KOREA UNIVERSITY
Center for Software Security and Assurance

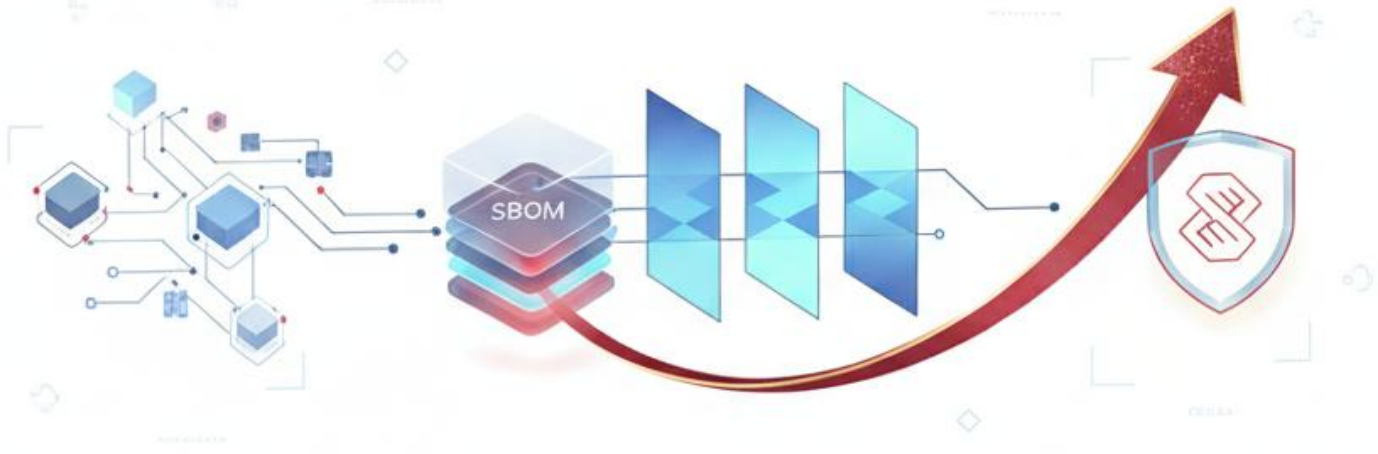
Securing Open Source with SBOM

From Visibility To Trust

Heejo Lee

Professor, Korea University

heejo@korea.ac.kr



Wednesday, November 5, 2025 | 16:35-17:05 KST
The Linux Foundation, Open Source Summit Korea 2025

KOREA UNIVERSITY
Center for Software Security and Assurance

Prof. Heejo Lee

Korea University
Dept. of Computer Science and Engineering
heejo@korea.ac.kr

Professor at Korea University and the **director of CSSA**, interested in open-source security and automated vulnerability analysis

📌 Key Achievements

- Founded South Korea’s first white hacker club 33 years ago
- Developing **IoTcube.net**, an automated vulnerability analysis platform
- Co-founded Labrador Labs, an SCA solution provider
- Former CTO of AhnLab, leading cybersecurity company in Korea
- Representing Korea as a cybersecurity policy advisor since 2006, joining the consultation of several Asian and American countries
- Recipient of the (ISC)² ISLA Award of Community Service Star (2016)

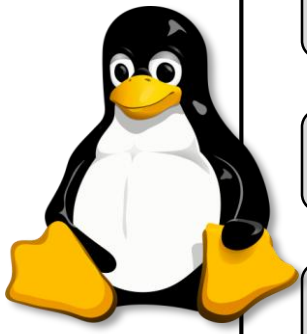
Our Journey Today

1. Open Source: Where Innovation Begins
2. Security Vulnerabilities and Challenges
    - Challenge #1 — Modified Components
    - Challenge #2 — Dependency Explosion
    - Challenge #3 — Name Mismatches
    - Challenge #4 — Incorrect CVE Mapping
3. Introducing **Hatbom (IoTcube 2.0)** Platform
4. Conclusion: Transparency Builds Trust

