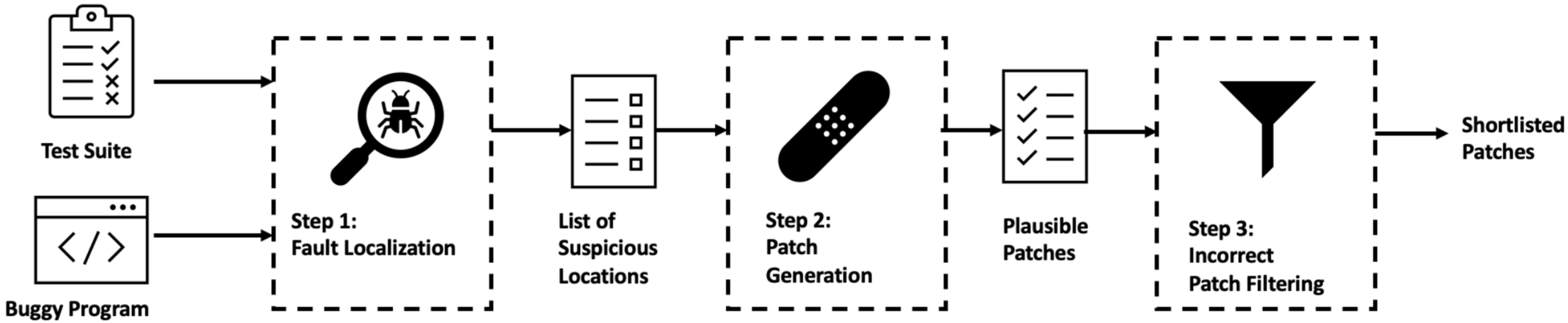
보안 취약성 패치 검증

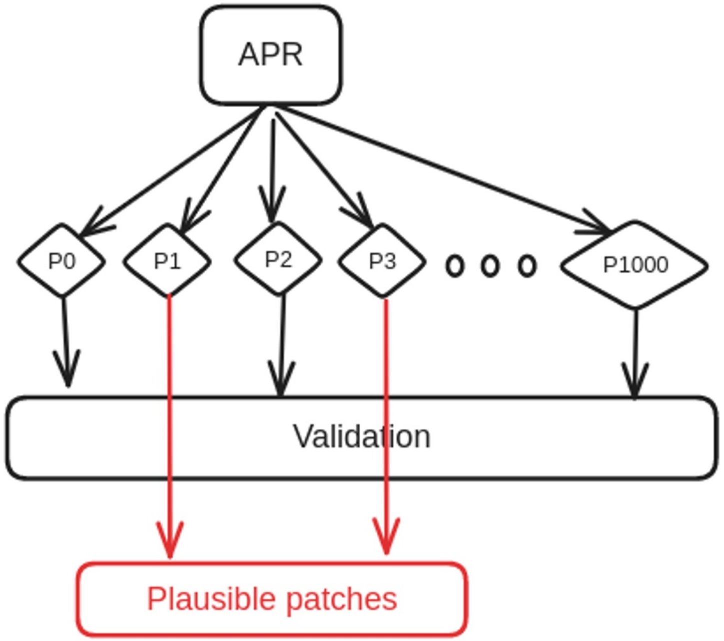
한승헌, 김영재, 이예승, 이주용(UNIST)

Introduction



- Security Vulnerabilities
  - Can be disclosed by crash-inducing inputs
  - Such inputs can be automatically discovered by fuzzing
  - Patches can be automatically generated by automated program repair
  - But can we trust those patches?

