

TIVER: Identifying Adaptive Versions of C/C++ Third-Party Open-Source components Using a Code Clustering Technique

Youngjae Choi, Seunghoon Woo

KOREA UNIVERSITY



고려대학교
KOREA UNIVERSITY



SSP LAB
Software Security and
Privacy Laboratory



Motivation

Open-source software (OSS) reuse is widely adopted

- > Can expose system owing to propagated vulnerabilities
- > Reused OSS components, consist of files from various versions

Why: Code-level reuse (C/C++) / Partial reuse / Backporting patches

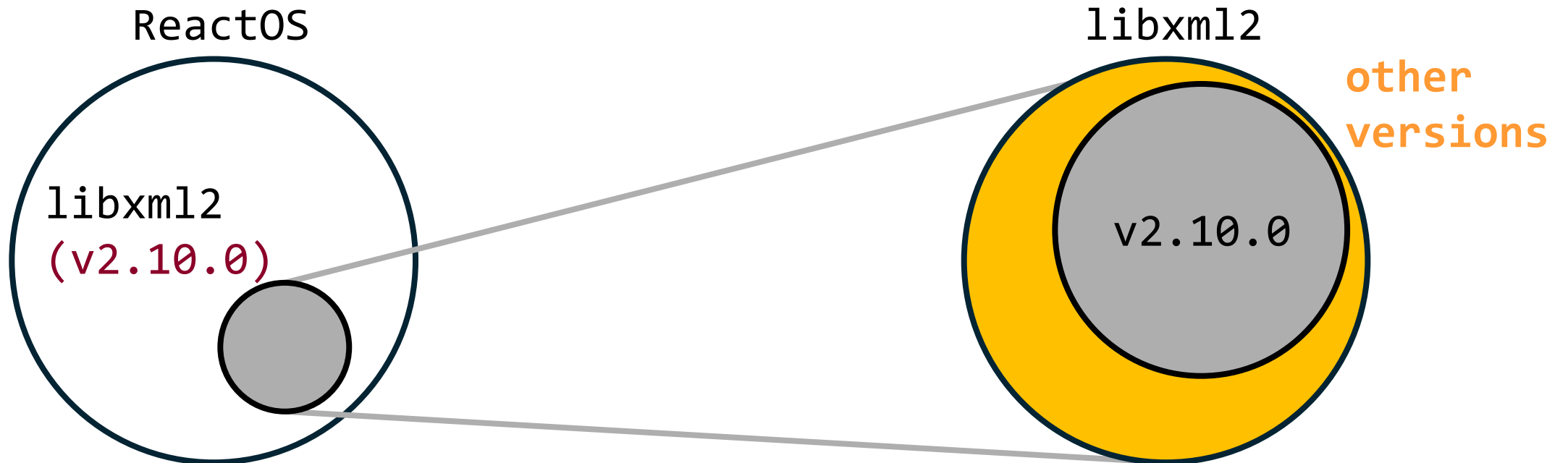
-> ***Current SBOM[†]: single version per OSS component***

-> **Is this single-version approach robust enough for modern supply chain security?**

Problem

- Assigning single specific version for reused OSS components

Single-version approach



Problem

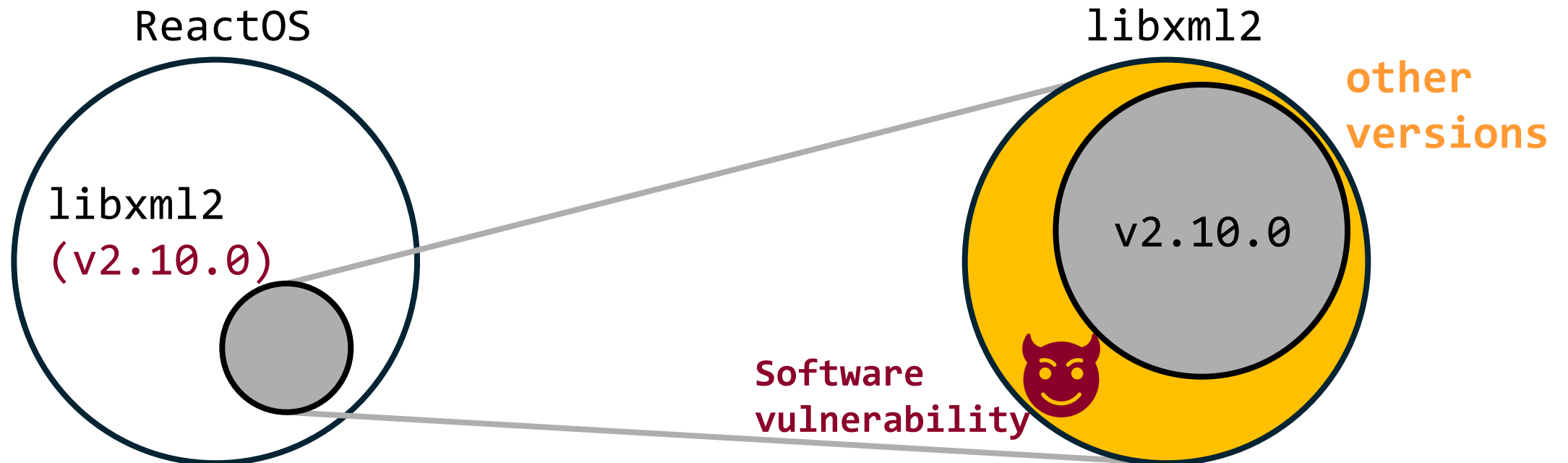
- Assigning si

TABLE I: Version distribution of reused Libxml2 source files in ReactOS (as of March 2024).

