Investigating the Effects of T-Wise Interaction Sampling for Vulnerability Discovery in Highly-Configurable Software Systems

Meeting on Feature-Oriented Software Development 2025

Tim Bächle, Erik Hofmayer, Christoph König, Tobias Pett, Ina Schaefer | 25. March 2025

Test, Validation, Analysis

Background

```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7      #ifdef CONFIG_SEND_DATA
8          sink(y);
9      #endif
10 #endif
11         // ...
12     }
13 }
```

A variable C function inspired by the example provided by Yamaguchi et al. [Yam+14]

Highly-Configurable Software Systems

- Often implemented as Software Product Lines (SPLs)
- Common core and variable features
- Product-based strategy common for analysis [Lie+13; Thü+14]

Background

```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7      #ifdef CONFIG_SEND_DATA
8          sink(y);
9      #endif
10 #endif
11         // ...
12     }
13 }
```

A variable C function inspired by the example provided by Yamaguchi et al. [Yam+14]

Highly-Configurable Software Systems

- Often implemented as Software Product Lines (SPLs)
- Common core and variable features
- Product-based strategy common for analysis [Lie+13; Thü+14]

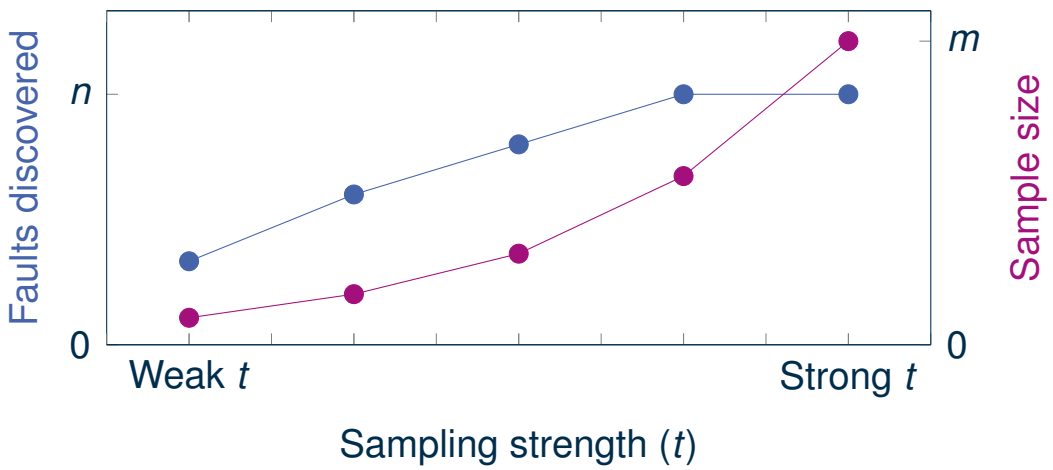
T-Wise Interaction Sampling

- **Idea**: All possible t-tuples of features appear in at least one sampled configuration
- t represents the interaction strength

Interaction Strength	PROCESS_INPUT	SEND_DATA
$t = 1$	✓	✓
	✗	✗
$t = 2$	✓	✓
	✗	✗
	✓	✗
	✗	✓

Background / Motivation

Observations of Medeiros et al. [Med+16] and Halin et al. [Hal+19]



Do these insights extend to the special class of faults/bugs that represent vulnerabilities?

T-Wise Interaction Sampling

- **Idea**: All possible t-tuples of features appear in at least one sampled configuration
- t represents the interaction strength

Interaction Strength	PROCESS_INPUT	SEND_DATA
$t = 1$	✓	✓
	×	×
$t = 2$	✓	✓
	×	×
	✓	×
	×	✓

Vulnerabilities in Highly-Configurable Systems

Inspiration: Bugs in Configurable Software

Variability Bug: A bug that occurs in one or more but not all configurations of a configurable system [Aba+17; ABW14; Mor+19]

Feature-Interaction Bug: A variability bug whose occurrence requires the interaction of at least two features [Aba+17; ABW14; GC11]

Vulnerabilities in Highly-Configurable Systems

Inspiration: Bugs in Configurable Software

Variability Bug: A bug that occurs in one or more but not all configurations of a configurable system [Aba+17; ABW14; Mor+19]

Feature-Interaction Bug: A variability bug whose occurrence requires the interaction of at least two features [Aba+17; ABW14; GC11]

Variability-Induced Vulnerability (VIV)

A **vulnerability** that is **present in some but not all configurations** of a configurable system.

```
1  void foo() {
2      int x = source(); // Attacker-controlled.
3      if(x < MAX){ // Does not enforce x >= 0.
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7          sink(y); // Security-sensitive operation.
8  #endif
9          // ...
10     }
11 }
```

A variable C function inspired by the example provided by Yamaguchi et al. [Yam+14]

Vulnerabilities in Highly-Configurable Systems

Inspiration: Bugs in Configurable Software

Variability Bug: A bug that occurs in one or more but not all configurations of a configurable system [Aba+17; ABW14; Mor+19]

Feature-Interaction Bug: A variability bug whose occurrence requires the interaction of at least two features [Aba+17; ABW14; GC11]

Feature-Interaction Vulnerability (FIV)

A **variability-induced vulnerability (VIV)** whose **presence is dependent** on the **interaction of two or more features'** selection or unselection.

```
1  void foo() {
2      int x = source(); // Attacker-controlled.
3      if(x < MAX){ // Does not enforce x >= 0.
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7      #ifdef CONFIG_SEND_DATA
8          sink(y); // Security-sensitive operation.
9      #endif
10 #endif
11         // ...
12     }
13 }
```

