SBOM Technology Development and Challenges for Securing Software Supply Chains

ITU Workshop, August 28, 2023

Prof. Heejo Lee
Dept. of Computer Science and Engineering
Korea University
(heejo@korea.ac.kr)
This is a joint work with Yoonjong Na.

About Speaker

Heejo Lee at Korea University
(heejo@korea.ac.kr)

- Professor, Dept. of Computer Sci. and Eng., Korea Univ. (2004-present)
- Director, Center for Software Security and Assurance (CSSA) (2015-present)
- Co-CEO, Labrador Labs, Inc. (CSSA Spin-off since 2018)
- Visiting Professor, CyLab/CMU (2010-2011)
- CTO, AhnLab Inc. (2001-2003)
- Editor, IEEE Trans. on Vehicular Technology, and Journal of Comm. and Networks
- ISC2 ISLA award winner of community service star in 2016
- Postdoc researcher, CERIAS at Purdue University (2000-2001)
- BS, MS, PhD from POSTECH, Korea (1989-2000)
- **PI, "Development of SBOM Technologies for Securing Software Supply Chains",** MSIT/IITP, Korea, 2022-2025

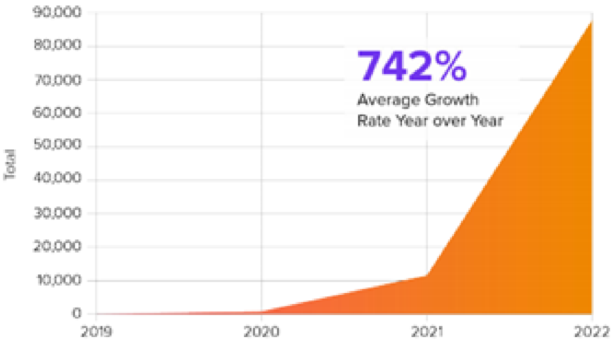
SBOM Overview

Increase of supply chain attacks

- SW supply chains rely on OSS ecosystems
  OSS (open source software) commonly reused by other OSS
- Dependency problem in OSS reuses
  Delayed updates of reused OSS result in vulnerability propagation
- Common attack surface of supply chain attacks
  Increase in frequency and patterns of supply chain attack

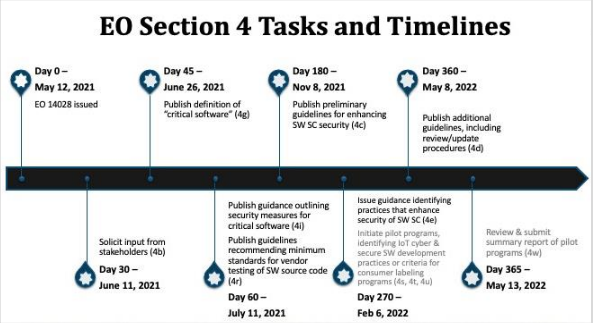
Why we need SBOM

- Software Bill of Materials (SBOM)
  A statement of SW components including reused OSS
- Advantages of SBOM management
  Provide transparency of SW supply chains
- US EO and EU CRA include SBOM regulations
  A project launched for developing SBOM technologies in Korea



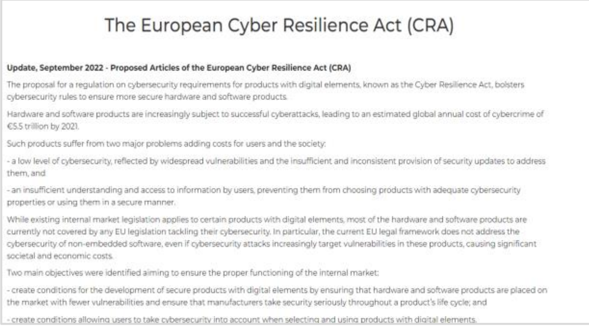
"Supply chain attack increase 742%"

(Sonatype, 2022)