1. Open Source: Where Innovation Begins

KOREA UNIVERSITY
Center for Software Security and Assurance

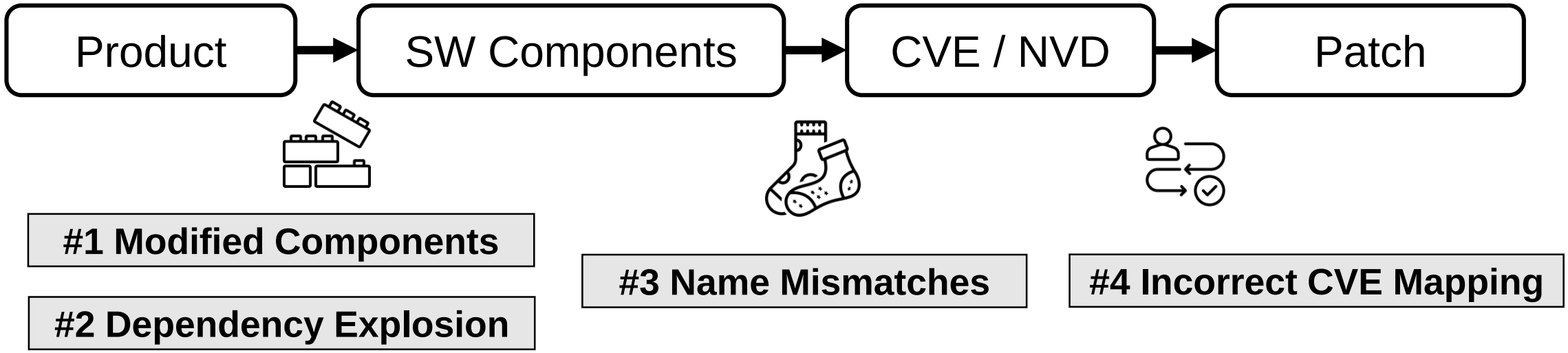
- **Open source drives innovation** across industries
  - From Linux, diverse platforms emerge: Android, AGL, ROS2 powering phones, cars, robots
- But shared components mean shared vulnerabilities
  - **Vulnerability management** is required for upstream and downstream projects



Linux		
Android		Smartphones, tablets, smart watches
AGL		In-vehicle infotainment, digital dashboards, ADAS
ROS2		Service robots, drones, autonomous vehicles
Ubuntu		Cloud servers, enterprise systems, research clusters
Tizen / WebOS		Smart TVs, home appliances, wearable devices

2. Security Vulnerabilities and Challenges

- How to do vulnerability management of a product or service
  - Find a list of SW components and map into CVE vulnerabilities for patching vulnerabilities
  - But there are **four main challenges** for vulnerability management