Version	#Reused files	Ratio
v2.9.10	4	6%
v2.9.12	7	10%
v2.10.0	48	71%
v2.10.1	1	1%
v2.10.2	2	3%
v2.10.3	6	9%
Total	68	100%

components

Single-version approach



Problem

- Assigning si

TABLE I: Version distribution of reused Libxml2 source files in ReactOS (as of March 2024).

Version	#Reused files	Ratio
v2.9.10	4	6%
v2.9.12	7	10%
v2.10.0	48	71%
v2.10.1	1	1%
v2.10.2	2	3%
v2.10.3	6	9%
Total	68	100%

components

Single-version approach

ReactOS

libxml2

other

Current SBOM cannot be used to identify this vulnerability!!!

vulnerability

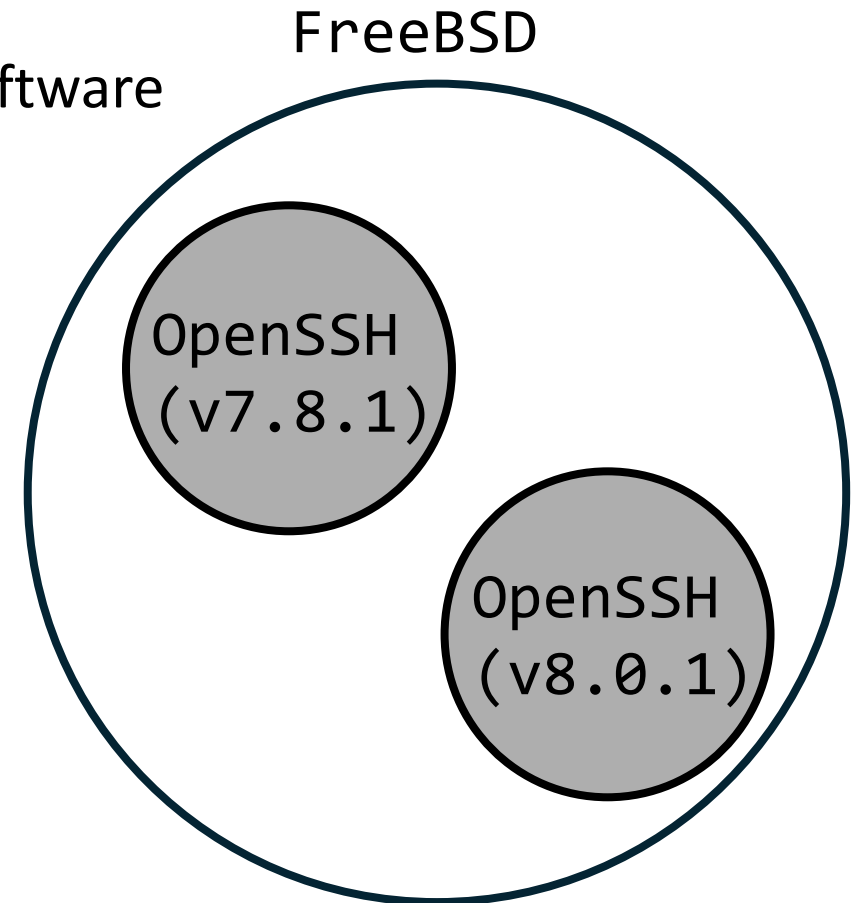
Goal

- Identifying “*adaptive version*” of reused OSS components in target software
- *Adaptive version*: A comprehensive representation that encompasses the various versions present in reused code

Challenges

1. Duplicate components

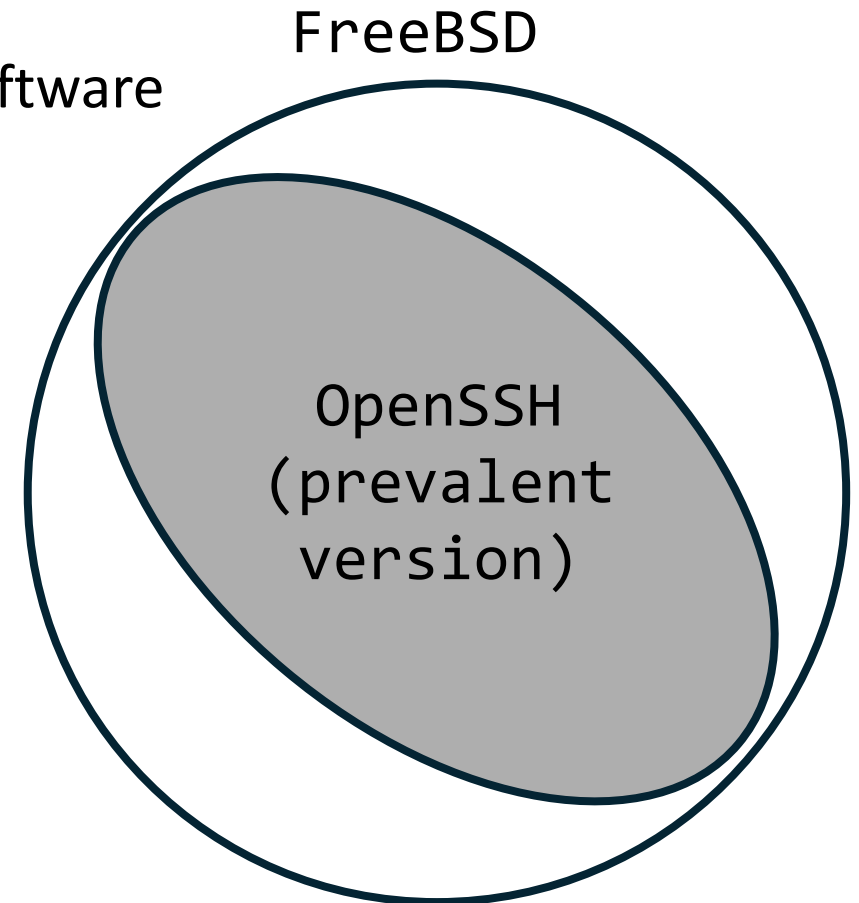
- Same OSS is reused in multiple parts of target software



