V1SCAN: Discovering 1-day Vulnerabilities in Reused C/C++ Open-source Software Components Using Code Classification Techniques

Seunghoon Woo, Eunjin Choi, Heejo Lee, Hakjoo Oh

Korea University

USENIX Security 2023

Korea University College of Informatics

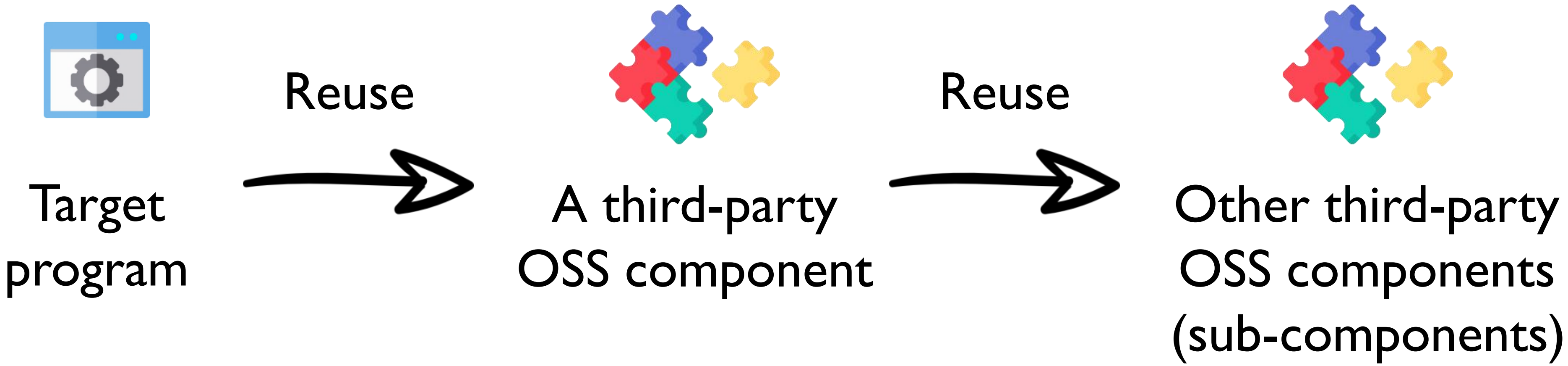
CSSA Center for Software Security and Assurance

CCSLAB Computer & Communication Security Laboratory

Motivation

Open-source software (OSS), a driving force behind innovative software development

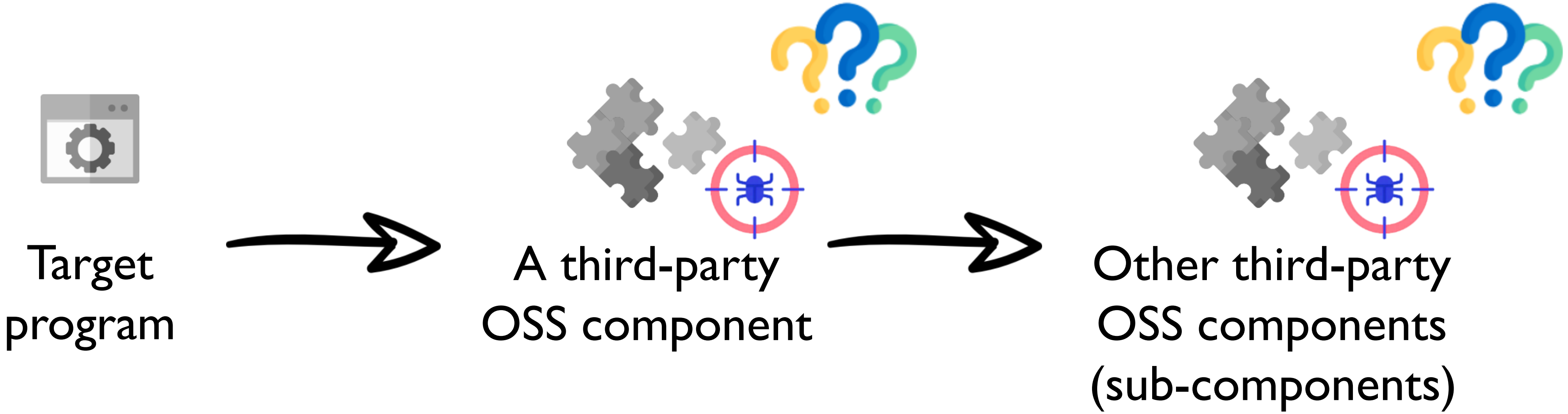
- Unmanaged OSS reuse can cause security threats