A variable C function inspired by the example provided by Yamaguchi et al. [Yam+14]

Sample-Based Vulnerability Discovery

Vari-Joern

- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]

Sample-Based Vulnerability Discovery

Vari-Joern

- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]

Sample-Based Vulnerability Discovery

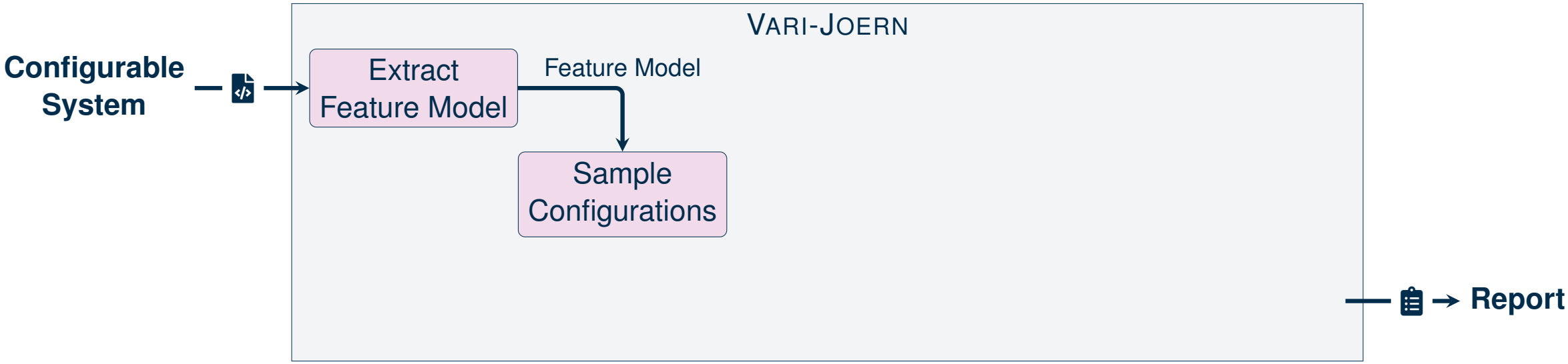
Vari-Joern

- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]

Sample-Based Vulnerability Discovery

Vari-Joern

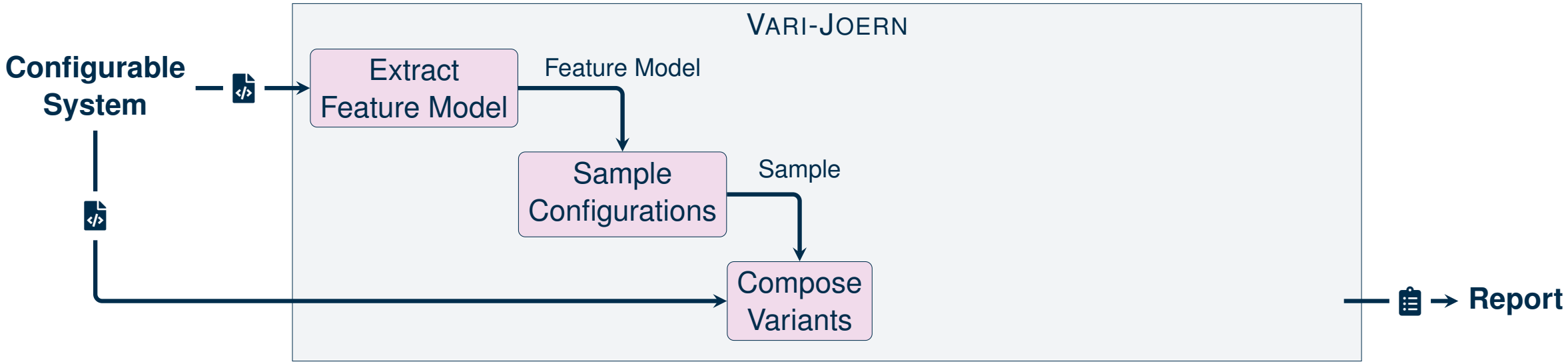
- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]



Sample-Based Vulnerability Discovery

Vari-Joern

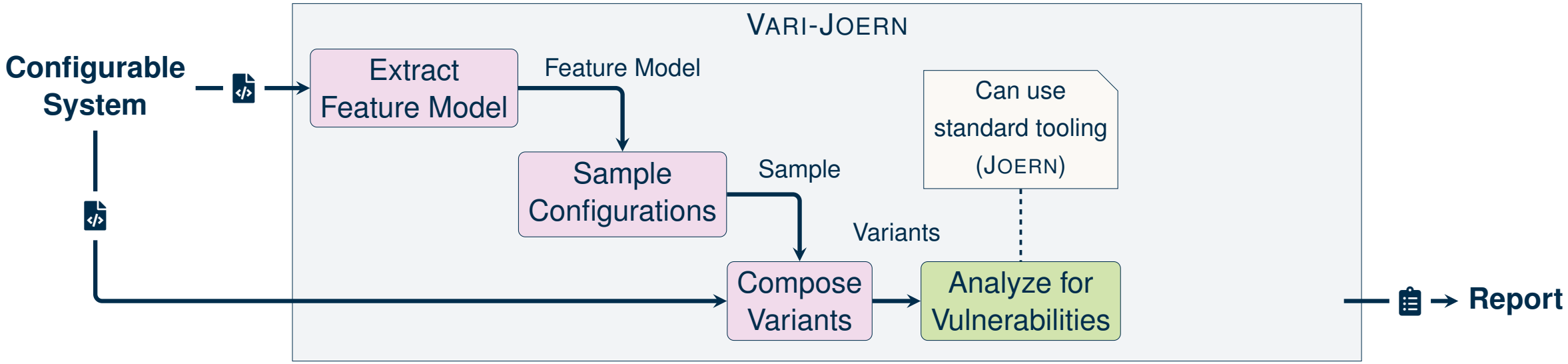
- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]