Challenges

1. Duplicate components

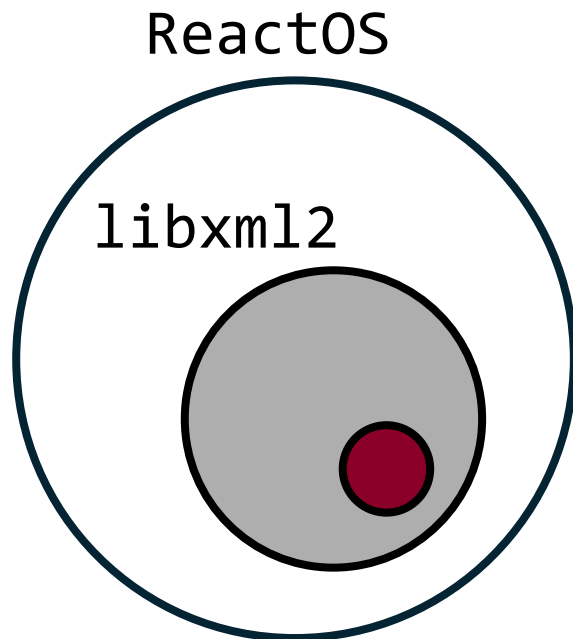
- Same OSS is reused in multiple parts of target software
- How single-version approach handles



Challenges

2. Noise

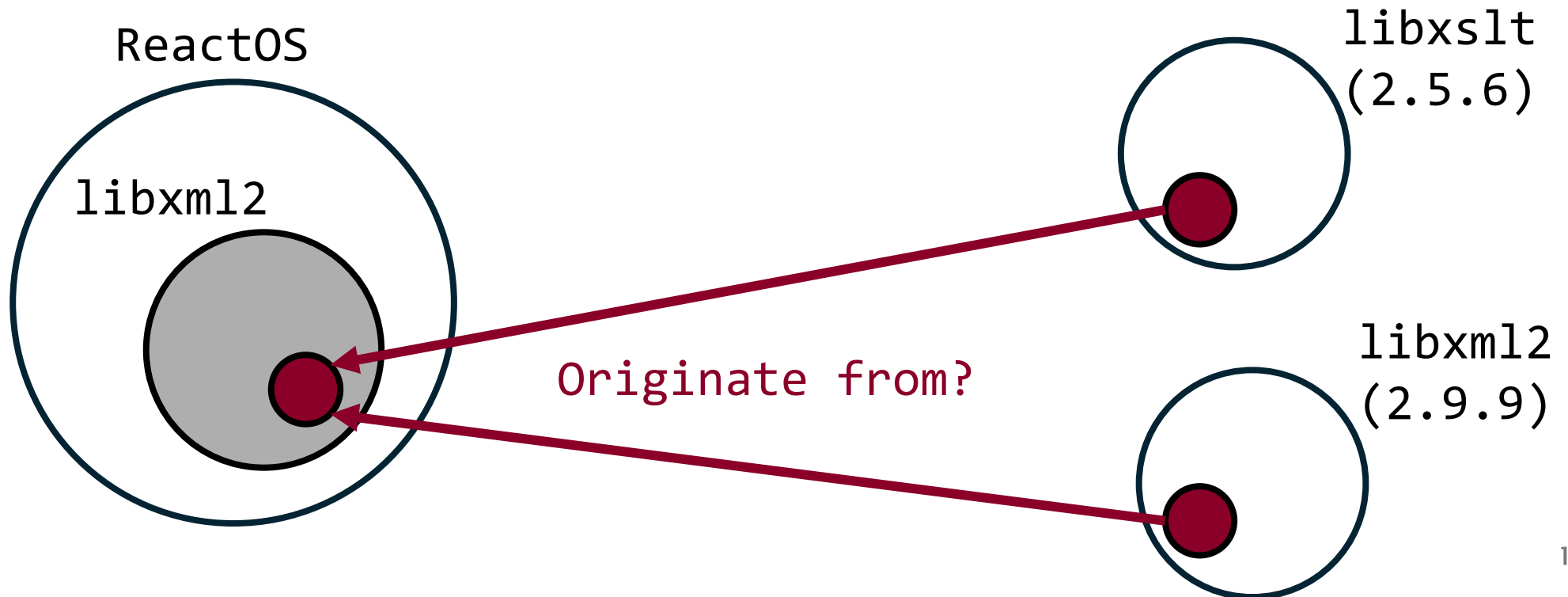
- Code snippets commonly found across diverse OSS
- Interferes accurate version identification by being misclassified as OSS