Motivation

Open-source software (OSS), a driving force behind innovative software development

- Unmanaged OSS reuse can cause security threats

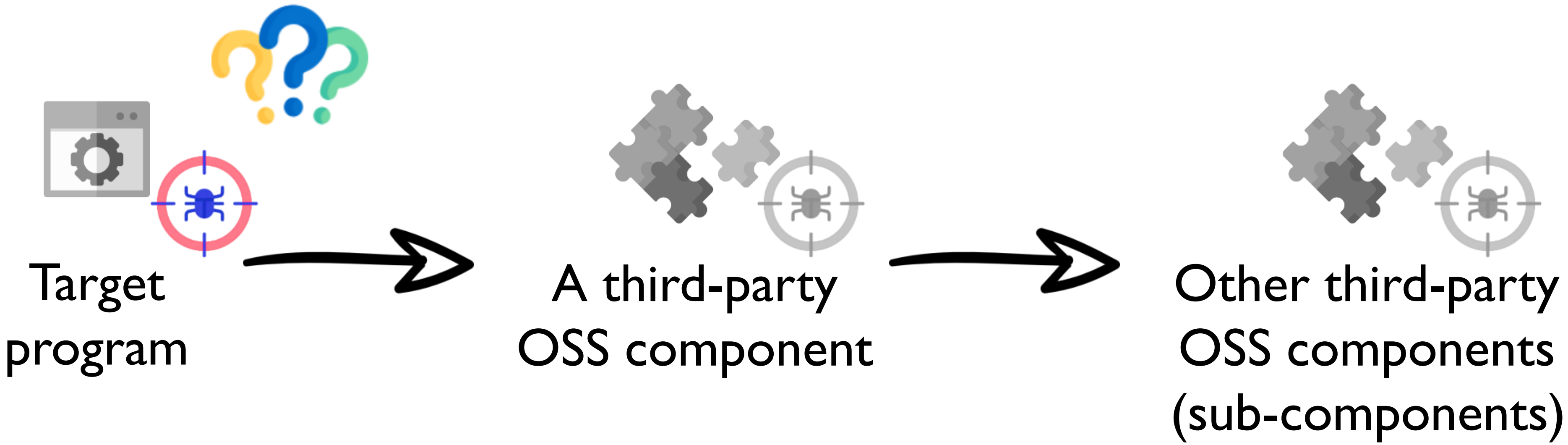


Q1. Do third-party OSS components contain vulnerabilities?

Motivation

Open-source software (OSS), a driving force behind innovative software development

- Unmanaged OSS reuse can cause security threats



Q2. Do vulnerabilities in third-party components exist in the target program?

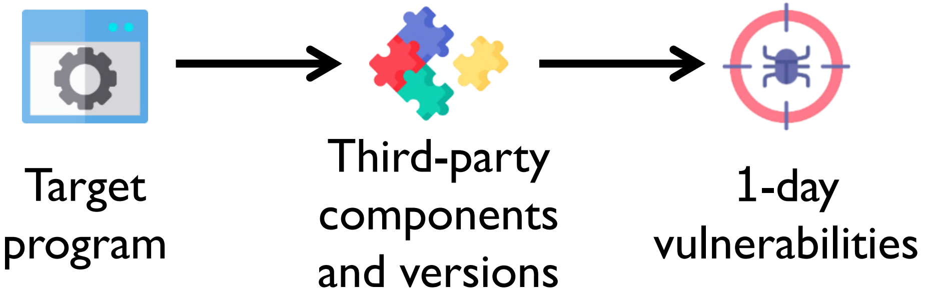
Motivation

Two main approaches for 1-day vulnerability discovery in C/C++ software

(1) Version-based approach

- Detecting vulnerabilities based on the version information of reused third-party OSS components