Introduction



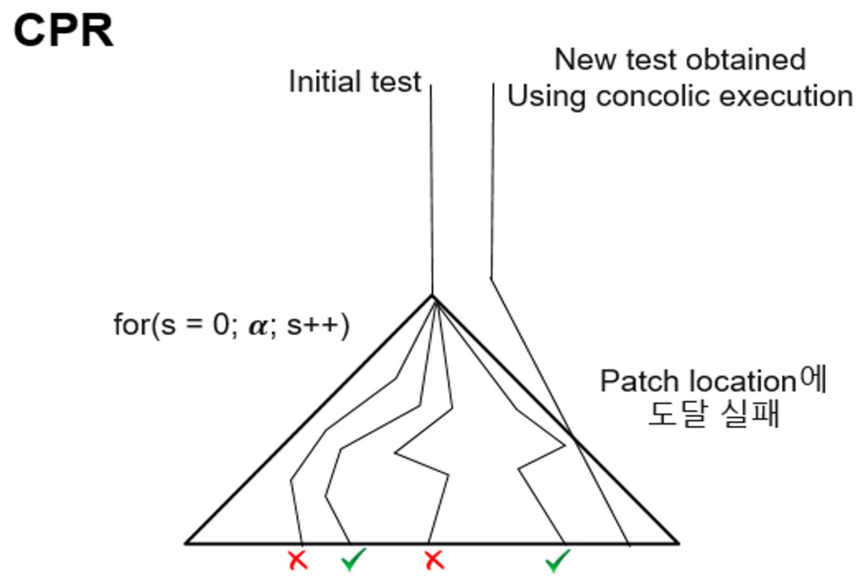

```c
int divide(int x, int y) {
    int res;
    y = x - 1;
    if (1) return -1; // Patch?
    res = 1000 / y;
    return res;
}
```

- Patch validation
  - If there is no crash after patch, can it be considered as correct patch?
  - How can we check patch correctness?

Methods

Symbolic Execution

- Theoretically, symbolic execution can verify all inputs and execution paths
- Scalability issue
    - Path explosion

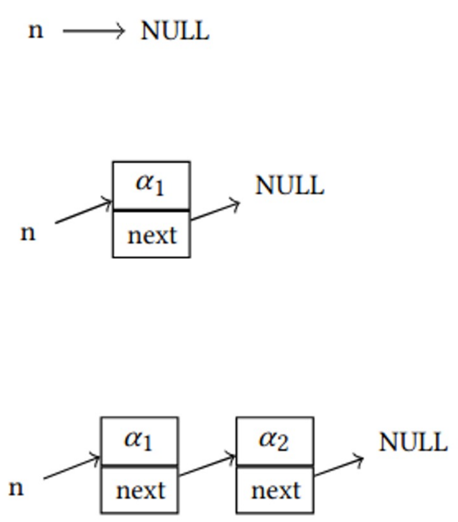


Methods

Under-Constrained Symbolic Execution

- Improves scalability by directly running target function
- Starts from empty memory state
- Use lazy initialization to generate inputs

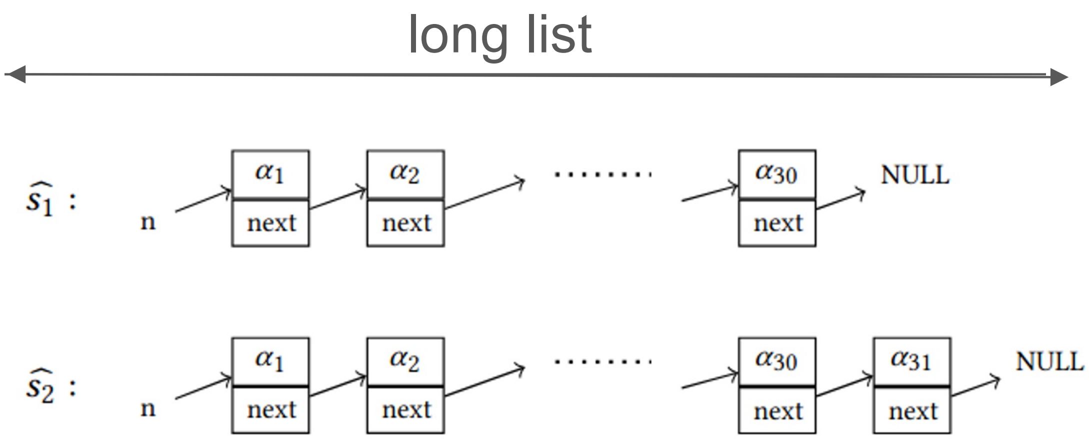
```
struct node {
    int val;
    struct node *next;
};

int listSum(struct node *n) {
    int sum = 0;
    /* int i = 1; */
    while (n) {
        sum += n->val;
        n = n->next;
        /* i *= 2; */
    }
    /* g = arr[i]; */
    return sum;
}
```