Sample-Based Vulnerability Discovery

Vari-Joern

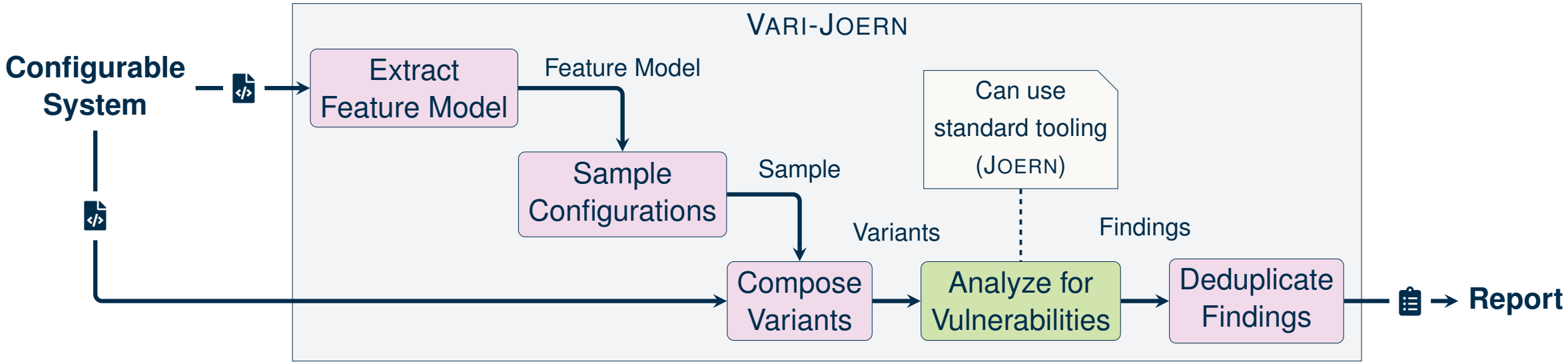
- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]



Sample-Based Vulnerability Discovery

Vari-Joern

- An analysis platform **realizing sample-based vulnerability discovery**
- **Built around** the static source code analysis tool **JOERN** [24c; Yam+14]



Research Questions

Motivation $\mathbf{RQ_1}$

Previous work [Hal+19; Med+16] showed:

1. Greater interaction sampling strength leads to the identification of more variability bugs
2. Sample size rapidly increases

⇒ Do these insights extend to the special class of bugs that are vulnerabilities?

Research Questions

Motivation RQ_1

Previous work [Hal+19; Med+16] showed:

1. Greater interaction sampling strength leads to the identification of more variability bugs
2. Sample size rapidly increases

⇒ Do these insights extend to the special class of bugs that are vulnerabilities?

RQ_1: What is the **trade-off between** the **sample size** and the **number of findings** when **using t-wise interaction sampling** of different strength **for vulnerability discovery** in highly-configurable software?

Research Questions

Motivation $\mathbf{RQ_1}$

- Previous work [Hal+19; Med+16] showed:
  1. Greater interaction sampling strength leads to the identification of more variability bugs
  2. Sample size rapidly increases

⇒ Do these insights extend to the special class of bugs that are vulnerabilities?

Motivation $\mathbf{RQ_2}$

- Interaction sampling of strength t should reliably identify FIVs of interaction degree $\leq t$

⇒ Is this the case in reality, or are FIV identified by:
- Lesser interction sampling strength due to coincidence?
- Greater interaction sampling strength due to a discrepancy between problem and solution space?

$\mathbf{RQ_1}$: What is the **trade-off between** the **sample size** and the **number of findings** when **using t-wise interaction sampling** of different strength **for vulnerability discovery** in highly-configurable software?

Research Questions

Motivation $\mathbf{RQ_1}$

- Previous work [Hal+19; Med+16] showed:
  1. Greater interaction sampling strength leads to the identification of more variability bugs
  2. Sample size rapidly increases

⇒ Do these insights extend to the special class of bugs that are vulnerabilities?

Motivation $\mathbf{RQ_2}$

- Interaction sampling of strength t should reliably identify FIVs of interaction degree $\leq t$

⇒ Is this the case in reality, or are FIV identified by:
  - Lesser interction sampling strength due to coincidence?
  - Greater interaction sampling strength due to a discrepancy between problem and solution space?

$\mathbf{RQ_1}$: What is the **trade-off between** the **sample size** and the **number of findings** when **using t-wise interaction sampling** of different strength **for vulnerability discovery** in highly-configurable software?

$\mathbf{RQ_2}$: Can the **increase in vulnerability warnings** reported using stronger t-wise interaction sampling be **attributed to** potential **FIVs of stronger interaction strength** being identified?