Challenges

2. Noise

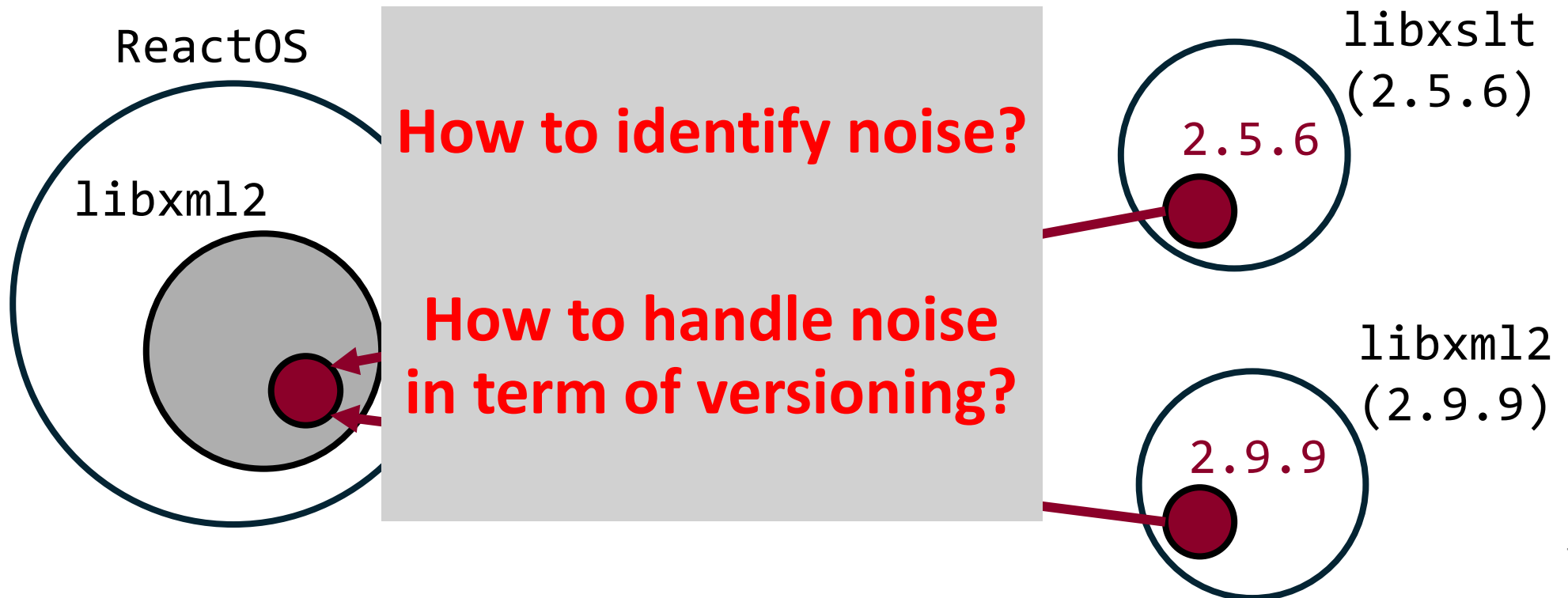
- Code snippets commonly found across diverse OSS
- Interferes accurate version identification by being misclassified as OSS



Challenges

2. Noise

- Code snippets commonly found across diverse OSS
- Interferes accurate version identification by being misclassified as OSS