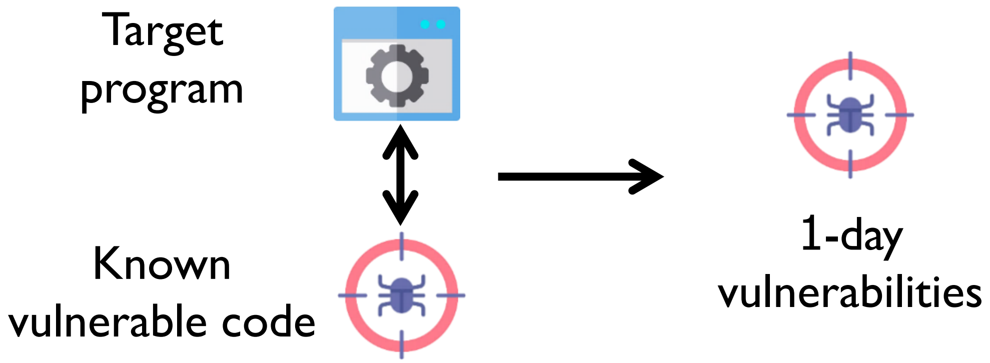
e.g., CENTRIS [ICSE '21], OSSFP [ICSE '23]



(2) Code-based approach

- Identifying codes syntactically or semantically similar to vulnerable code

e.g., VUDDY [S&P '17], MVP [SECURITY '20], MOVERY [SECURITY '22]



Challenge

Addressing **modified OSS reuse**

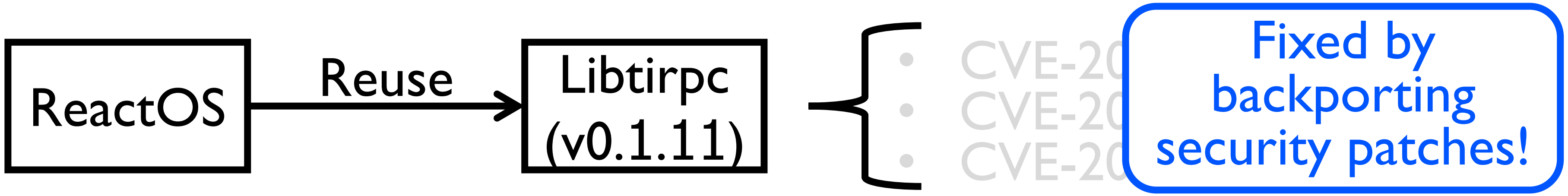
- Existing version-based approaches for C/C++ software
  - Producing false positives
  - Unused or resolved vulnerabilities cannot be addressed effectively

- Existing code-based approaches for C/C++ software
  - Producing false negatives
  - Vulnerabilities in modified code cannot be detected effectively

Challenge

Addressing **modified OSS reuse**

- Existing version-based approaches for C/C++ software
  - Producing false positives
  - Unused or resolved vulnerabilities cannot be addressed effectively



reactos/reactos

[LIBTIRPC] Fix CVE-2018-14622 by backporting its fix

CORE-15005

- Existing code-based approaches for C/C++ software
  - Producing false negatives
  - Vulnerabilities in modified code cannot be detected effectively

V1SCAN: Discovering 1-day Vulnerabilities in Reused C/C++ Open-source Software Components Using Code Classification Techniques

Design of V1SCAN

An approach for the precise and comprehensive discovery of 1-day vulnerabilities

- A new way to combine improved version- and code-based approaches
  - Key techniques: code classification techniques