Experimental Setup

Subject Systems

- Real-world systems
- Significant share implemented in C
- Configuration management via KCONFIG

Name	Version	C-LoC	#Features
AXTLS [24a]	2.1.5	17,556	63
FIASCO [25]	Commit 4076045	46,013	99
BUSYBOX [24b]	1.36.1	182,966	1,027

Experimental Setup

Subject Systems

- Real-world systems
- Significant share implemented in C
- Configuration management via KCONFIG

Name	Version	C-LoC	#Features
AXTLS [24a]	2.1.5	17,556	63
FIASCO [25]	Commit 4076045	46,013	99
BUSYBOX [24b]	1.36.1	182,966	1,027

RQ_1 - Trade-off: Sample Size $\Leftrightarrow$ Findings

- Perform an analysis with VARI-JOERN for the subject systems
- Compare sample sizes and number of findings for different sampling strengths (t)
- Repeat accross 10 analysis runs

Experimental Setup

Subject Systems

- Real-world systems
- Significant share implemented in C
- Configuration management via KCONFIG

Name	Version	C-LoC	#Features
AXTLS [24a]	2.1.5	17,556	63
FIASCO [25]	Commit 4076045	46,013	99
BUSYBOX [24b]	1.36.1	182,966	1,027

RQ_1 - Trade-off: Sample Size ⇔ Findings

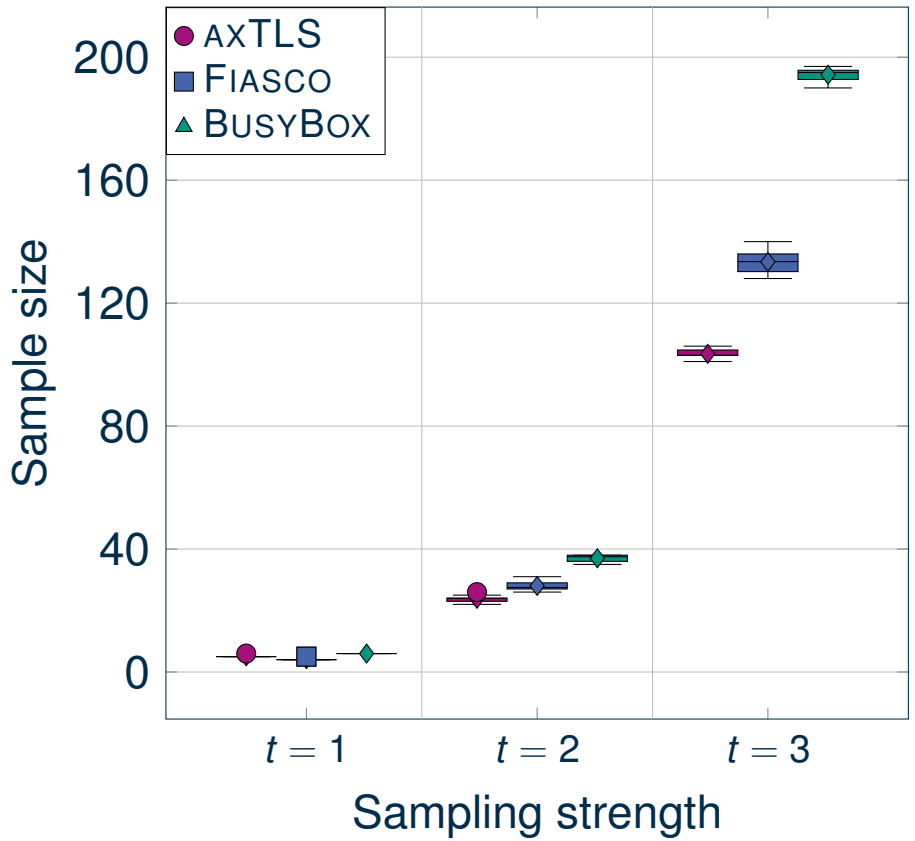
- Perform an analysis with VARI-JOERN for the subject systems
- Compare sample sizes and number of findings for different sampling strengths (t)
- Repeat accross 10 analysis runs

RQ_2 - Reason for Increase in Findings

- Focus on BUSYBOX
- Analyze presence condition of vulnerability warnings not identified by weaker sampling
- Compare required sampling strength (t) and true interaction strength of vulnerability warnings