EO Section 4 Tasks and Timelines
Day 0 –
May 12, 2021
EO 14028 issued
Day 45 –
June 26, 2021
Publish definition of "critical software" (4g)
Day 180 –
Nov 8, 2021
Publish preliminary guidelines for enhancing SW SC security (4c)
Day 360 –
May 8, 2022
Publish additional guidelines, including review/update procedures (4d)
Solicit input from stakeholders (4b)
Day 30 –
June 11, 2021
Publish guidance outlining security measures for critical software (4i)
Publish guidelines recommending minimum standards for vendor testing of SW source code (4r)
Day 60 –
July 11, 2021
Issue guidance identifying practices that enhance security of SW SC (4e)
Initiate pilot programs, identifying IoT cyber & secure SW development practices or criteria for consumer labeling programs (4s, 4t, 4u)
Day 270 –
Feb 6, 2022
Review & submit summary report of pilot programs (4w)
Day 365 –
May 13, 2022

"US Executive Order"

(US, 2021)



The European Cyber Resilience Act (CRA)

Update, September 2022 - Proposed Articles of the European Cyber Resilience Act (CRA)

The proposal for a regulation on cybersecurity requirements for products with digital elements, known as the Cyber Resilience Act, bolsters cybersecurity rules to ensure more secure hardware and software products.

Hardware and software products are increasingly subject to successful cyberattacks, leading to an estimated global annual cost of cybercrime of €5.5 trillion by 2021.

Such products suffer from two major problems adding costs for users and the society:

- a low level of cybersecurity, reflected by widespread vulnerabilities and the insufficient and inconsistent provision of security updates to address them, and

- an insufficient understanding and access to information by users, preventing them from choosing products with adequate cybersecurity properties or using them in a secure manner.

While existing internal market legislation applies to certain products with digital elements, most of the hardware and software products are currently not covered by any EU legislation tackling their cybersecurity. In particular, the current EU legal framework does not address the cybersecurity of non-embedded software, even if cybersecurity attacks increasingly target vulnerabilities in these products, causing significant societal and economic costs.

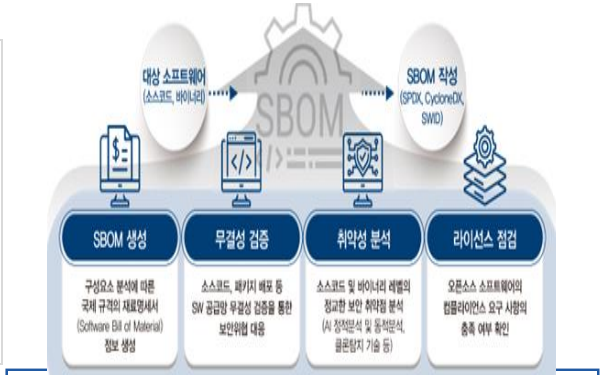
Two main objectives were identified aiming to ensure the proper functioning of the internal market:

- create conditions for the development of secure products with digital elements by ensuring that hardware and software products are placed on the market with fewer vulnerabilities and ensure that manufacturers take security seriously throughout a product's life cycle; and

- create conditions allowing users to take cybersecurity into account when selecting and using products with digital elements.

"EU Cyber Resilience Act"

(EU, 2022)


대상 소프트웨어
(소스코드, 바이너리)
SBOM
SBOM 작성
(SPDX, CycloneDX, SWID)
SBOM 생성
구성요소 분석에 따른 국제 규격의 재료명세서 (Software Bill of Material) 정보 생성
무결성 검증
소스코드, 패키지 배포 등 SW 공급망 무결성 검증을 통한 보안위협 대응
취약성 분석
소스코드 및 바이너리 레벨의 정교한 보안 취약점 분석 (AI 정적분석 및 동적분석, 클론탐지 기술 등)
라이선스 점검
오픈소스 소프트웨어의 컴플라이언스 요구 사항의 충족 여부 확인