Complementing FNs

Improved version-based approach

Improved code-based approach

Filtering out FPs

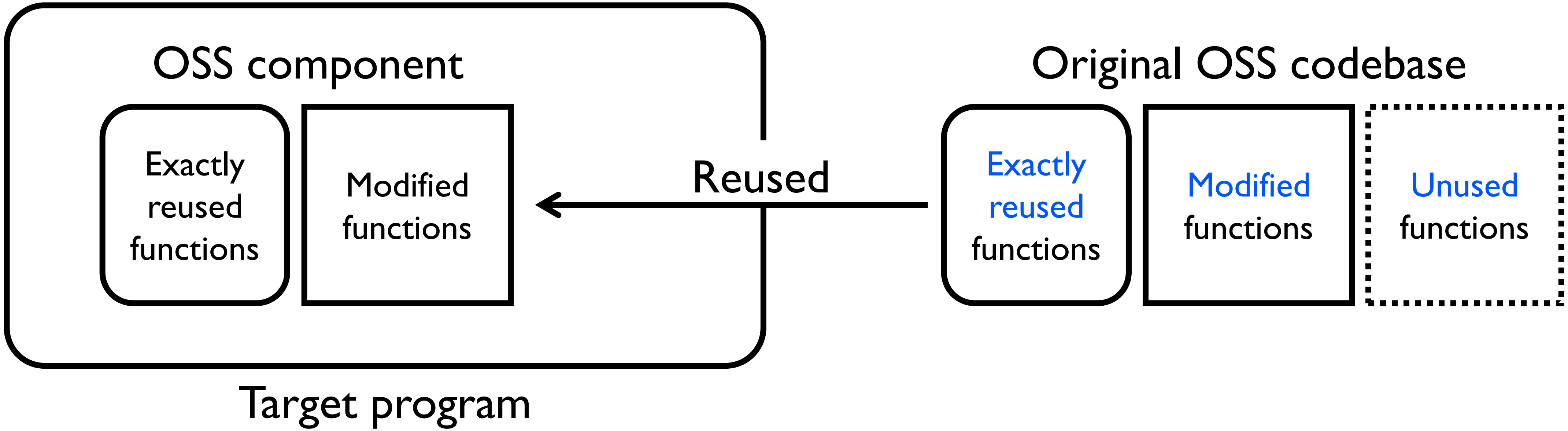
Reused code classification

Vulnerable code classification

Design of V1SCAN

Improved version-based approach

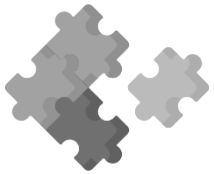
- Addressing false positives based on the reused code classification technique



Design of V1SCAN

Improved version-based approach

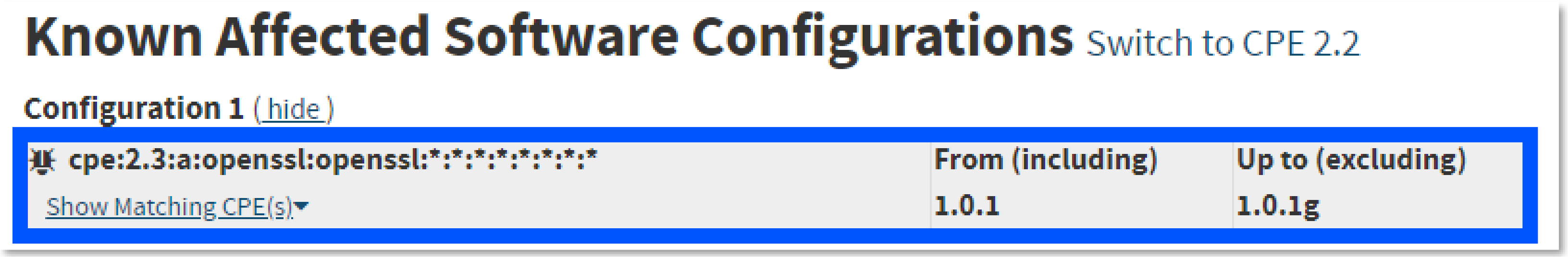
- Vulnerability detection



Third-party OSS components and versions

e.g., Common Platform Enumeration (CPE) of NVD →

Candidate vulnerabilities



Known Affected Software Configurations Switch to CPE 2.2

Configuration 1 (hide)

	From (including)	Up to (excluding)
cpe:2.3:a:openssl:openssl:*:*:*:*:*:*:*:*	1.0.1	1.0.1g
Show Matching CPE(s)		