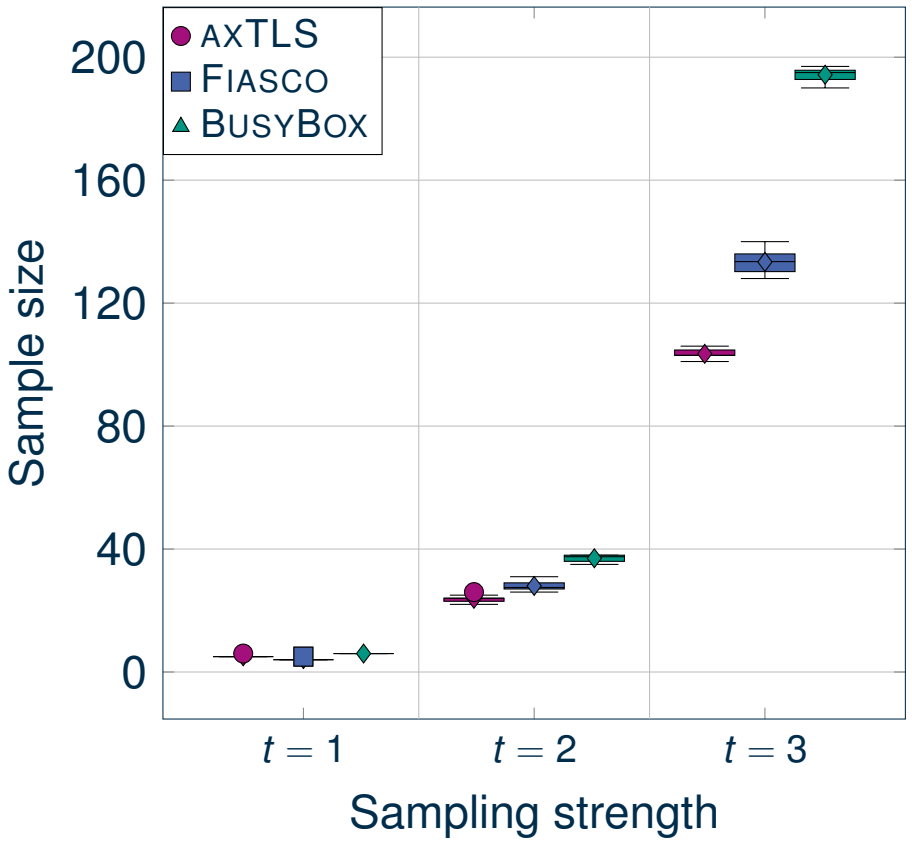
First Results

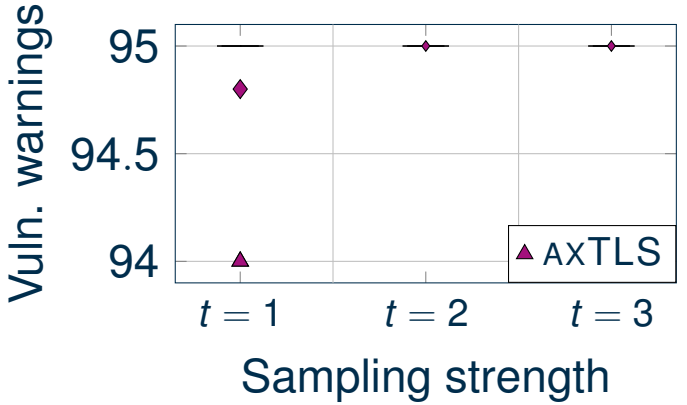
RQ_1 - Trade-off: Sample Size $\Leftrightarrow$ Findings