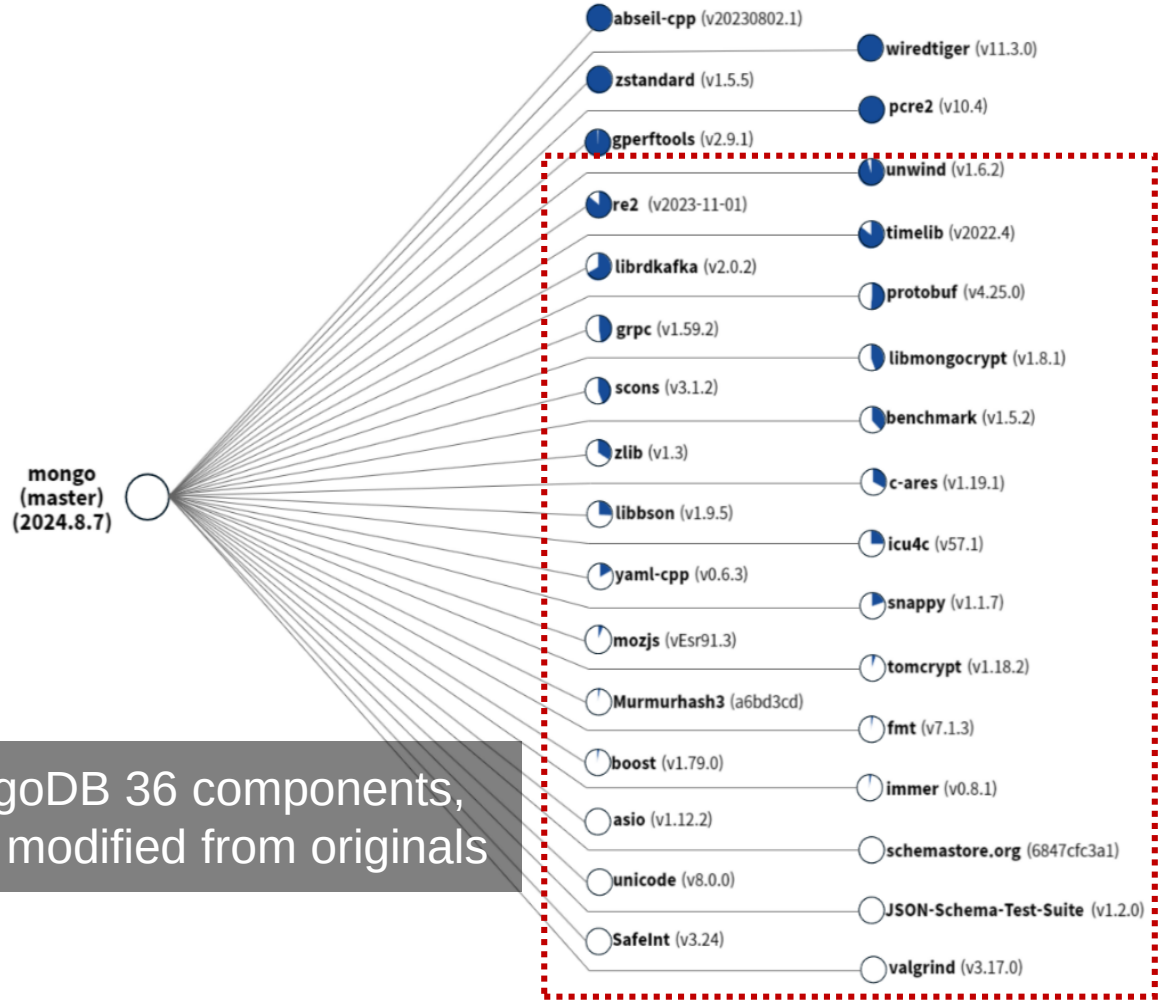


The four challenges make it difficult to find and fix critical vulnerabilities

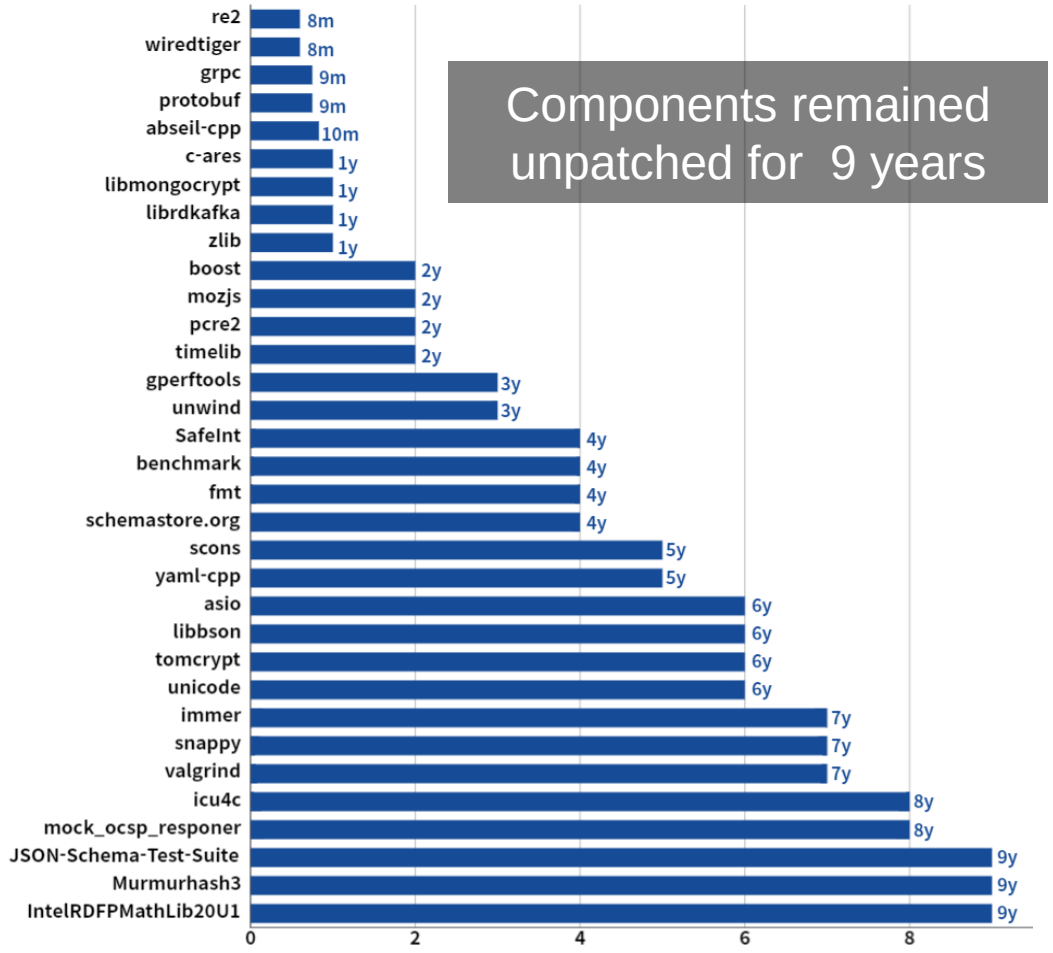
Challenge #1 — Modified Components

KOREA UNIVERSITY
Center for Software Security and Assurance

- Invisible components when reused with modification (Centris, ICSE 2021)
  - MongoDB includes 36 components, some of which are 9 years old



MongoDB 36 components, many modified from originals