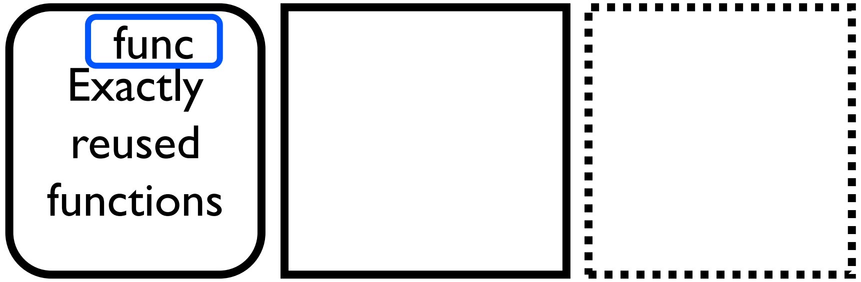
Example CPE for CVE-2014-0160

Design of V1SCAN

Improved version-based approach

- Filtering FPs (func : a vulnerable function of a detected vulnerability)
  - func is exactly reused in the target program
    - ❖ True vulnerability (no filtering is applied)

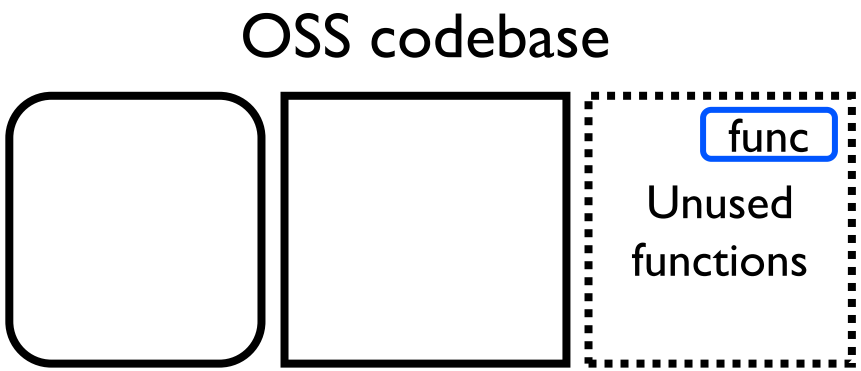
OSS codebase



Design of V1SCAN

Improved version-based approach

- Filtering FPs (func : a vulnerable function of a detected vulnerability)
  - func is not used in the target program
    - ❖ Filtering out func (i.e., false alarm)

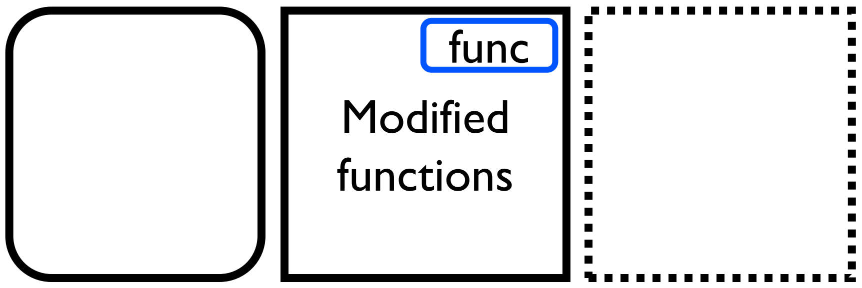


Design of V1SCAN

Improved version-based approach

- Filtering FPs (func : a vulnerable function of a detected vulnerability)
  - func is reused with code changes
    - ❖ Compare func to the vulnerable and patched functions of the vulnerability
      - \- func is more similar with the patched function → Filtering out func (e.g., backporting)