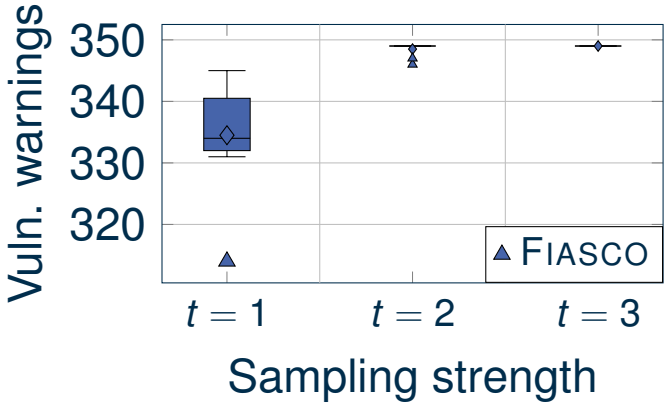


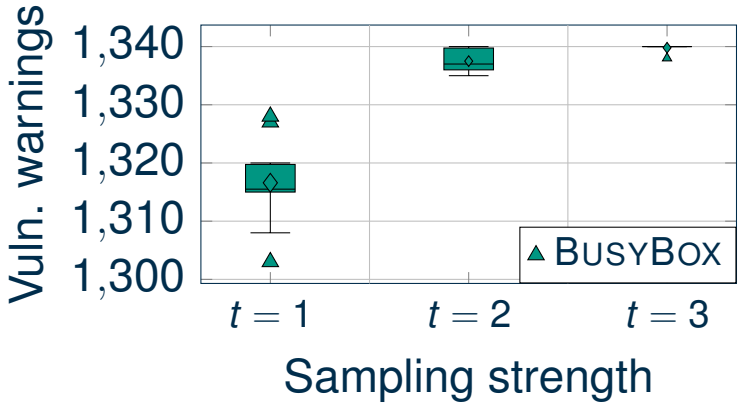
First Results

RQ_1 - Trade-off: Sample Size ⇔ Findings









First Results

RQ_2 - Reason for Increase in Findings

Expectations

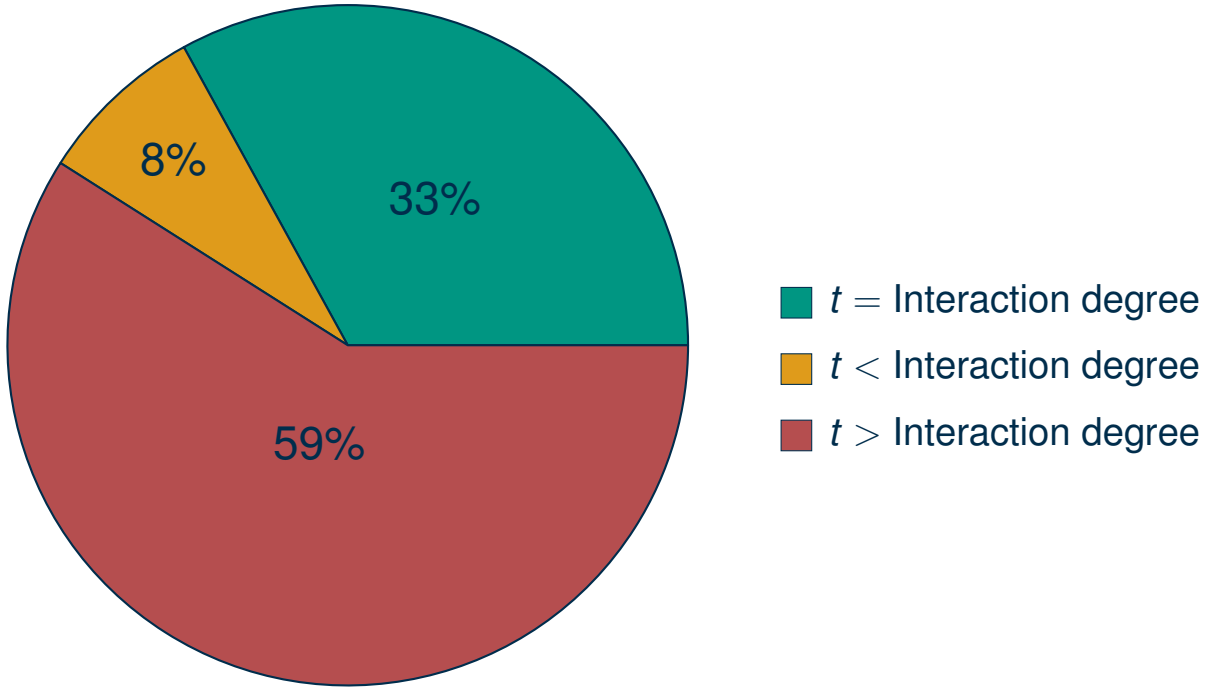
- Findings identified by t-wise but not $t-1$-wise sampling are **mostly of interaction strength** t
- By **collateral effect** [Pet+13], some findings of **higher interaction strength** can be identified
- Findings with **lesser interaction strength** are already **identified by weaker sampling**

First Results

RQ_2 - Reason for Increase in Findings

Expectations

- Findings identified by t-wise but not $t-1$-wise sampling are **mostly of interaction strength** t
- By **collateral effect** [Pet+13], some findings of **higher interaction strength** can be identified
- Findings with **lesser interaction strength** are already **identified by weaker sampling**



Ongoing Work

Current Issues

1. Some vulnerability warnings are raised only at a **sampling strength greater than** their actual **interaction strength**

Objectives

1. Analyze why certain findings are raised only by sampling stronger than their interation strength
   - Heuristics used in the analysis tool JOERN?
   - Discrepancy between problem space and solution space?

Identified by	JOERN query	File name	Line	Interaction strength	Presence condition
...	...	...	...	...	...
3-wise but not 2-wise	file-operation-race	miscutils/man.c	146	1	And(MAN)
3-wise but not 2-wise	file-operation-race	archival/bbunzip.c	75	1	Or(ZCAT ··· XZ)
...	...	...	...	...	...

Ongoing Work

Current Issues

1. Some vulnerability warnings are raised only at a **sampling strength greater than** their actual **interaction strength**
2. Calculation of actual **interaction strength does not yet consider feature model constraints**

Objectives

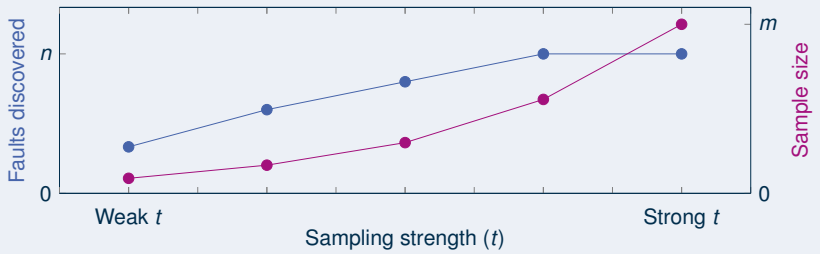
1. Analyze why certain findings are raised only by sampling stronger than their interation strength
   - Heuristics used in the analysis tool JOERN?
   - Discrepancy between problem space and solution space?
2. Devise a scalable method for calculating interaction strengths taking into account presence conditions and the feature model constraints

Identified by	JOERN query	File name	Line	Interaction strength	Presence condition
...	...	...	...	...	...
3-wise but not 2-wise	file-operation-race	miscutils/man.c	146	1	And(MAN)
3-wise but not 2-wise	file-operation-race	archival/bbunzip.c	75	1	Or(ZCAT ··· XZ)
...	...	...	...	...	...

Summary

Motivation

- Previous observations on variability bugs:

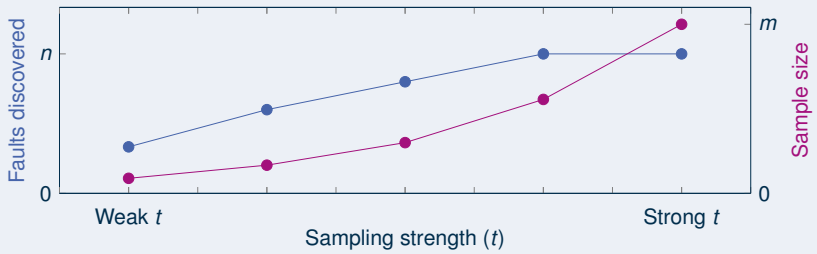


- Do these insights extend to vulnerabilities?

Summary

Motivation

- Previous observations on variability bugs:



- Do these insights extend to vulnerabilities?

