Computational complexity of **Greedy** vs **KDynamic** in isolation



Eclipse Jetty (ε=1): temporal development **Greedy** and **KDynamic** with **5 different seed inputs combined**

(lines and bands show averages and 95% confidence intervals across 30 repetitions)

yannic.noller@acm.org
saeid@utep.edu

RQ2 Detection

Benchmark	Version	QFuzz		DifFuzz	Time (s)			
		p_{max}	δ_{max}	δ_{max}	QFuzz, $\overline{p} > 1$	DifFuzz, $\overline{\delta} > 0$	Blazer	Themis
Array	Safe	1	0	1	-	7.40 (+/- 1.21)	1.60	0.28
Array	Unsafe	2	192	195	5.70 (+/- 0.21)	7.40 (+/- 0.93)	0.16	0.23
LoopAndbranch	*Safe*	*2*	*4*	*4,278,268,702*	*1045.33 (+/- 43.51)*	*18.60 (+/- 6.40)*	*0.23*	*0.33*
LoopAndbranch	Unsafe	2	4	4,294,838,782	1078.63 (+/- 61.04)	10.60 (+/- 2.62)	0.65	0.16
Sanity	Safe	1	0	0	-	-	0.63	0.41
Sanity	Unsafe	2	3,537,954,539	4,290,510,883	1414.13 (+/- 102.27)	163 (+/- 40.63)	0.30	0.17
Straightline	Safe	1	0	0	-	-	0.21	0.49
Straightline	Unsafe	2	8	8	7.47 (+/- 0.18)	14.60 (+/- 6.53)	22.20	5.30
unixlogin	Safe	-	-	3	-	510 (+/- 91.18)	0.86	-
unixlogin	Unsafe	2	6,400,000,008	3,200,000,008	1784.47 (+/- 21.27)	464.20 (+/- 64.61)	0.77	-
modPow1	Safe	1	0	0	-	-	1.47	0.61
modPow1	Unsafe	22	117	3,068	4.73 (+/- 0.16)	4.80 (+/- 1.11)	218.54	14.16
modPow2	Safe	1	0	9	-	-	1.62	0.75
modPow2	Unsafe	31	1	5,206	294.70 (+/- 104.66)	23.00 (+/- 3.48)	7813.68	141.36
passwordEq	Safe	1	0	0.00	-	-	2.70	1.10
passwordEq	Unsafe	93	2	127	4.57 (+/- 0.22)	8.60 (+/-2.11)	1.30	0.39
k96	Safe	1	0	0	-	-	0.70	0.61
k96	Unsafe	93	2	3,087,339	4.57 (+/- 0.22)	3.40 (+/- 0.98)	1.29	0.54
gpt14	*Safe*	*12*	*1*	*517*	*5.00 (+/- 0.00)*	*4.20 (+/- 0.80)*	*1.43*	*0.46*
gpt14	Unsafe	92	2	12,965,890	5.87 (+/- 0.12)	4.40 (+/- 1.03)	219.30	1.25
login	Safe	1	0	0	-	-	1.77	0.54
login	Unsafe	17	2	62	7.77 (+/- 0.69)	10.00 (+/- 2.92)	1.79	0.70

same vulnerabilities detected

additional information about the **strength** of leaks and the **exploitability** of vulnerabilities

large values for K may slow down **QFuzz**, but eventually, enable the exploration of many partitions

RQ3 Quantification

Modulo	Len	#Partitions	QFuzz (ϵ=0, 1h) $\overline{p}$	p_{max}	Time (s): p_{max}	t_{min}	MaxLeak (default) #Obs	Time (s)	MaxLeak (No solver) #Obs	Time (s)
1717	3	7	7.00 (+/- 0.00)	7	1.00 (+/- 0.00)	1	6	20.892	9	1.047
1717	4	10	10.00 (+/- 0.00)	10	7.43 (+/- 0.45)	5	9	152.332	12	1.370
1717	5	13	13.00 (+/- 0.00)	13	20.40 (+/- 3.87)	6	12	839.788	15	2.916
1717	6	16	16.00 (+/- 0.00)	16	294.60 (+/- 53.17)	22	15	**3731.328**	18	8.006
1717	7	19	18.37 (+/- 0.25)	19	2484.30 (+/- 451.42)	385		**> 4 h**	21	19.241
1717	8	22	20.43 (+/- 0.45)	22	3168.07 (+/- 303.47)	508		**> 4 h**	24	91.821
1717	9	25	22.20 (+/- 0.36)	*24*	3489.03 (+/- 169.19)	1009		**> 4 h**		> 8 GB
1717	10	28	24.40 (+/- 0.49)	*27*	3548.63 (+/- 57.73)	2929		**> 4 h**		> 8 GB
834443	3	7	7.00 (+/- 0.00)	7	13.40 (+/- 1.96)	8	6	7.416	9	1.188
834443	4	10	10.00 (+/- 0.00)	10	40.33 (+/- 12.14)	6	9	42.684	12	1.385
834443	5	13	12.93 (+/- 0.09)	13	645.70 (+/- 329.43)	74	12	215.929	15	2.953
834443	6	16	15.40 (+/- 0.20)	16	2711.87 (+/- 433.23)	271	15	936.921	18	7.511
834443	7	19	16.80 (+/- 0.33)	18	3227.60 (+/- 275.29)	952	18	**4021.150**	21	19.068
834443	8	22	17.93 (+/- 0.54)	22	3556.70 (+/- 83.44)	2301		**> 4 h**	24	96.360
834443	9	25	20.13 (+/- 0.59)	*24*	3572.83 (+/- 37.16)	3110		**> 4 h**		> 8 GB
834443	10	28	21.83 (+/- 0.46)	*24*	3504.13 (+/- 121.70)	1845		**> 4 h**		> 8 GB
1964903306	3	7	6.47 (+/- 0.18)	7	2228.30 (+/- 542.13)	119	6	12.167	9	1.085
1964903306	4	10	8.67 (+/- 0.19)	10	3494.30 (+/- 203.69)	429	9	70.805	12	1.535
1964903306	5	13	10.70 (+/- 0.19)	12	3594.00 (+/- 11.56)	3420	12	2306.261	15	3.391
1964903306	6	16	12.90 (+/- 0.11)	*13*	1337.90 (+/- 443.89)	206		**> 4 h**	18	7.506
1964903306	7	19	14.10 (+/- 0.27)	*15*	2984.67 (+/- 362.05)	503		**> 4 h**	21	19.486
1964903306	8	22	15.33 (+/- 0.36)	*17*	3398.37 (+/- 204.45)	1411		**> 4 h**	24	98.325
1964903306	9	25	16.30 (+/- 0.51)	*19*	3562.33 (+/- 54.24)	2819		**> 4 h**		> 8 GB
1964903306	10	28	17.30 (+/- 0.48)	*20*	3559.67 (+/- 77.72)	2390		**> 4 h**		> 8 GB