Methods

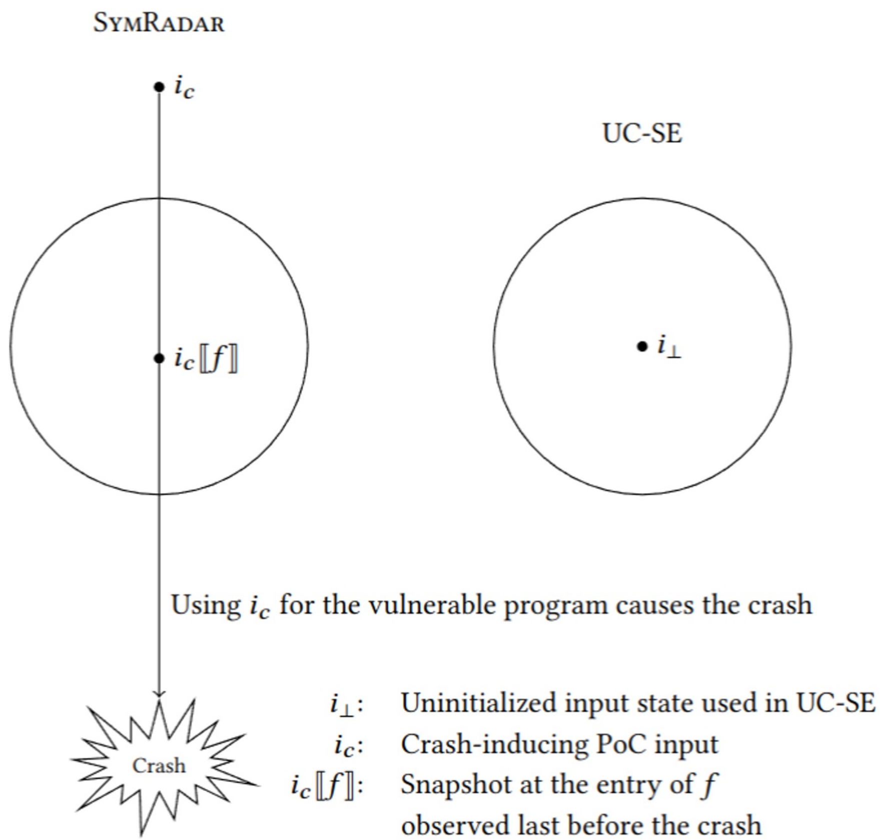
Under-Constrained Symbolic Execution

- Limitation
  - Large & Complex memory structure

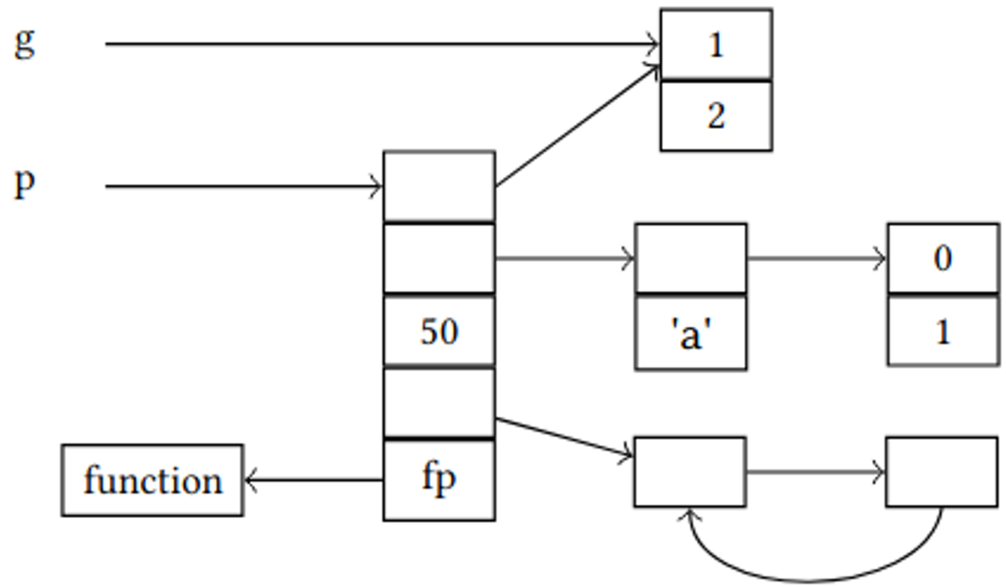


Our Approach

- How to handle such programs?
  - PoC-centered bounded verification
  - Verify near the buggy memory state