Vulnerabilities in Highly-Configurable Systems

```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7          sink(y);
8  #endif
9          // ...
10     }
11 }
```

Variability-Induced Vulnerability

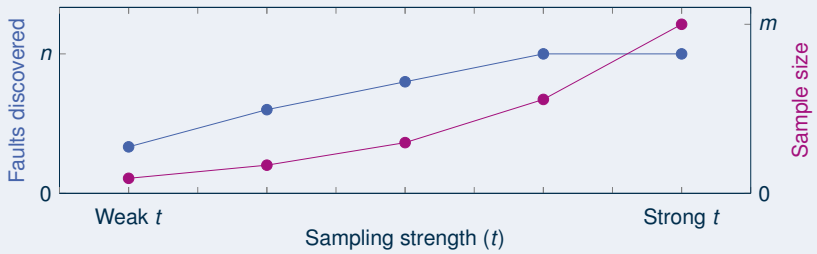
```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7      #ifdef CONFIG_SEND_DATA
8          sink(y);
9      #endif
10 #endif
11         // ...
12     }
13 }
```

Feature-Interaction Vulnerability

Summary

Motivation

- Previous observations on variability bugs:



- Do these insights extend to vulnerabilities?

Vulnerabilities in Highly-Configurable Systems

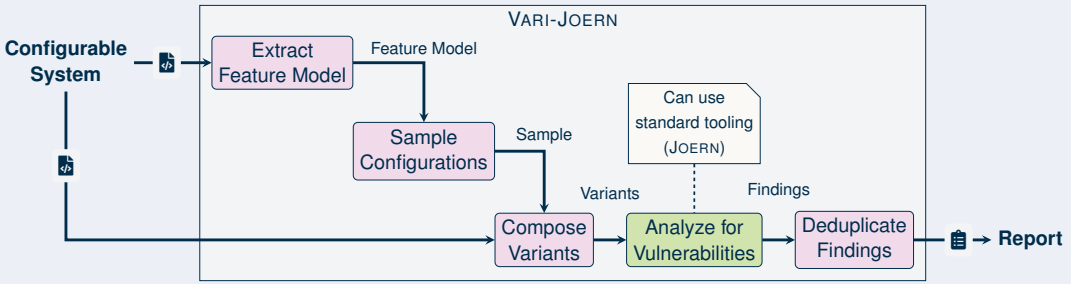
```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7          sink(y);
8  #endif
9          // ...
10     }
11 }
```

Variability-Induced Vulnerability

```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7      #ifdef CONFIG_SEND_DATA
8          sink(y);
9      #endif
10 #endif
11         // ...
12     }
13 }
```

Feature-Interaction Vulnerability

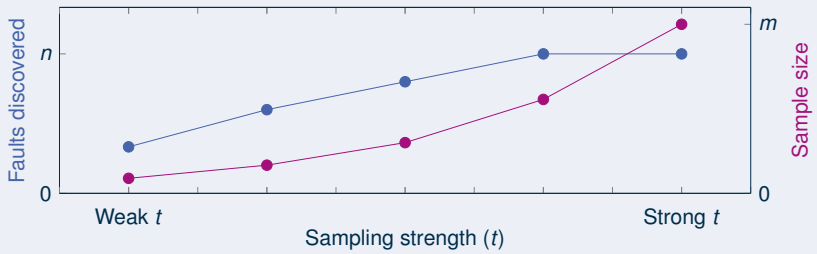
Sample-Based Vulnerability Discovery



Summary

Motivation

- Previous observations on variability bugs:



- Do these insights extend to vulnerabilities?

Vulnerabilities in Highly-Configurable Systems

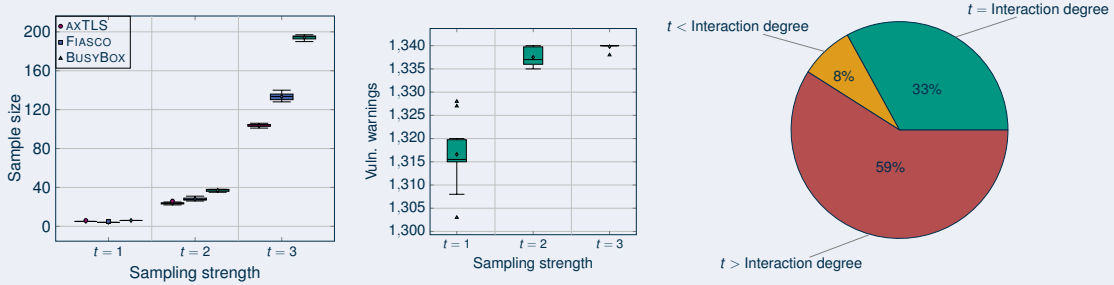
```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7          sink(y);
8  #endif
9          // ...
10     }
11 }
```

Variability-Induced Vulnerability

```
1  void foo() {
2      int x = source();
3      if(x < MAX){
4          int y = 0;
5  #ifdef CONFIG_PROCESS_INPUT
6          y = 2 * x;
7      #ifdef CONFIG_SEND_DATA
8          sink(y);
9      #endif
10 #endif
11         // ...
12     }
13 }
```