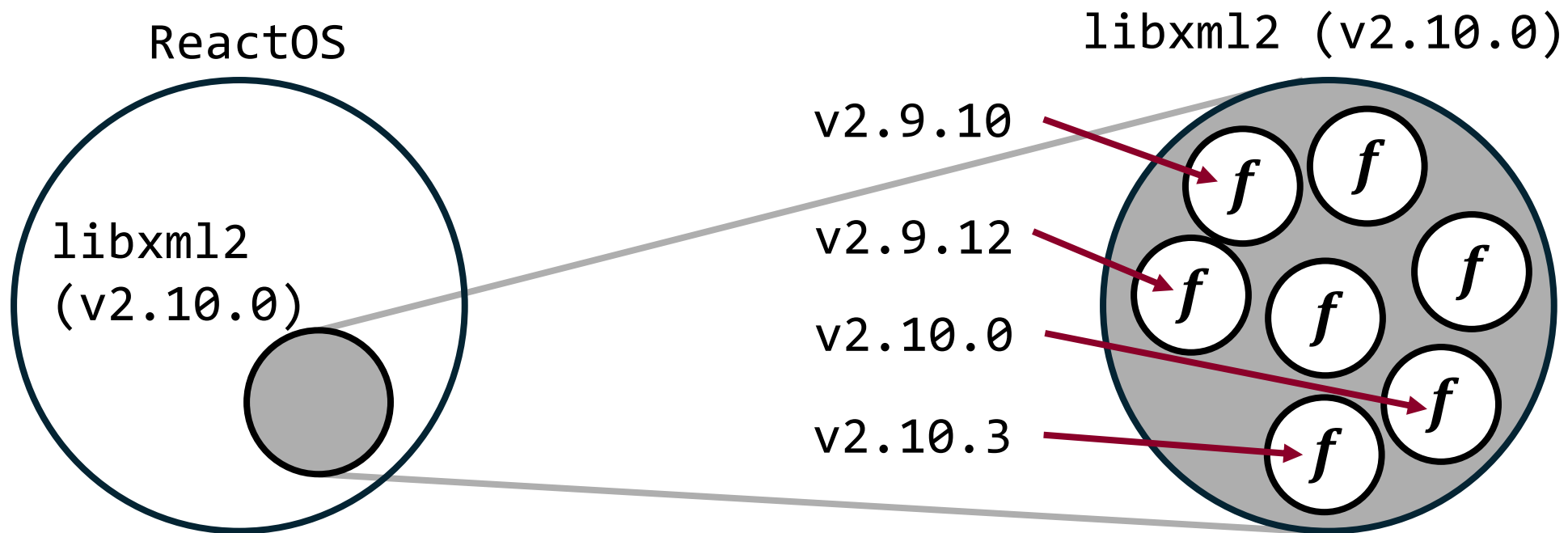


TIVER

- adapTive Version analyzER
 - Novel approach to identify *adaptive version* of OSS components
- Key techniques to overcome challenges
 - Function-level versioning
 - Code clustering