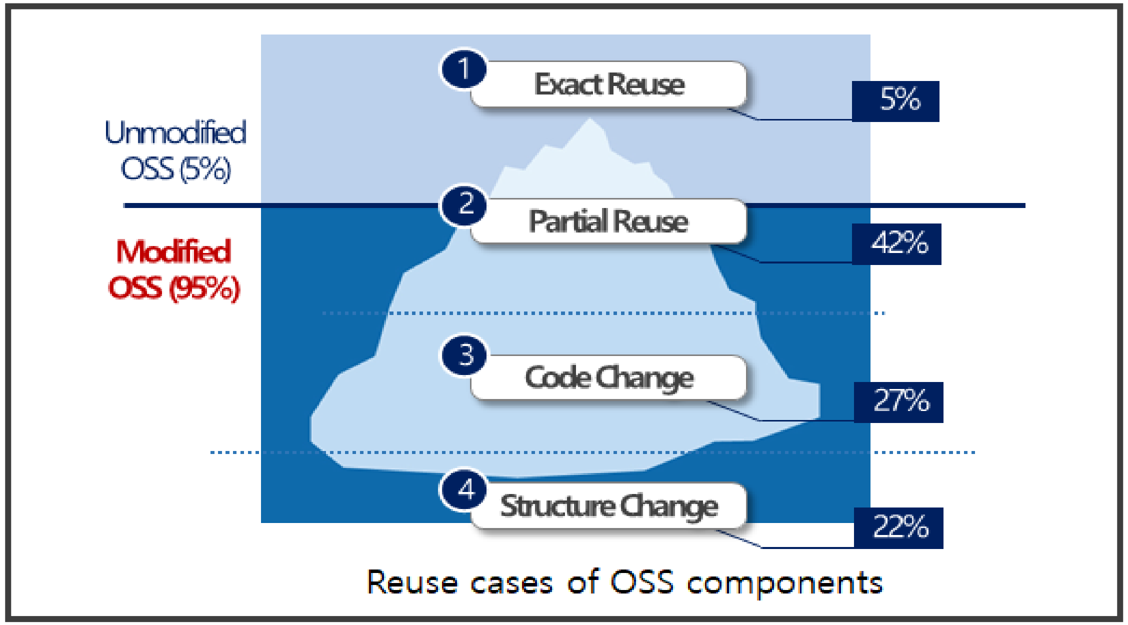
"Development of SBOM Technologies"

(KR, 2022-25)

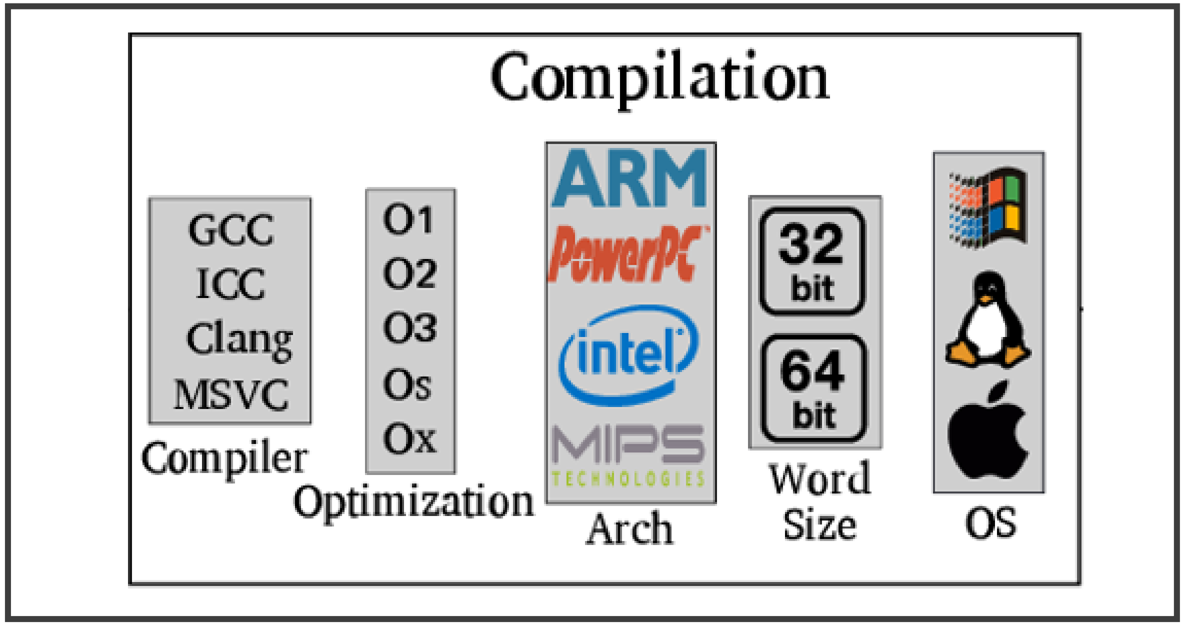
Challenges in SBOM Generation: Accuracy

Difficulty of generating precise SBOM

- Source code – Detection of reused OSS is challenging due to **partial reuse and modification of OSS**
- Binary code – Detection of reused OSS is challenging due to **diversity of compile environments**