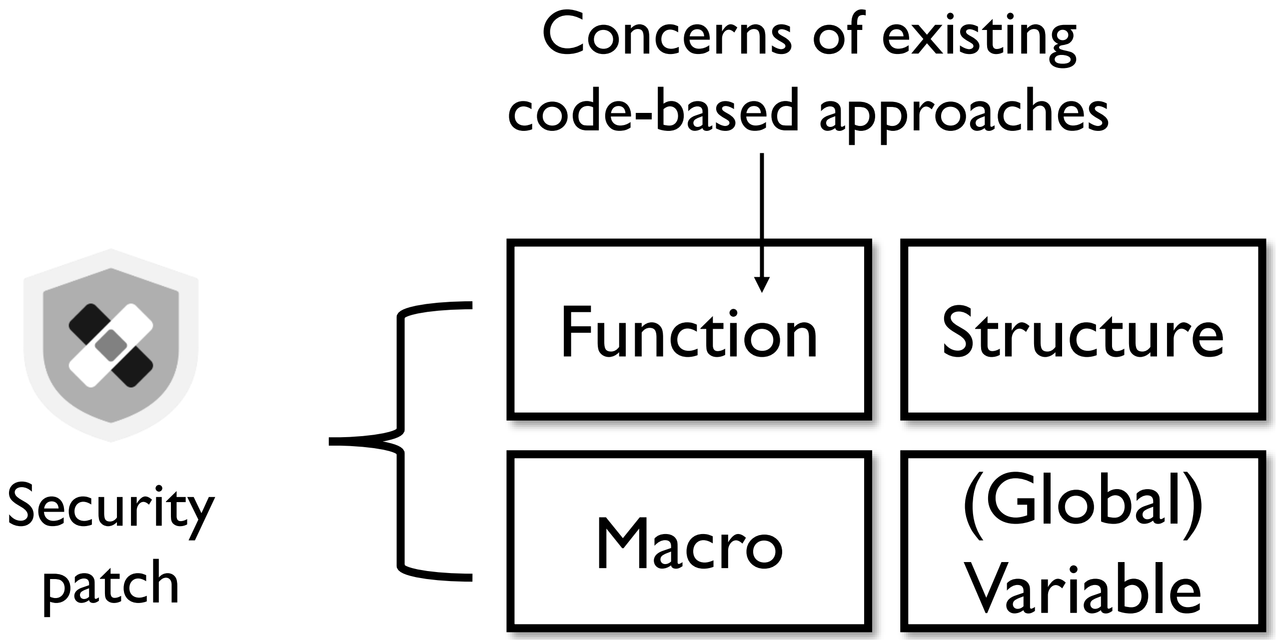
OSS codebase



Design of V1SCAN

Improved code-based approach

- Addressing false negatives based on the vulnerable code classification technique



(Over 97% of the code repaired by security patches was included in one of the four locations)

Design of V1SCAN

Improved code-based approach

- Signature generation

```
//libgcrypt/cipher/cipher-gcm.c
...
+ #ifdef HAVE_GCC_ATTRIBUTE_ALIGNED
+ # define ATTR_ALIGNED_64 __attribute__ ((aligned (64)))
...
- static const u16 gcmR[256] = {
- 0x0000, 0x01c2, 0x0384, 0x0246, 0x0708, 0x06ca, 0x048c,
...
+ static struct {
+   volatile u32 counter_head;
...
  void prefetch_table(const void *tab, size_t len) {
...
-   for (i = 0; i < len; i += 8 * 32)
+   for (i = 0; len - i >= 8 * 32; i += 8 * 32)
```

Security patch
(e.g., CVE-2019-12904)

```
MACRO
 + #ifdef HAVE_GCC_ATTRIBUTE_ALIGNED
 + # define ATTR_ALIGNED_64 __attribute__ ((aligned (64)))

VARIABLE
 - static const u16 gcmR[256] = {
 - 0x0000, 0x01c2, 0x0384, 0x0246, 0x0708, 0x06ca, 0x048c,

STRUCTURE (HASH: 3A5F116800...)
 + static struct {
 + volatile u32 counter_head;

FUNCTION (HASH: BBC0994B88...)
 - for (i = 0; i < len; i += 8 * 32)
 + for (i = 0; len - i >= 8 * 32; i += 8 * 32)
```

Vulnerability signature

Design of V1SCAN

Improved code-based approach

- Vulnerability detection: macro and variable
  - If the two conditions are satisfied, we conclude that 1-day vulnerabilities exist

All code lines deleted in the patch
(- code lines)