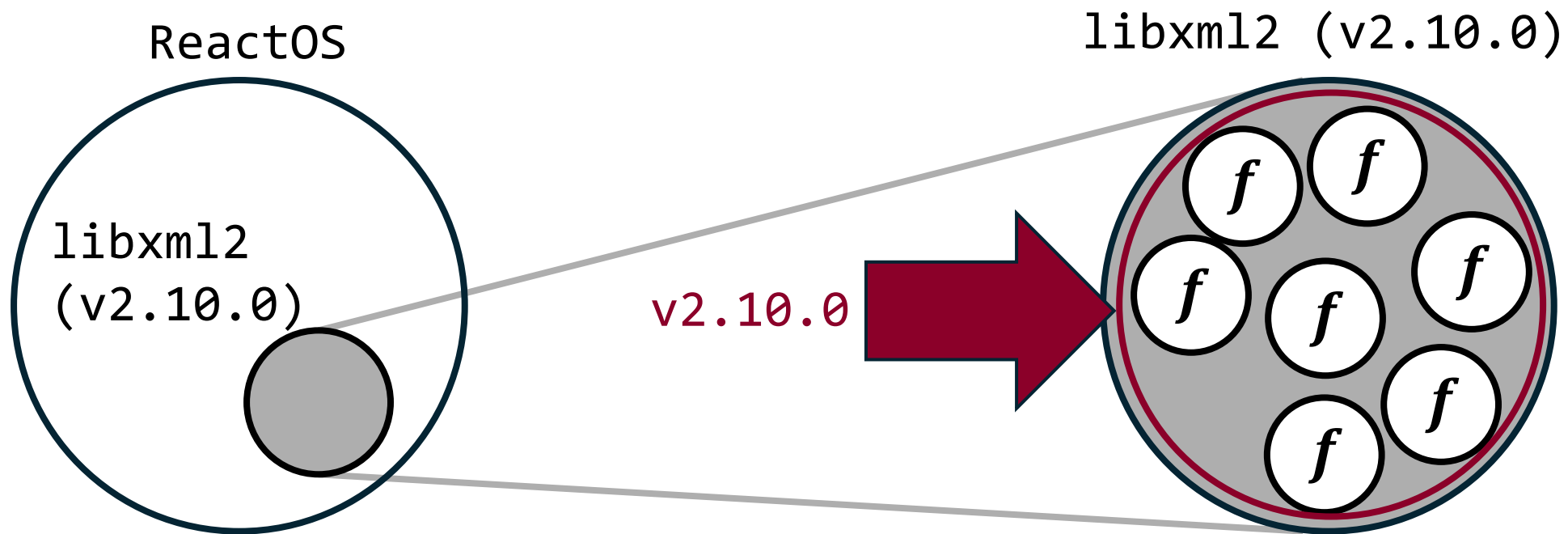
TIVER: Function-level versioning

- Existing single-version approaches



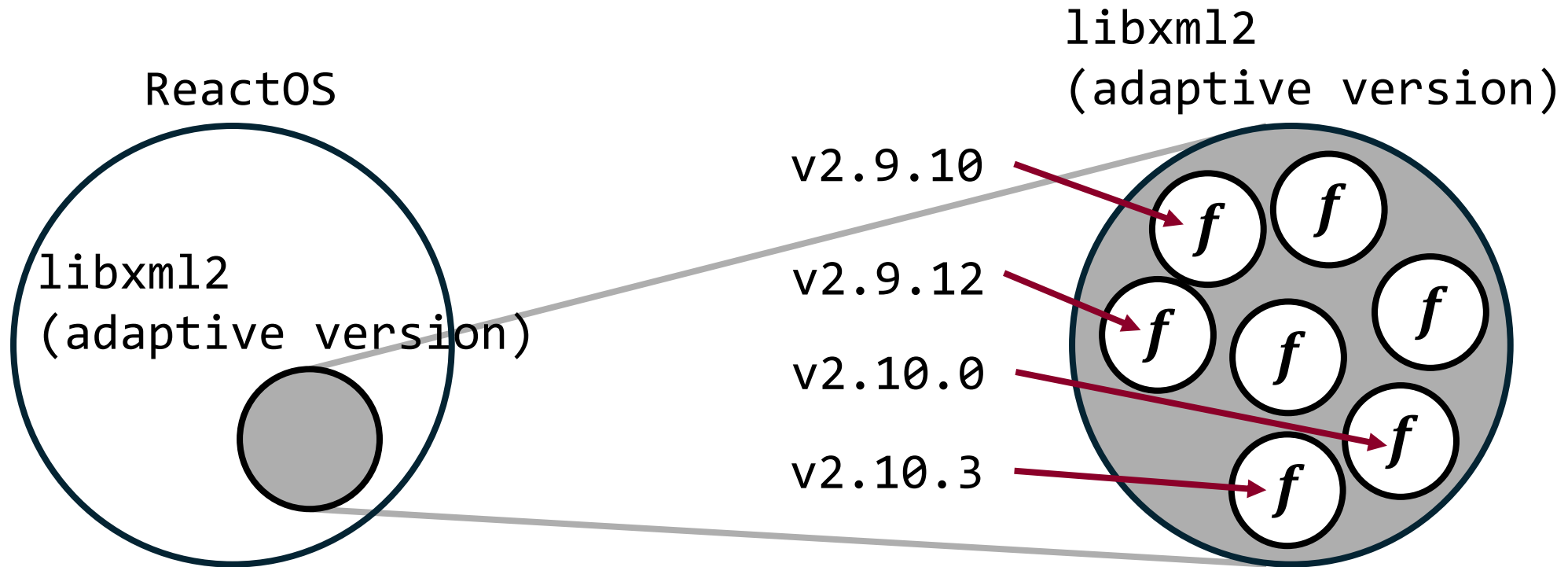
TIVER: Function-level versioning

- Existing single-version approaches



TIVER: Function-level versioning

- TIVER: Function-level versioning



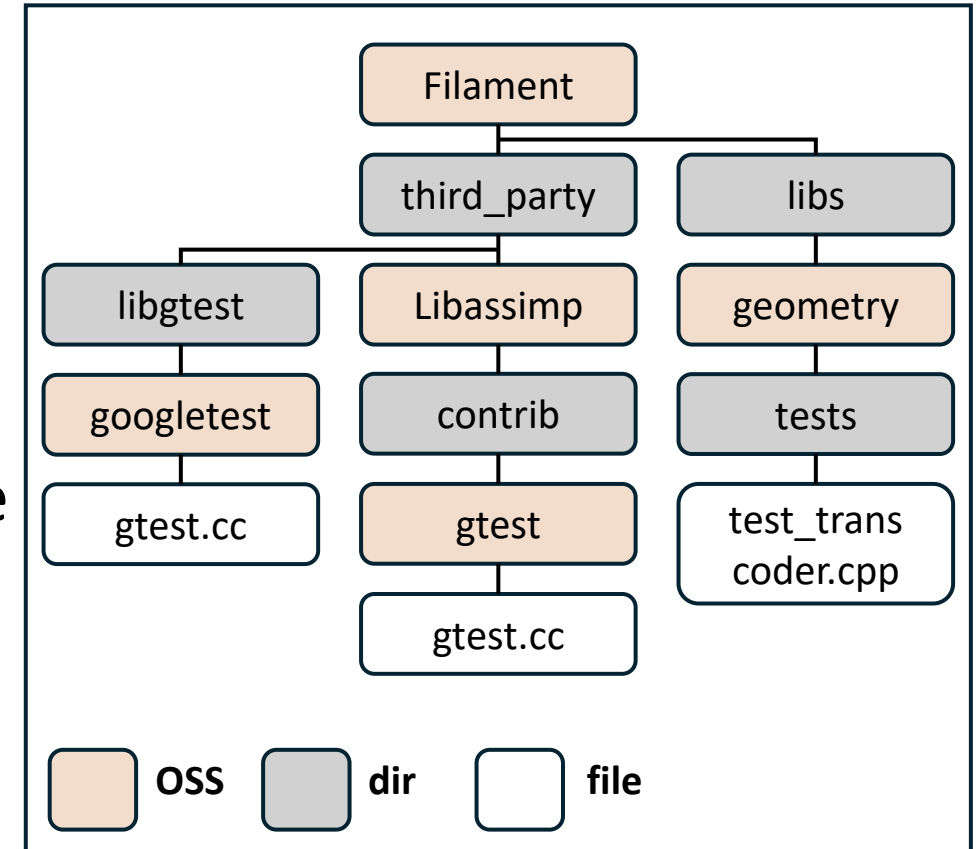
TIVER: Code Clustering

Directory hierarchy of OSS (GoogleTest)

- TIVER uses filename as indicator
- *known duplicates*
 - Examine *known duplicates* before clustering process
 - Same filename coexist in target software
 - > Redundant OSS reuse