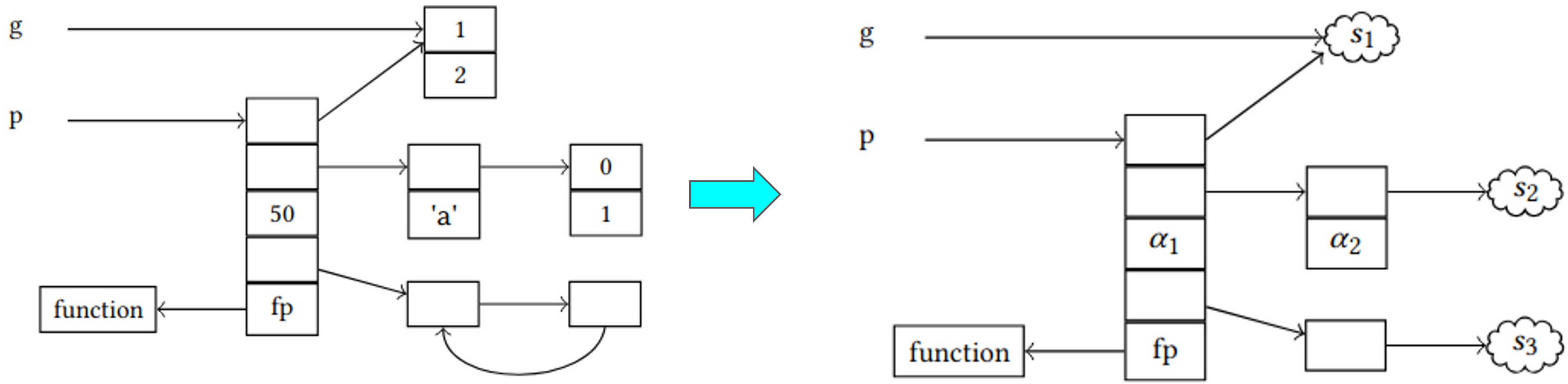


Concrete Snapshot



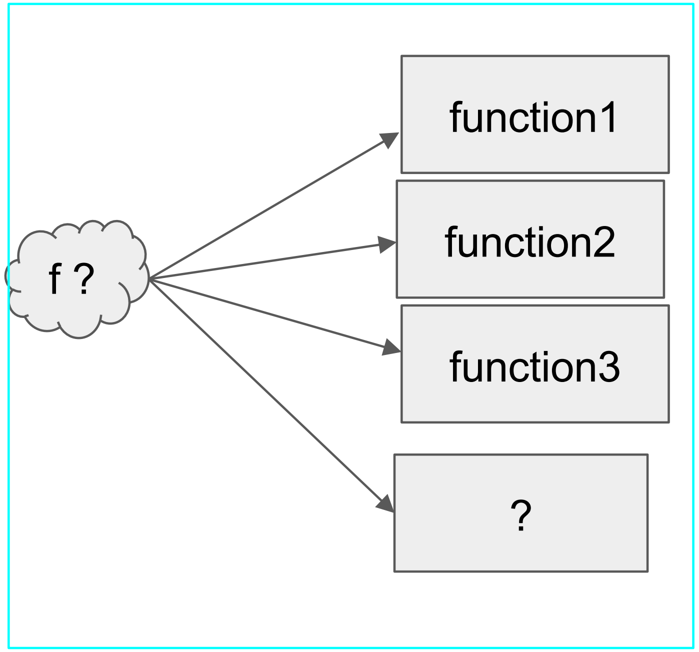
- Run program with crash-inducing input
- Take a snapshot
- Obtain memory graph, memory access log, …