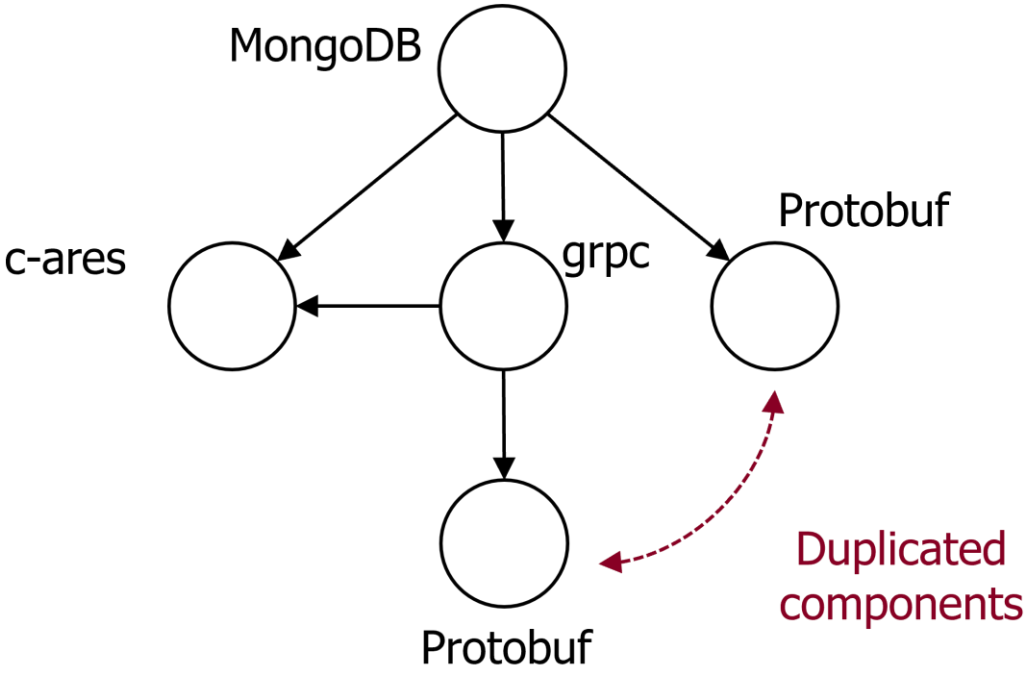


Components remained unpatched for 9 years

Challenge #2 — Dependency Explosion

KOREA UNIVERSITY
Center for Software Security and Assurance

- OSS projects use other OSS projects (Cneps, ICSE 2024)
  - One OSS project has 10 direct dependencies, which expand into 683 transitive ones
  - Using one library implicitly brings hundreds more, making vulnerability management harder