due to its dynamic analysis, **QFuzz** is **more scalable** than **MaxLeak**

QFuzz has precision comparable to **MaxLeak** that uses symbolic execution with model counting

even for complex scenarios **QFuzz** provides reasonable **lower-bound guarantees**

yannic.noller@acm.org
saeid@utep.edu

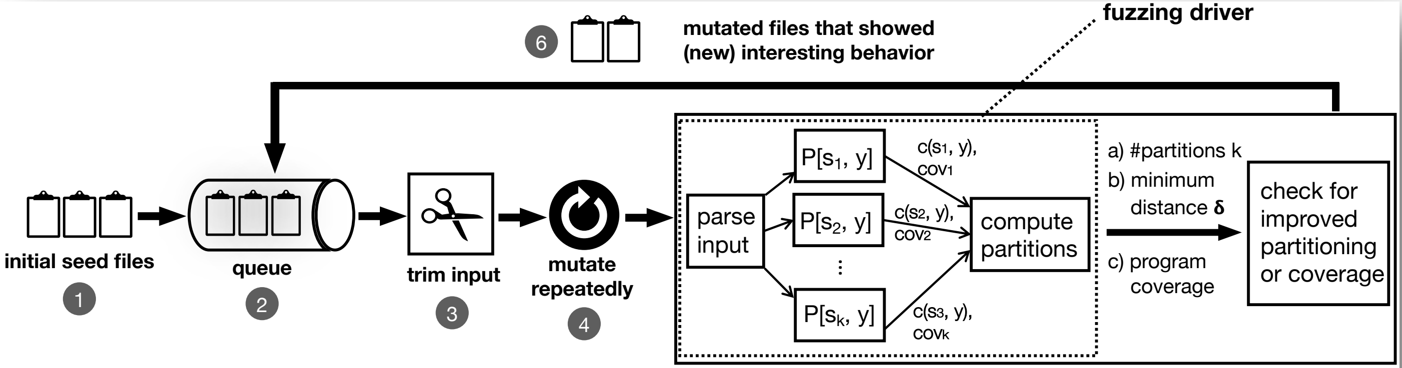
QFuzz: Quantitative Fuzzing for Side Channels

ISSTA 2021

stringEquals (Original Jetty, v1)

```
boolean stringEquals(String s1, String s2) {
  if (s1 == s2)
    return true;
  if (s1 == null || s2 == null ||
      s1.length() != s2.length())
    return false;
  for (int i = 0; i < s1.length(); ++i)
    if (s1.charAt(i) != s2.charAt(i))
      return false;
  return true;
}
```

conditional early return causes leakage



$$max_{s_1,\dots,s_K,y}\ |Part_\varepsilon(c(s_1,y),\dots,c(s_K,y))| + (1 - e^{-0.1*\delta})$$

Maximize number of **partitions** — **Maximize** the difference **between** the partitions

Detection vs Quantification

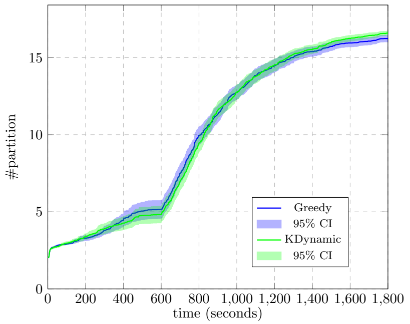
Is there a **vulnerability**?

⇔

How much information can be leaked?

Challenges

- How to go **beyond** non-interference?
- How to avoid **expensive** symbolic execution?
- How to **scale** to larger programs?
- How to provide **guarantees** for vulnerability?



additional information about the **strength** of leaks and the **exploitability** of vulnerabilities

due to its dynamic analysis, **QFuzz** is **more scalable** than **MaxLeak**

QFuzz has precision comparable to **MaxLeak** that uses symbolic execution with model counting

even for complex scenarios **QFuzz** provides reasonable **lower-bound guarantees**

yannic.noller@acm.org
saeid@utep.edu