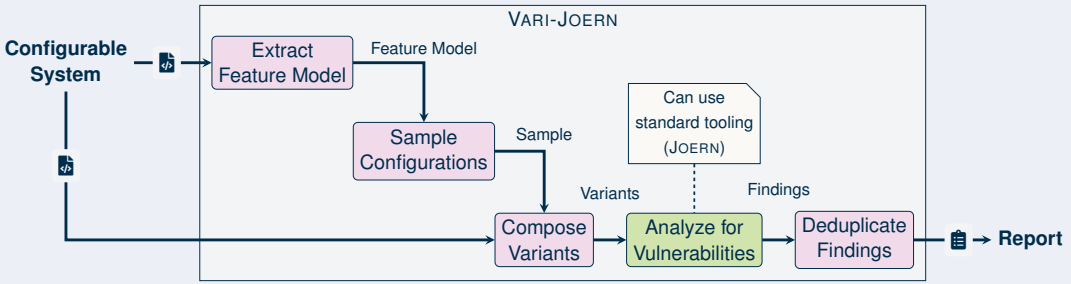
Feature-Interaction Vulnerability

Results and Ongoing Work



- ☑ **RQ_1**: Can confirm previous observations
- ☐ **RQ_2**: Further analysis and work required

Sample-Based Vulnerability Discovery



References I

[24a] *axTLS Embedded SSL*. Website. May 2024. URL: https://axtls.sourceforge.net/ (visited on 05/09/2024).

[24b] *BusyBox*. Website. May 2024. URL: https://busybox.net/ (visited on 05/09/2024).

[24c] *Joern - The Bug Hunter's Workbench*. Website. June 2024. URL: https://joern.io/ (visited on 06/02/2024).

[25] *The L4Re Microkernel*. Website. Mar. 2025. URL: https://os.inf.tu-dresden.de/fiasco/ (visited on 03/03/2025).

[Aba+17] Iago Abal et al. "Variability Bugs in Highly Configurable Systems: A Qualitative Analysis". In: *ACM Transactions on Software Engineering and Methodology* 26.3 (July 2017), pp. 1–34. ISSN: 1049-331X, 1557-7392. DOI: 10.1145/3149119. URL: https://dl.acm.org/doi/10.1145/3149119 (visited on 05/15/2024).

References II

[ABW14] Iago Abal, Claus Brabrand, and Andrzej Wasowski. “42 Variability Bugs in the Linux Kernel: A Qualitative Analysis”. In: *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. Vasteras Sweden: ACM, Sept. 2014, pp. 421–432. ISBN: 978-1-4503-3013-8. DOI: 10.1145/2642937.2642990. URL: https://dl.acm.org/doi/10.1145/2642937.2642990 (visited on 05/20/2024).

[GC11] Brady J. Garvin and Myra B. Cohen. “Feature Interaction Faults Revisited: An Exploratory Study”. In: *2011 IEEE 22nd International Symposium on Software Reliability Engineering*. Hiroshima, Japan: IEEE, Nov. 2011, pp. 90–99. ISBN: 978-1-4577-2060-4 978-0-7695-4568-4. DOI: 10.1109/ISSRE.2011.25. URL: http://ieeexplore.ieee.org/document/6132957/ (visited on 02/20/2025).

[Hal+19] Axel Halin et al. “Test Them All, Is It Worth It? Assessing Configuration Sampling on the JHipster Web Development Stack”. In: *Empirical Software Engineering* 24.2 (Apr. 2019), pp. 674–717. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-018-9635-4. URL: http://link.springer.com/10.1007/s10664-018-9635-4 (visited on 02/20/2025).

References III

[Lie+13] Jörg Liebig et al. “Scalable Analysis of Variable Software”. In: *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. Saint Petersburg Russia: ACM, Aug. 2013, pp. 81–91. ISBN: 978-1-4503-2237-9. DOI: 10.1145/2491411.2491437. URL: https://dl.acm.org/doi/10.1145/2491411.2491437 (visited on 04/28/2024).

[Med+16] Flávio Medeiros et al. “A Comparison of 10 Sampling Algorithms for Configurable Systems”. In: *Proceedings of the 38th International Conference on Software Engineering*. Austin Texas: ACM, May 2016, pp. 643–654. ISBN: 978-1-4503-3900-1. DOI: 10.1145/2884781.2884793. URL: https://dl.acm.org/doi/10.1145/2884781.2884793 (visited on 11/01/2024).

[Mor+19] Austin Mordahl et al. “An Empirical Study of Real-World Variability Bugs Detected by Variability-Oblivious Tools”. In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Tallinn Estonia: ACM, Aug. 2019, pp. 50–61. ISBN: 978-1-4503-5572-8. DOI: 10.1145/3338906.3338967. URL: https://dl.acm.org/doi/10.1145/3338906.3338967 (visited on 04/15/2024).

References IV

[Pet+13] Justyna Petke et al. “Efficiency and Early Fault Detection with Lower and Higher Strength Combinatorial Interaction Testing”. In: *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. Saint Petersburg Russia: ACM, Aug. 2013, pp. 26–36. ISBN: 978-1-4503-2237-9. DOI: 10.1145/2491411.2491436. URL: https://dl.acm.org/doi/10.1145/2491411.2491436 (visited on 03/24/2025).

[Thü+14] Thomas Thüm et al. “A Classification and Survey of Analysis Strategies for Software Product Lines”. In: *ACM Computing Surveys* 47.1 (July 2014), pp. 1–45. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2580950. URL: https://dl.acm.org/doi/10.1145/2580950 (visited on 04/16/2024).

[Yam+14] Fabian Yamaguchi et al. “Modeling and Discovering Vulnerabilities with Code Property Graphs”. In: *2014 IEEE Symposium on Security and Privacy*. San Jose, CA: IEEE, May 2014, pp. 590–604. ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.44. URL: http://ieeexplore.ieee.org/document/6956589/ (visited on 04/15/2024).