- Two copies of Protobuf
  - Need to patch two locations
- Two dependencies in c-ares
  - Check backward compatibility in both MongoDB and grpc

Challenge #3 — Name Mismatches

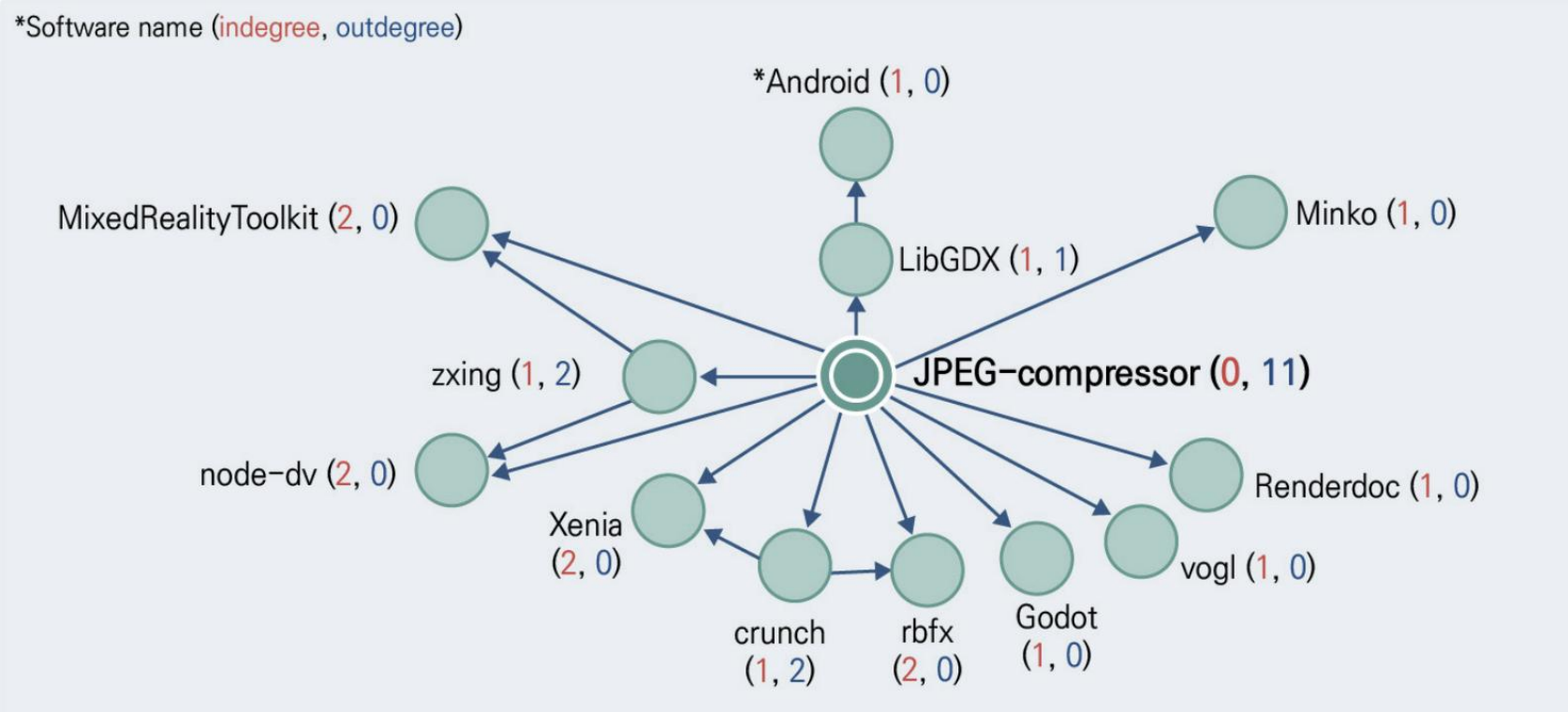
- No standard naming conventions
  - Inconsistencies between OSS projects and CVE databases can cause **missed vulnerabilities**
  - Inconsistent naming increases the risk of name confusion attacks, such as typo-squatting