Source – modification of OSS makes accurate component detection difficult (CENTRIS, ICSE 2022)



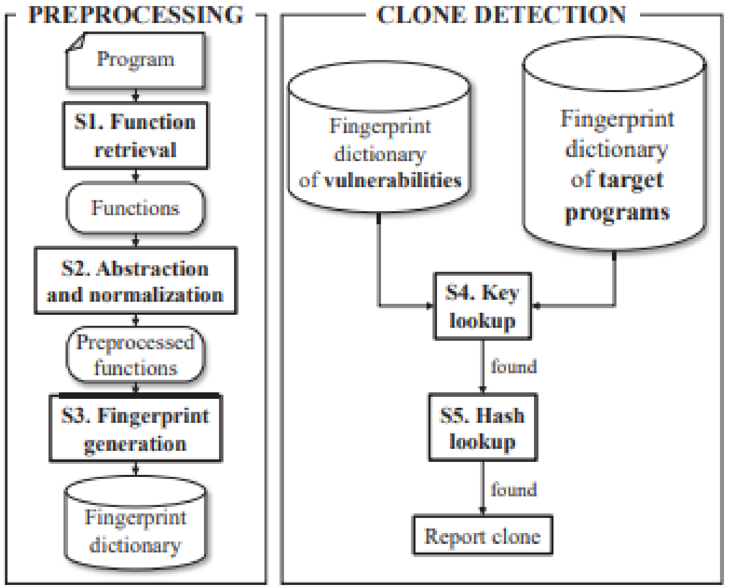
Binary – various compile environments make OSS detection difficult

Challenges in SBOM Management: Exploitability

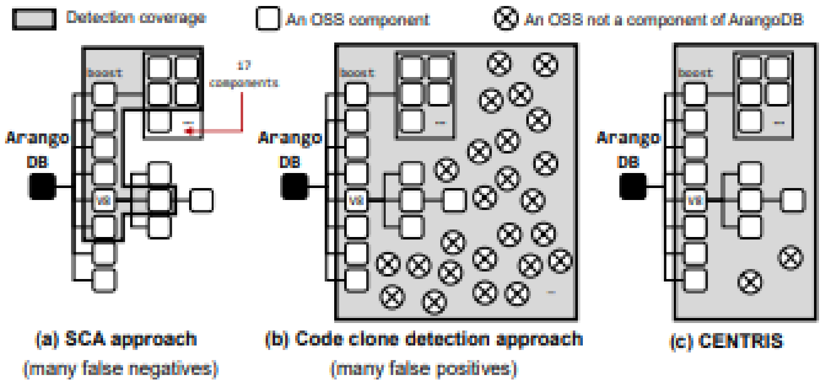
Exploitability of vulnerabilities

- SBOM allows to recognize vulnerable components, by the use their name and version

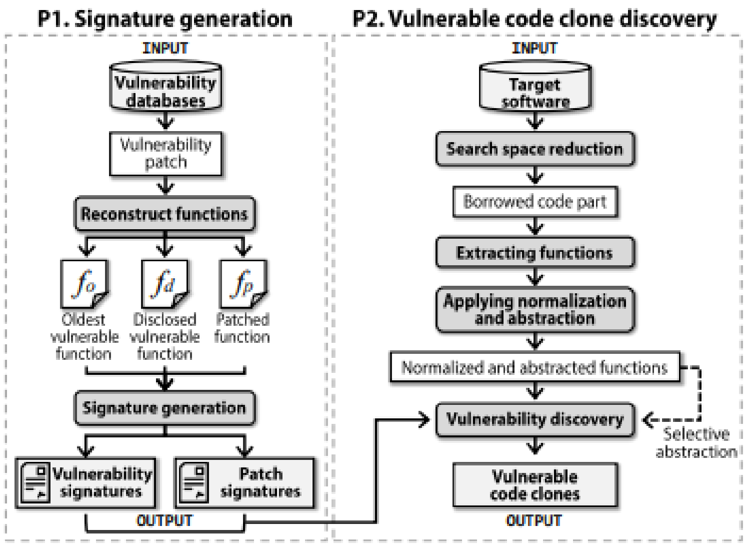
  However, version-based vulnerability detection (SBOM analysis) yields 77% false positives (V1scan; USENIX Security'23)
- The exploitability of each vulnerability found is very hard to be determined in a systematic way