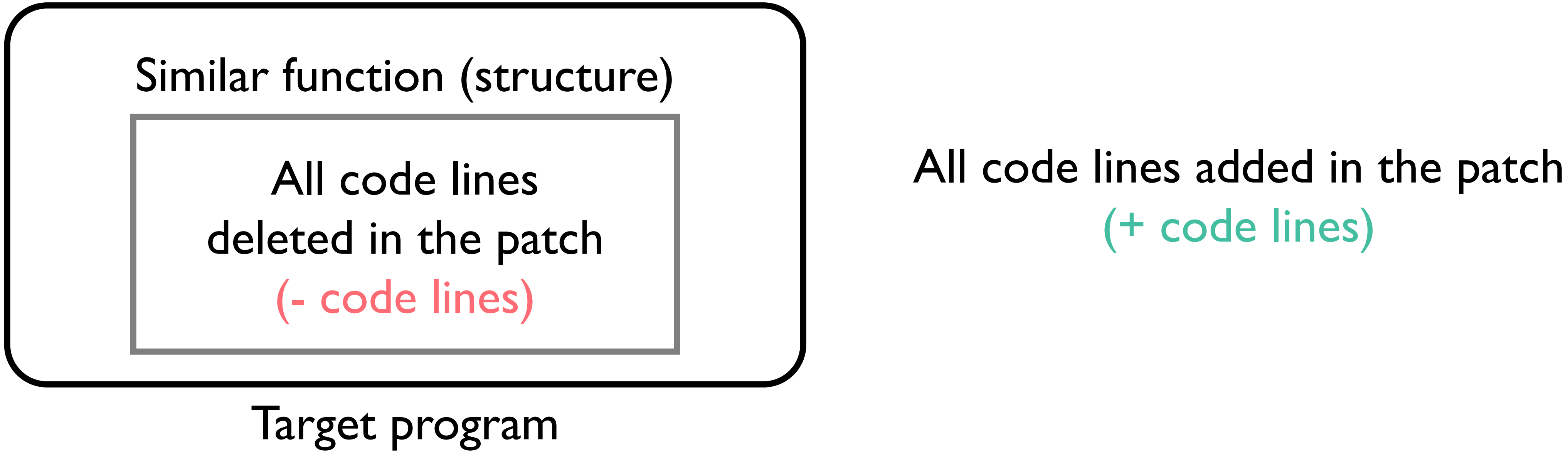
All code lines added in the patch
(+ code lines)

Target program

Design of V1SCAN

Improved code-based approach

- Vulnerability detection: function and structure
  - We first verify whether a function (or structure) similar to the vulnerable function exist
  - If the two conditions are satisfied, we conclude that 1-day vulnerabilities exist



Evaluation

Dataset

- CVE dataset (collected from NVD)
  - Vulnerable codes from 4,612 C/C++ security patches
    - Functions, structures, macros, and variables
- CPEs from all CVEs (as of August 2022)
- Target software dataset
  - Collected from GitHub
  - Popular, containing many OSS components

Table 4: Target software overview.

IDX	Name	Version	#CVE†	#OSS	#C/C++ Line	#Star§	Domain
S1	Turicreate	v6.4.1	69	28	4,091,413	10.7K	Machine learning
S2	ReactOS	v0.4.13	67	23	6,419,855	10.8K	Operating system
S3	TizenRT	3.0_GBM	62	22	2,156,848	439	Operating system
S4	Aseprite	v1.2.25	53	12	846,500	17K	Animation tool
S5	FreeBSD	v12.2.0	30	47	14,489,534	6.4K	Operating system
S6	MongoDB	r4.2.11	28	13	2,822,534	21.5K	Database
S7	MAME	0228	24	26	4,541,014	5.8K	Emulator
S8	Filament	v1.9.9	16	16	1,295,918	13.8K	Rendering engine
S9	Godot	v3.2.2	16	21	1,298,228	48.1K	Game engine
S10	ArangoDB	v3.6.12	15	22	5,465,881	12.2K	Database
Total	-	-	380	230	43,427,725	147K	-

†: #CVEs discovered by the version-based approach, §: #Stargazers.

Evaluation

Accuracy measurement

- Comparison targets: V0Finder [Security '21] and MOVERY [Security '22]
  - V1SCAN outperformed existing approaches
    - ❖Discovered 50% more 1-day vulnerabilities than MOVERY