CVE Description	CPE (vendor, product)	Project Repository
Linux Kernel	**linux**:linux_kernel	github.com/**torvalds**/linux
Tensorflow	**google**:tensorflow	github.com/**tensorflow**/tensorflow
MongoDB	mongodb:**mongodb**	github.com/mongodb/**mongo**
Kerberos5	**mit**:**kerberos_5**	github.com/**krb5**/**krb5**
LibXML2	**xmlsoft**:libxml2	gitlab.gnome.org/**GNOME**/libxml2

Challenge #4 — Incorrect CVE Mapping

KOREA UNIVERSITY
Center for Software Security and Assurance

- Misclassified vulnerabilities in CVEs (V0finder, Usenix Security 2021)
  - Vulnerabilities for wrong products leading to blind spots
  - Android, rather than JPEG-compressor, being registered for CVE-2017-0700, which interferes vulnerability disclosure and causes vulnerable projects remain unpatched



2. Security Vulnerabilities and Challenges

- To manage these challenges, we need visibility and context
  - **SBOM** and **VEX** are supply chain security frameworks designed to enhance product security
  - As adopted in global regulations such as US EO 14028, EU CRA, US FDA, and KR MFDS

	Software Bill of Materials (SBOM)	**Vulnerability Exploitability eXchange (VEX)**
Role	**Transparency** — lists all OSS & proprietary components	**Risk prioritization** — states whether known vulnerabilities actually affect the product
Focus	What's inside	What's exploitable ※ *Affected / Not affected / Fixed / Under investigation*
Together	SBOM + VEX = Visibility + Actionability — clarity amid noise (e.g., not every Log4Shell was exploitable).	

2. Security Vulnerabilities and Challenges

- Current SBOM tools lack sufficient visibility into reused components
  - Beyond metadata, components and vulnerabilities must be **analyzed at the code level**

	Components		Vulnerabilities		SBOM		VEX Export	Other
	Meta	Code	Meta	Code	SPDX	Cyclone DX		
GitHub Export	O	X	X	X	O	X	X	
Syft	O	X	△	X	O	O	△	△: Supported via external tools (Grype)
Cdxgen	O	O	△	X	O	△	△	△: Supported via external tools (depscan and CycloneDX CLI)
IoTcube Hatbom	△*	O	X	O	△	O	O	△*: Planned for support in 2026 △: Supported by IoTcube 1.0

3. Introducing Hatbom (IoTcube 2.0) Platform

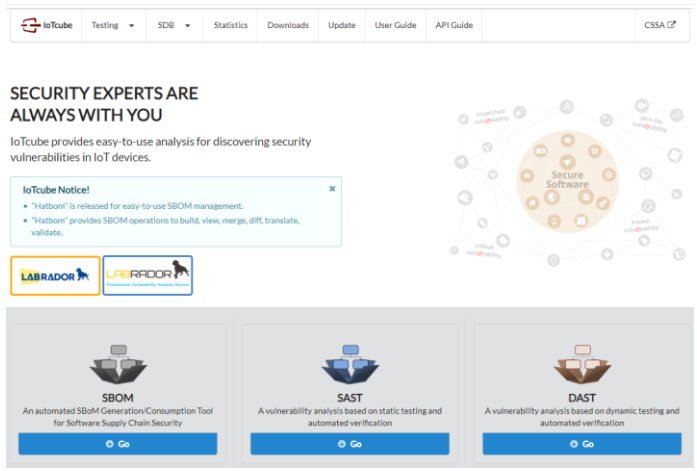
KOREA UNIVERSITY
Center for Software Security and Assurance

CSSA (2015~)