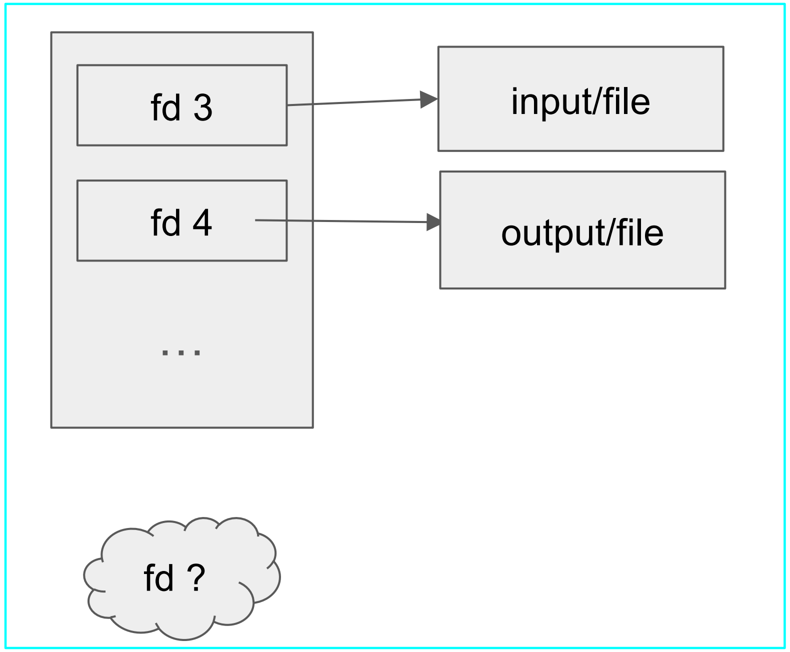
Abstract Snapshot



- Select some pointer to the terminal node, and all primitive values
- Symbolize selected part of the concrete snapshot
- Lazy initialization

Optimization

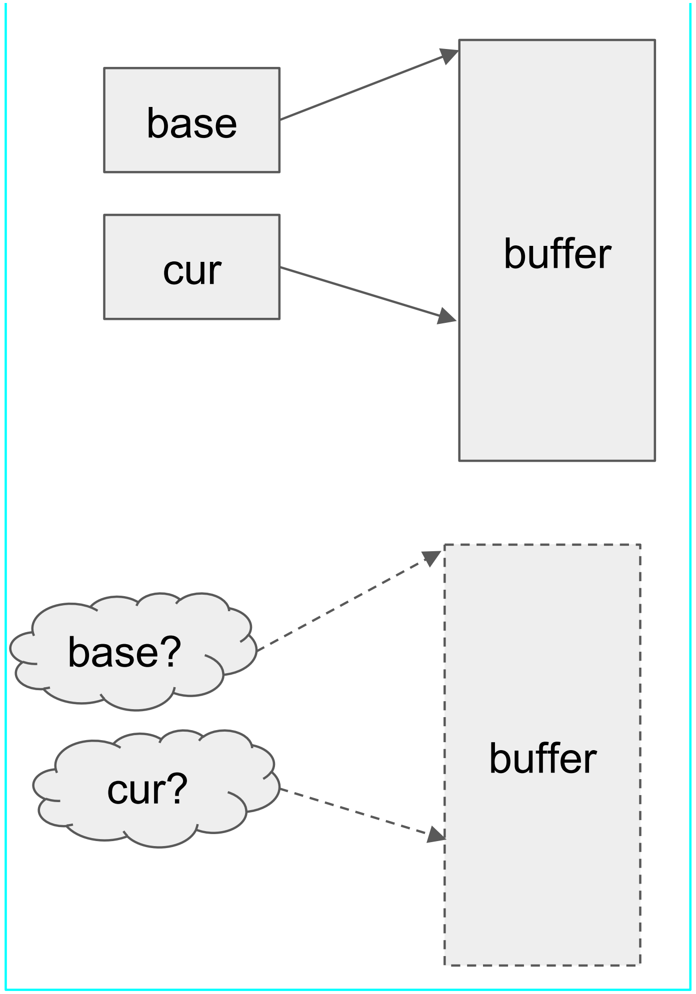




- Inevitable concretization
    - Function pointer, external function call
    - Concretize based on concrete snapshot

Optimization

- Order-preserving lazy initialization
  - Preserve ordering of the pointers those point same object



Patch Classification

Case	r_i	r'_i	Condition	Classification
1	Crash	Normal	-	Safe
2	Crash C	Crash	$C = C_{PoC}$	Unsafe
3	Crash C	Crash	$C \neq C_{PoC}$	Safe
4	Normal	Crash	$C = C_{PoC}$	Unsafe
5	Normal	Crash	$C \neq C_{PoC}$	Safe
6	Normal	Normal	-	Check regression

- No crash, no regression error
- Problem: infeasible input
- Heuristic:
    - Check original crash location with PoC input
    - If crash location is different, that is likely to due to the infeasible input