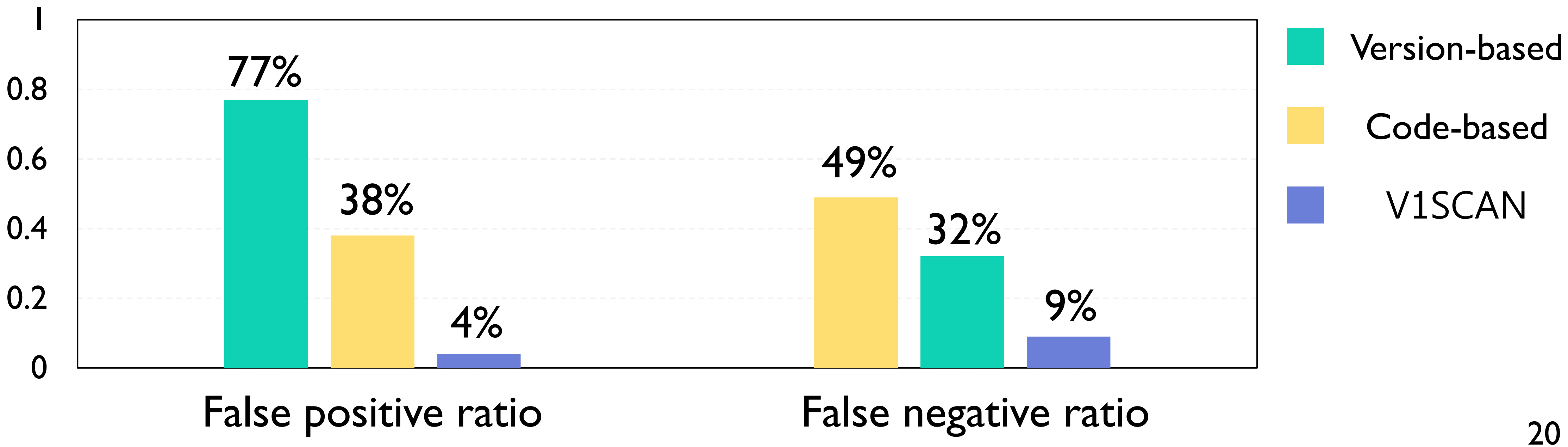
Target program	CVEs*	V1SCAN					MOVERY					V0Finder				
		#TP	#FP	#FN	P†	R‡	#TP	#FP	#FN	P	R	#TP	#FP	#FN	P	R
Turicreate	36	32	1	4	**0.97**	**0.89**	22	5	14	0.81	0.61	22	2	14	0.92	0.61
ReactOS	29	26	1	3	**0.96**	**0.90**	24	3	5	0.89	0.83	18	4	11	0.82	0.62
FreeBSD	23	19	2	4	**0.90**	**0.83**	13	4	10	0.76	0.57	12	7	11	0.63	0.52
MongoDB	14	14	0	0	**1.00**	**1.00**	4	0	10	**1.00**	0.29	4	0	10	**1.00**	0.29
Filament	14	14	0	0	**1.00**	**1.00**	10	0	4	**1.00**	0.71	4	0	10	**1.00**	0.29
TizenRT	10	9	0	1	**1.00**	**0.90**	4	1	6	0.80	0.40	3	1	7	0.75	0.30
Aseprite	8	8	0	0	**1.00**	**1.00**	6	0	2	**1.00**	0.75	1	1	7	0.50	0.13
MAME	8	7	2	1	0.78	**0.88**	6	1	2	**0.86**	0.75	2	1	6	0.67	0.25
Godot	4	4	0	0	**1.00**	**1.00**	1	3	3	0.25	0.25	1	2	3	0.33	0.25
ArangoDB	4	4	0	0	**1.00**	**1.00**	0	0	4	N/A	0.00	0	1	4	0.00	0.00
Total	150	137	6	13	0.96	0.91	90	17	60	0.84	0.60	67	19	83	0.78	0.45

CVEs*: Total number of TPs detected by V1SCAN, MOVERY, and V0Finder, P†: Precision, R‡: TP detection rate.

Evaluation

Effectiveness

- Comparison targets: CENTRIS [ICSE '21] (version-based) and VUDDY [S&P '17] (code-based)
  - V1SCAN reduced false positive ratio of the version-based approach from 77% to 4%
  - V1SCAN reduced false negative ratio of the code-based approach from 49% to 9%



Conclusion

Conclusion

- 1-day vulnerabilities have various propagation patterns
  - E.g., propagate with code modifications or resolved after propagation
- V1SCAN
  - An effective approach for discovering 1-day vulnerabilities in third-party OSS components
  - V1SCAN significantly outperformed existing approaches
    - ❖High vulnerability detection accuracy: 96% precision and 91% recall
- Equipped with vulnerability discovery results from V1SCAN
  - Developers can address threats caused by propagated vulnerabilities in OSS components