International research center at Korea University advancing software supply-chain security through developing SBOM/VEX technologies

https://cssa.korea.edu/

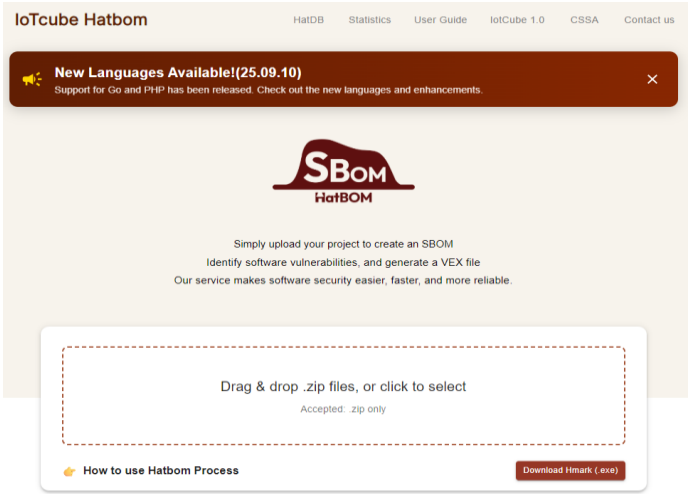
IoTcube.net (2016~)



Automated vulnerability analysis platform with 20+ security tools from top conference papers and 27K+ users worldwide

https://iotqv.korea.ac.kr

Hatbom (2025.08~)



One-stop platform for SBOM generation and VEX verification, powered by our Centris, Cneps, and Vuddy algorithms

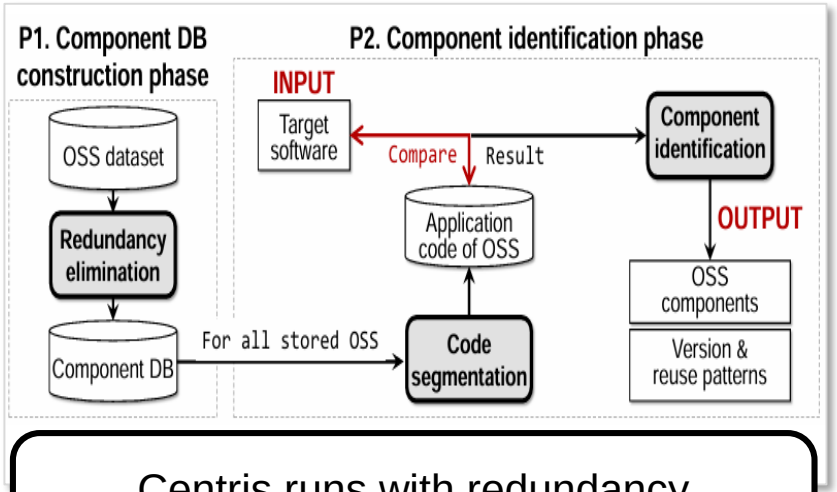
https://iotcube.net/

※ **Labrador Labs, Inc.** is a KU spin-off providing SBOM and open-source security solutions, operating both domestically and internationally (2018~, https://labradorlabs.ai)