Evaluation: Patch Filtering

	CPR †	SPIDER	VULNFIX †	UC-KLEE	SYMRADAR
Recall	96%	77%	62%	88%	100%
Specificity	8%	59%	66%	57%	78%
Balanced Accuracy	52%	67%	64%	73%	89%

- CPR: APR tool
- SPIDER: static analysis
- VulnFix: system-level fuzzing + snapshot fuzzing
- UC-KLEE: under-constrained symbolic execution

Evaluation: Patch Filtering

Program	Bug ID	Patches		CPR †		SPIDER		VULNFIX †		UC-KLEE		SYMRADAR	
		C	I	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
coreutils	gnubug-25003	1	138	0	119	0	69	0	131	0	122	0	18
coreutils	gnubug-25023	1	43	0	43	0	43	0	27	0	3	0	0
coreutils	bugzilla-19784	1	3	0	3	0	0	0	1	0	3	0	3
coreutils	bugzilla-26545	1	966	0	890	1	0	n.a.	n.a.	0	4	0	4
jasper	CVE-2016-8691	1	90	0	66	0	90	1	0	0	45	0	31
jasper	CVE-2016-9387	0	45	0	11	0	45	n.a.	n.a.	0	40	0	0
binutils	CVE-2017-15025	1	89	0	89	0	89	0	2	0	89	0	89
binutils	CVE-2018-10372	1	1	0	1	0	1	0	1	0	1	0	1
libjpeg	CVE-2012-2806	1	3	0	3	0	3	0	3	0	3	0	3
libjpeg	CVE-2017-15232	1	510	0	510	1	0	0	248	0	248	0	75
libjpeg	CVE-2018-14498	1	89	0	87	1	0	1	0	0	89	0	89
libjpeg	CVE-2018-19664	1	89	0	89	0	89	1	0	0	89	0	2
libtiff	CVE-2014-8128	1	1	0	1	0	1	n.a.	n.a.	0	1	0	1
libtiff	CVE-2016-10094	1	89	0	52	0	55	1	3	0	89	0	87
libtiff	CVE-2016-3186	1	89	0	89	0	89	n.a.	n.a.	0	3	0	14
libtiff	CVE-2016-3623	1	90	0	69	1	0	0	90	0	45	0	21
libtiff	CVE-2016-5314	1	138	0	136	0	138	n.a.	n.a.	0	138	0	4
libtiff	CVE-2016-5321	1	1	0	1	0	1	0	1	0	1	0	1
libtiff	CVE-2016-9273	1	9	0	8	1	0	1	0	0	9	0	2
libtiff	CVE-2017-7595	1	90	0	90	0	90	1	0	0	90	0	90
libtiff	CVE-2017-7601	1	63	0	63	0	63	1	0	0	63	0	21
libtiff	bugzilla-2611	1	89	0	78	0	89	0	80	0	89	0	29
libtiff	bugzilla-2633	1	89	0	89	0	89	0	1	0	15	0	45
libxml2	CVE-2012-5134	1	1	1	0	0	1	0	1	0	1	0	1
libxml2	CVE-2016-1838	1	138	0	138	0	138	1	0	1	12	0	13
libxml2	CVE-2016-1839	1	45	0	45	0	45	0	1	1	4	0	1
libxml2	CVE-2017-5969	1	13	0	13	1	0	0	13	0	13	0	13
Total	-	26	3011	1	2783	6	1228	8	603	3	1306	0	657

Evaluation: Ablation Study

Tool	Bound	Heuristic	FN	FP	Recall	Specificity	B.Acc
SymRadar	3	Y	0	657	100%	78%	89%
		N	8	617	69%	80%	74%
	6	Y	2	687	92%	77%	85%
		N	7	661	73%	78%	76%
	9	Y	2	693	92%	77%	85%
		N	8	664	69%	78%	74%
UC-KLEE	3	Y	3	1306	88%	57%	73%
		N	7	1266	73%	58%	66%
	6	Y	3	1323	88%	56%	72%
		N	7	1060	73%	65%	69%
	9	Y	3	1322	88%	56%	72%
		N	7	1061	73%	65%	69%

- Lazy initialization bound (k= 3, 6, 9)
- Impact of heuristic (infeasible input removal based on crash location)