OR

-> *Already exist in original OSS*

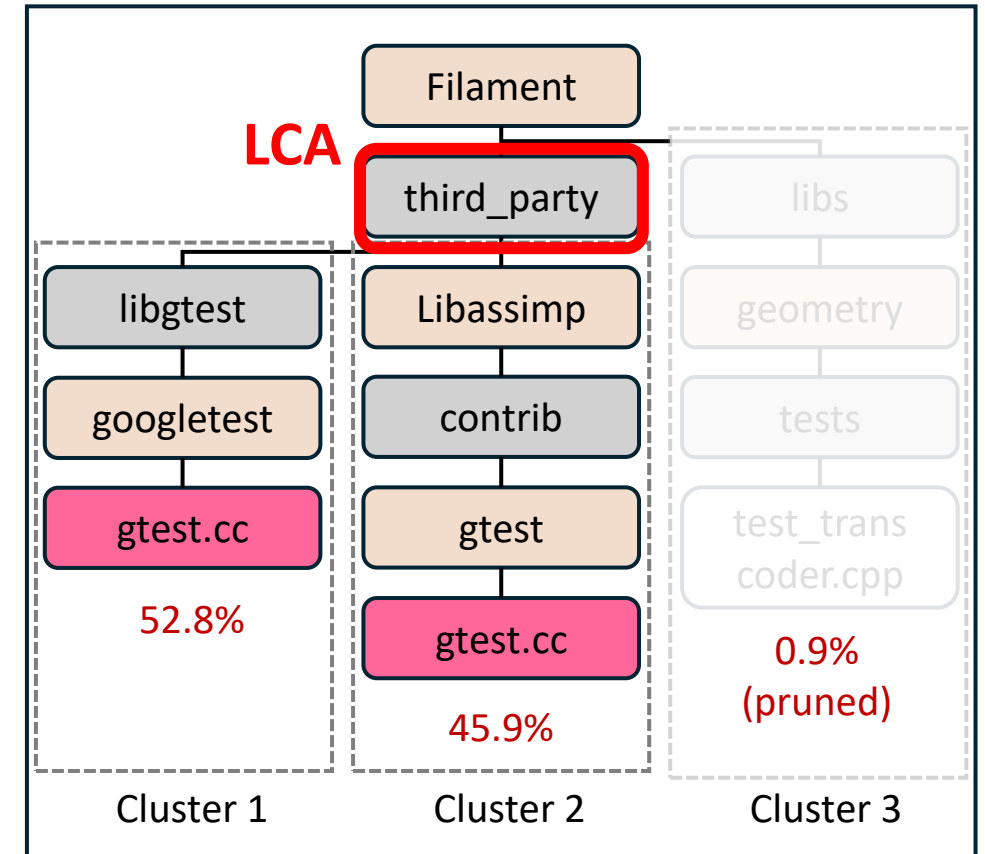


Known duplicates: NONE

TIVER: Code Clustering

Directory hierarchy
of OSS (GoogleTest)

- **Code Clustering**
 - Use LCA (Lowest Common Ancestor)
 - Distinguish duplicate components
- **Cluster pruning**
 - Eliminate noise
 - $th = 3\%$



Known duplicates: NONE

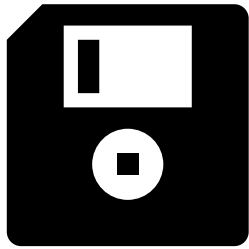
TIVER: Adaptive version

```
{1.2.0} -> 1.2.0
{1.2.0, invalid_ver†} -> +1.2.0
{3.2.0, 2.2.5, 1.2.0} -> *1.2.0
{1.2.0, 1.2.5, 1.3.2} -> ^1.2.0
{1.2.0, 1.2.5, 1.2.7} -> ~1.2.0
```

Per cluster

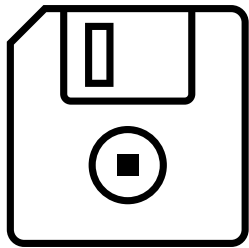
Evaluation

- Dataset




Functions

Functions present in all versions of 10,417 OSS projects
- 4,720,744 version strings



Repositories

Popular 2,025 repositories in  GitHub (C/C++)
- Ranked by the number of stars
- 570 million lines of code

Evaluation

- **Accuracy**
- **# Duplicate component distinction**
 - 88% Precision & 92% Recall
 - 230/273 components were TP
- **# Noise elimination**
 - 86% Precision & 87% Recall
 - 264/307 clusters were TP

Evaluation

- **Accuracy**

- **# Duplicate component distinction**

- 230/273 components were TP
- 88% Precision & 92% Recall

- **# Noise elimination**

- 264/307 clusters were TP
- 86% Precision & 87% Recall

VS. CNEPS (ICSE 2024)

	TIVER	CNEPS
TP	46	20
FN	6	28
Recall	0.88	0.42

Evaluation

- **Effectiveness**

- VS. V1SCAN (Single version based vulnerability detector)
- USENIX SECURITY 2023
- On average,