"VUDDY: A Scalable Approach for Vulnerable Code Clone Discovery" (IEEE S&P, 2017)



"CENTRIS: A Precise and Scalable Approach for Identifying Modified OSS Reuse" (ICSE, 2021)



"V1SCAN: Discovering 1-day Vulnerabilities in Reused C/C++ Open-source Software Components Using Code Classification Techniques" (USENIX Security, 2023)

HatBOM: An Automated SBOM Tool in the IoTcube Platform

- HatBOM: The SBOM Caring Hat (developed by CSSA, Korea University)
  - A collective tool of SBOM operations, which are available at https://iotcube.net
  - HatBOM provides operations of build, view, translate, merge, diff, and validate, which cover most SBOM operations (7 out of 9) proposed by NTIA SBOM Tool Taxonomy
  - Top 6 SBOM tools in GitHub star ranks were compared with HatBOM
    - https://github.com/awesomeSBOM/awesome-sbom

C: CycloneDX SBOM

S: SPDX SBOM

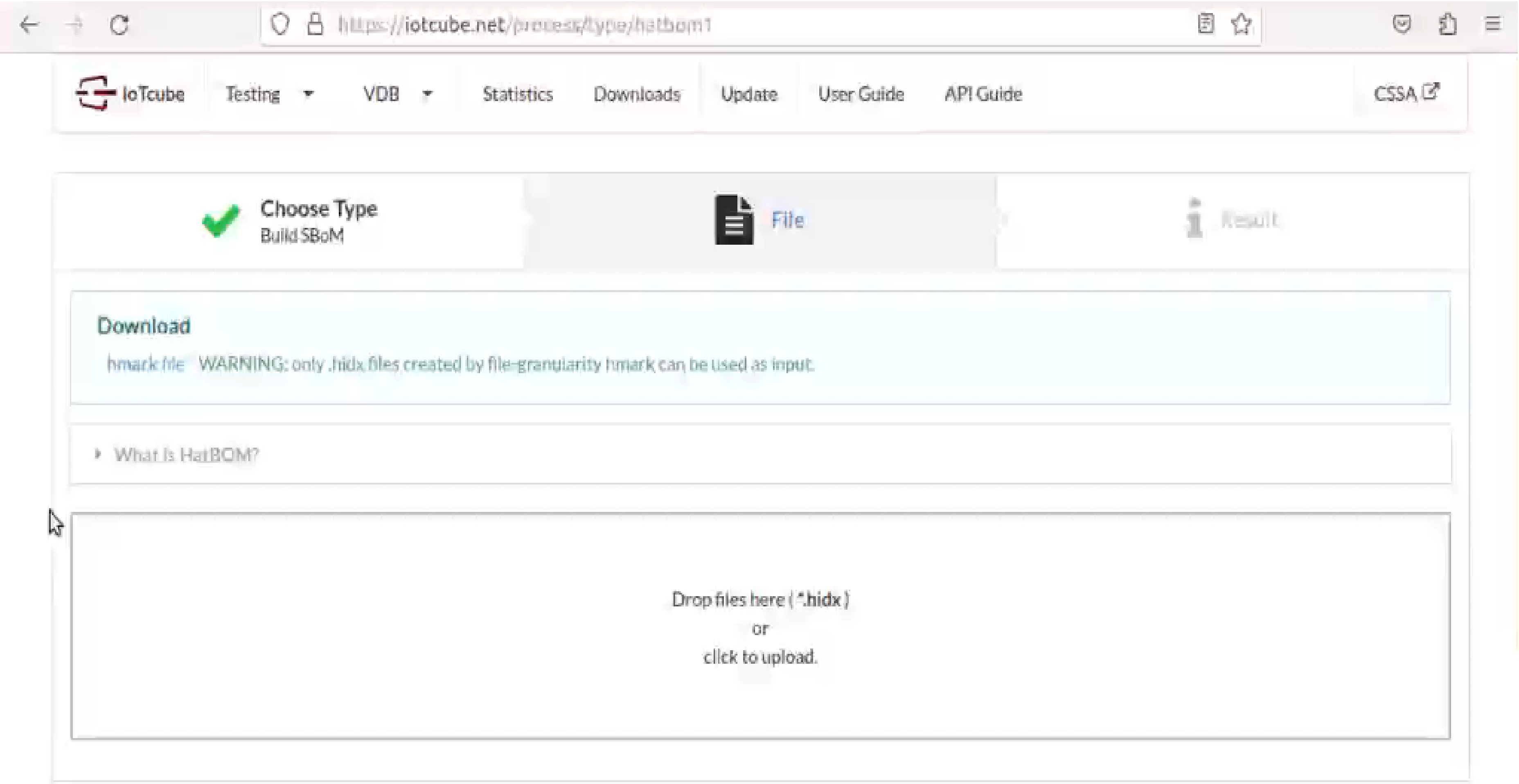
Δ: To be supported

SBOM Tool	F1 (Build)	F2 (Analyze)	F3 (Edit)	F4 (View)	F5 (Diff)	F6 (Import)	F7 (Translate)	F8 (Merge)	F9 (Tool Support)
bomber (DFKM)		C, S		C, S					
MS SBOM Tool	S								
Syft	C, S	C, S	C, S						
Tern	C, S								
Aqua Trivy	C, S	C, S	C, S						
CycloneDX CLI			C		C		C, S	C	
HatBOM	**Δ**	**C, S**		**C, S**	**C**	**C, S**	**C, S**	**C**	**Δ**

NOT SUPPORTED

How HatBOM works

- IoTcube Hatbom for SBOM operations with Redis 6 and Redis 7



Download

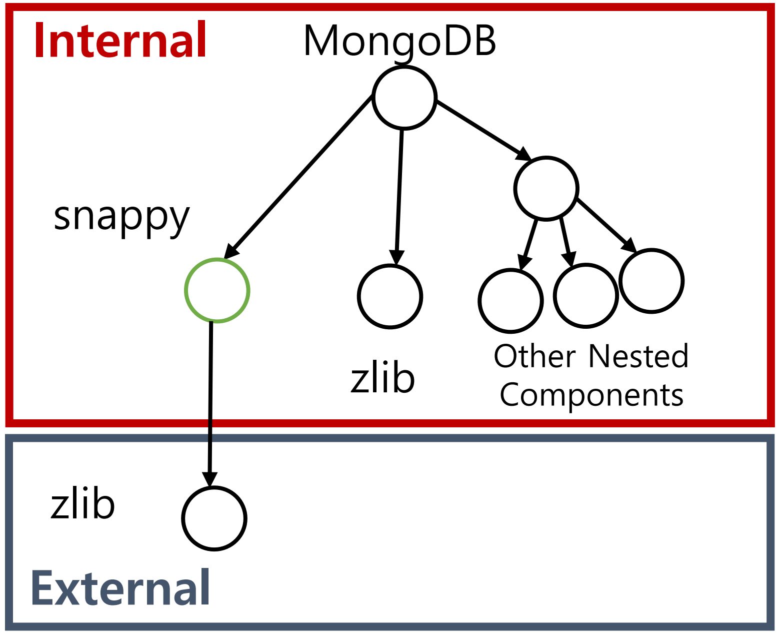
hmark file WARNING: only .hidx files created by file-granularity hmark can be used as input.

Drop files here (*.hidx)
or
click to upload.