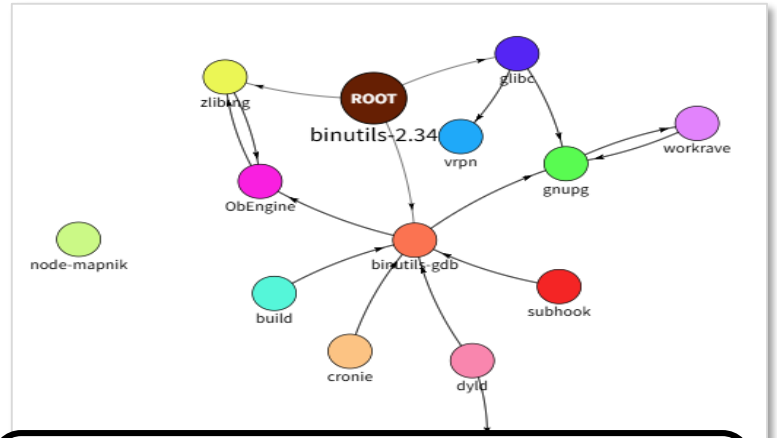
Code-level Scanning Algorithms

KOREA UNIVERSITY
Center for Software Security and Assurance

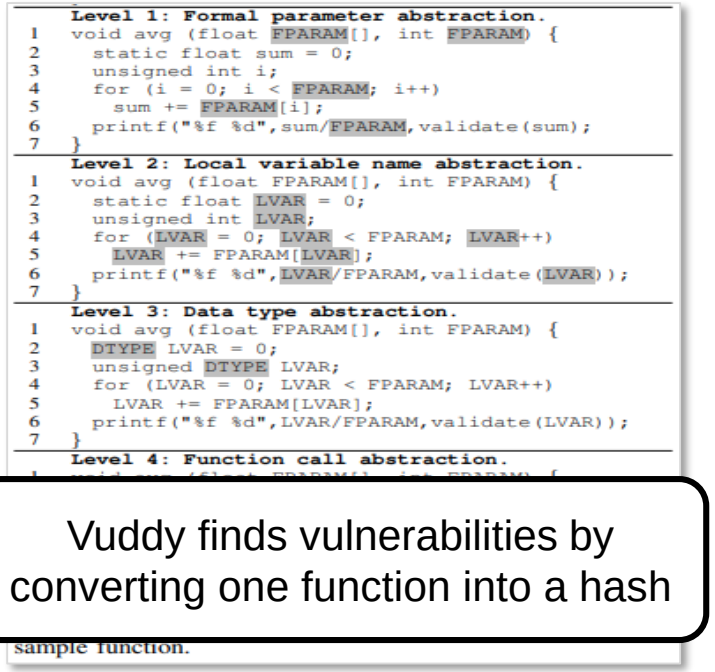
- "How do I accurately detect OSS components?" – Centris, ICSE 2021
  - Centris detects over 90% components with modification — extending SBOM visibility and trust
- "How do I accurately detect OSS dependencies?" – Cneps, ICSE 2024
  - Cneps provides the relationship among components with 75% more dependencies
- "How do I find hidden vulnerabilities in OSS?" – Vuddy, IEEE S&P 2017
  - Vuddy finds vulnerabilities 1,000x faster than existing methods with higher accuracy



Centris runs with redundancy elimination and code segmentation



Cneps traces module dependencies rather than function calls



Level 1: Formal parameter abstraction.

```
void avg (float FPARAM[], int FPARAM) {
  static float sum = 0;
  unsigned int i;
  for (i = 0; i < FPARAM; i++)
    sum += FPARAM[i];
  printf("%f %d",sum/FPARAM,validate(sum));
}
```

Level 2: Local variable name abstraction.

```
void avg (float FPARAM[], int FPARAM) {
  static float LVAR = 0;
  unsigned int LVAR;
  for (LVAR = 0; LVAR < FPARAM; LVAR++)
    LVAR += FPARAM[LVAR];
  printf("%f %d",LVAR/FPARAM,validate(LVAR));
}
```

Level 3: Data type abstraction.

```
void avg (float FPARAM[], int FPARAM) {
  DTYPE LVAR = 0;
  unsigned DTYPE LVAR;
  for (LVAR = 0; LVAR < FPARAM; LVAR++)
    LVAR += FPARAM[LVAR];
  printf("%f %d",LVAR/FPARAM,validate(LVAR));
}
```

Level 4: Function call abstraction.

sample function.

Vuddy finds vulnerabilities by converting one function into a hash

IoTcube Hatbom, https://iotcube.net

STEP1. SBOM Generation

4. Conclusion: Transparency Builds Trust

KOREA UNIVERSITY
Center for Software Security and Assurance

- Open source powers innovation — yet trust begins with transparency
  - SBOM and VEX should be more widely adopted in open-source projects
  - IoTcube Hatbom aims to help by integrating and automating both in a unified platform
- Four technical challenges complicate vulnerability management
  - More efforts are required including code-level analysis, not only for origin projects but also for **upstream** and **downstream projects**
  - Our work is still at an early stage, and we are excited to collaborate on shaping next steps

Contact

Center for Software Security & Assurance (CSSA), Korea University

Center: https://cssa.korea.edu | cssa@korea.ac.kr

- ✓ For IoTcube Hatbom, visit https://iotcube.net
- ✓ For students, visit CCS Lab https://ccs.korea.ac.kr
- ✓ For enterprise solutions, visit Labrador Labs https://labradorlabs.ai