V1SCAN covers 1 version per component

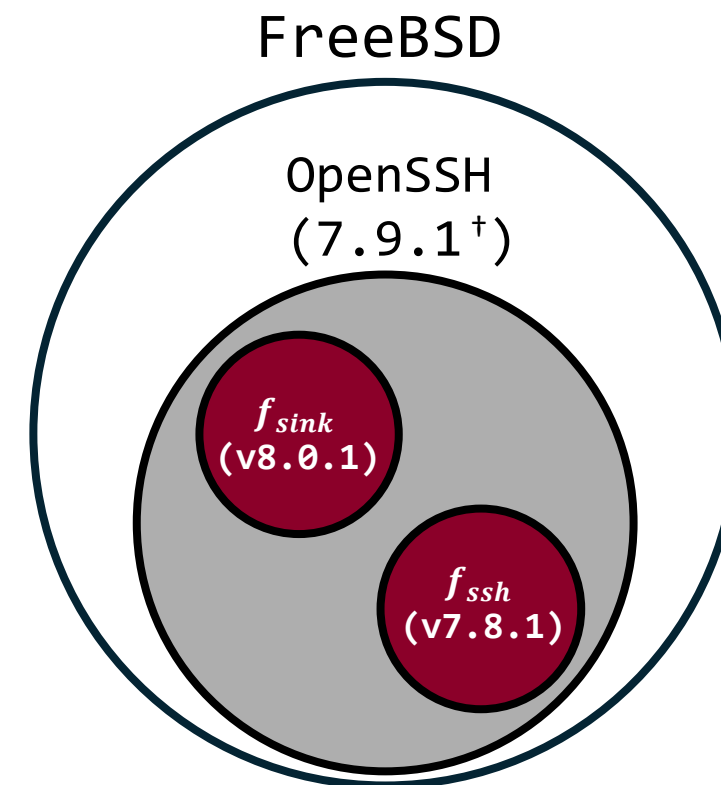
TIVER covers **3.49** versions per component

V1SCAN cleanses 0 noisy region per component

TIVER cleanses **3.31** noisy clusters per component

Implication

- **Value of TIVER**
 - Enhances supply chain security through precise version tracking



CVE-id	vulnerable func	reused version	previous	TIVER
CVE-2018-20685	f_{sink} (~7.9.1)	8.0.1	Vulnerable (FP)	Safe
CVE-2018-15919	f_{ssh} (~7.8.1)	7.8.1	Safe (FN)	Vulnerable

Conclusion

- **TIVER**: novel approach for identifying adaptive versions of C/C++ OSS components
 - Function-level versioning
 - OSS code clustering
- TIVER can be used to
 - Perform effective OSS management
 - Covers 3.49 versions & Cleanses 3.31 noisy clusters per component
 - Enhance supply chain security
 - Eliminated 81% of FPs from functions flagged as vulnerable by single-version approach

Q & A

Thank you for your attention!

- TIVER repository (<https://github.com/Genius-Choi/TIVER-public>)
- Dataset (<https://zenodo.org/records/14862460>)

Contact

- Youngjae Choi (youngjae Choi@korea.ac.kr)
- Software Security & Privacy Laboratory
 - SSP LAB (<https://ssp.korea.ac.kr>)



고려대학교
KOREA UNIVERSITY



SSP LAB
Software Security and
Privacy Laboratory

Appendix

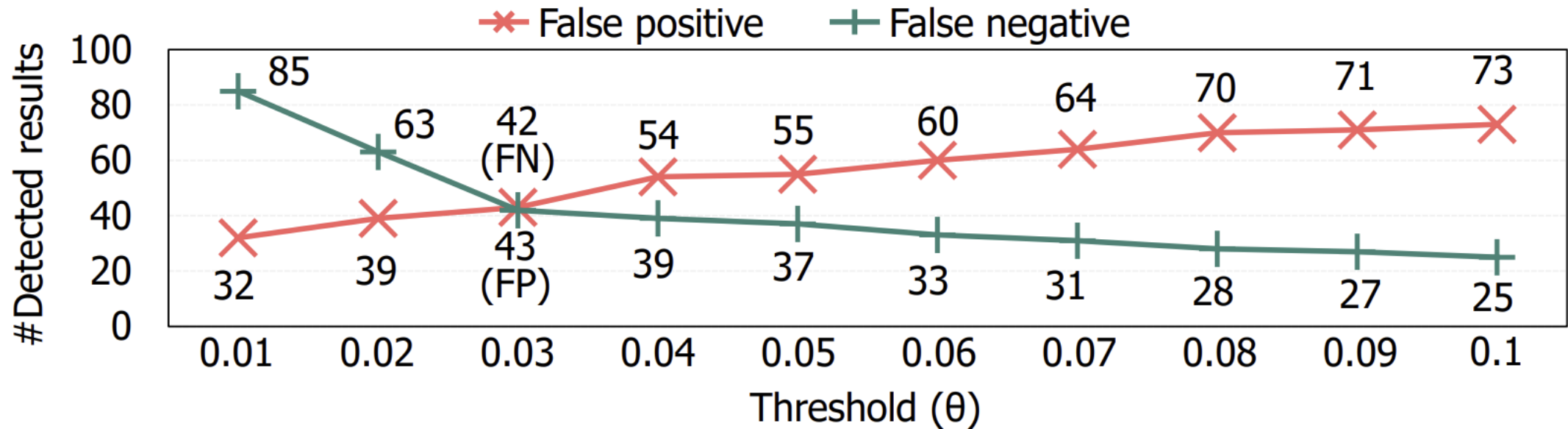


Fig. 3: Experimental results for measuring sensitivity of θ .

Appendix

Avg: 1.67s