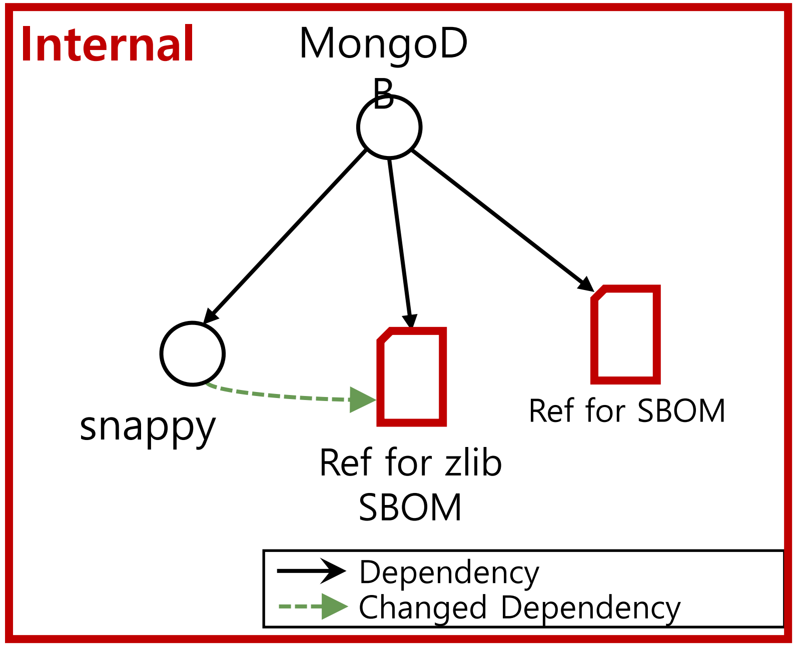
1. How to describe dependencies

Standard status

- Dependencies among reused components are complicated and even nested
- Minimizing SBOM by only showing top-level and modified components can improve readability



Expected dependency of MongoDB



Actual Dependency of MongoDB

2. How to store the compile environments

Standard status

- SPDX currently provides very few features for binary SBOM
- Even in the same source code, different output will come in a different compile environment

Countermeasures

- We are studying on additional fields that can help component detection in binary SW
  - Unit for binary analysis of OSS components
  - Provisioning compile environments such as compiler, options and build environments
  - Describing component version of binary files

3. How to verify the non-existence of vulnerabilities

Standard status

- Vulnerability databases (NVD, Google OSV, GSD) do not provide sufficient information
  - Common vulnerability database like STIX™ 2.1, MITRE CVE and NVD provide vulnerability information, but insufficient for verifying the existence of the vulnerability in the target software
  - The hash values of vulnerable and patched functions will help to determine the existence of the vulnerability in the software

Additional fields for vulnerability databases

TTAK.KO-12.0384

	Field	Description	Required
N1	Vulnerability origin software name	The software name where vulnerability is first discovered	Required
N2	The version of the origin software	The version the of origin software where vulnerability is first discovered	Required
N3	Vulnerable file/function source code	The source code of the function containing vulnerability (e.g., Github link)	Optional
N4	**Vulnerable file/function hash value**	The hash value of a file or function containing the vulnerability	Required
N5	Vulnerability exploit approach	The description for exploiting the vulnerability (e.g., PoC)	Optional
N6	Vulnerability path information	The path of vulnerability in the origin software	Optional
N7	Vulnerability patch	The information containing vulnerability (e.g., "patch" file)	Optional
N8	**Patched file/function hash value**	The hash value of the patched file or function	Required

Takeaways and Conclusions

- Precise SBOM generation is challenging
  - Modified OSS components and dependency changes make it hard to generate a correct SBOM
  - Diverse compile environments make binary component detection difficult
- Vulnerability scanning in SBOM is challenging
  - Additional information needs to be considered for vulnerability database to improve precision
  - Security patch sharing system should also be considered

Suggestions for SG17

- How can we collaborate in order to overcome these technical challenges?
  - Research collaboration and standard efforts can inspire us to find solutions!

Thank you~
(Q&A)

How to Contact: ***IoTcube finds all bugs!***

- KU CSSA: https://iotcube.net, cssa@korea.ac.kr
- Labrador Labs Inc.: https://labradorlabs.ai, contact@labradorlabs.ai