

CSSA(Center for Software Security and Assurance)

Hatbom User Guide

【Step I : SBOM, Step II : Vulnerability, StepIII: VEX, StepIV: Result】

Center for Software Security and
Assurance(CSSA), Korea University
2025-8-26

1. Introduction of Hatbom

Hatbom is a “Security Vulnerability Automated Analysis Platform” developed by the Center for Software Security and Assurance (CSSA) at Korea University.

1-1. IoTcube 1.0

IoTcube 1.0 is the initial version of the “security vulnerability automated analysis platform” service, publicly released on April 19, 2016.

- A tool that identifies and analyzes security vulnerabilities embedded in software.
- Designed so that anyone — not only security experts — can use it easily via drag-and-drop of source files.
- Provides utilities to manage the security of software deployed on a variety of devices, including IoT devices.

1-2. IoTcube 2.0

IoTcube 2.0 represents an evolution from version 1.0 and was released under the name “Hatbom” on August 26, 2025.

- Reorganized and enhanced analysis features with a focus on SBOM and VEX.
- One-stop, user-friendly process that requires only the software input to start analysis.
- Visualizations of components and dependencies that reflect the strengths of the Korea University research team.
- Optimized database and server performance for faster and more convenient analysis.

2. Key Functions of Hatbom

Hatbom is built around three main functions: automated SBOM generation, vulnerability detection, and generation of VEX documents that can be used as guidance for managing detected vulnerabilities.

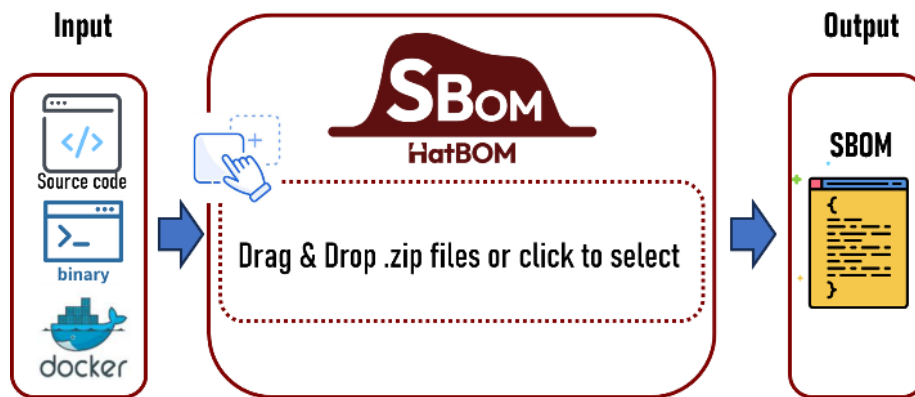
Supported languages for each function are listed below and will be expanded over time.

- Supported languages (to be expanded)

	① SBOM Step		② Vulnerability Step	③ VEX Step
	SBOM	Dependency Graph	CVE vulnerabilities	Static Analysis
Support Language	C/C++, Java, Python, GO, PHP	C/C++	C/C++, Java, Python	C/C++

2.1 SBOM Generation Step

The software can automatically generate an SBOM document when software is uploaded.



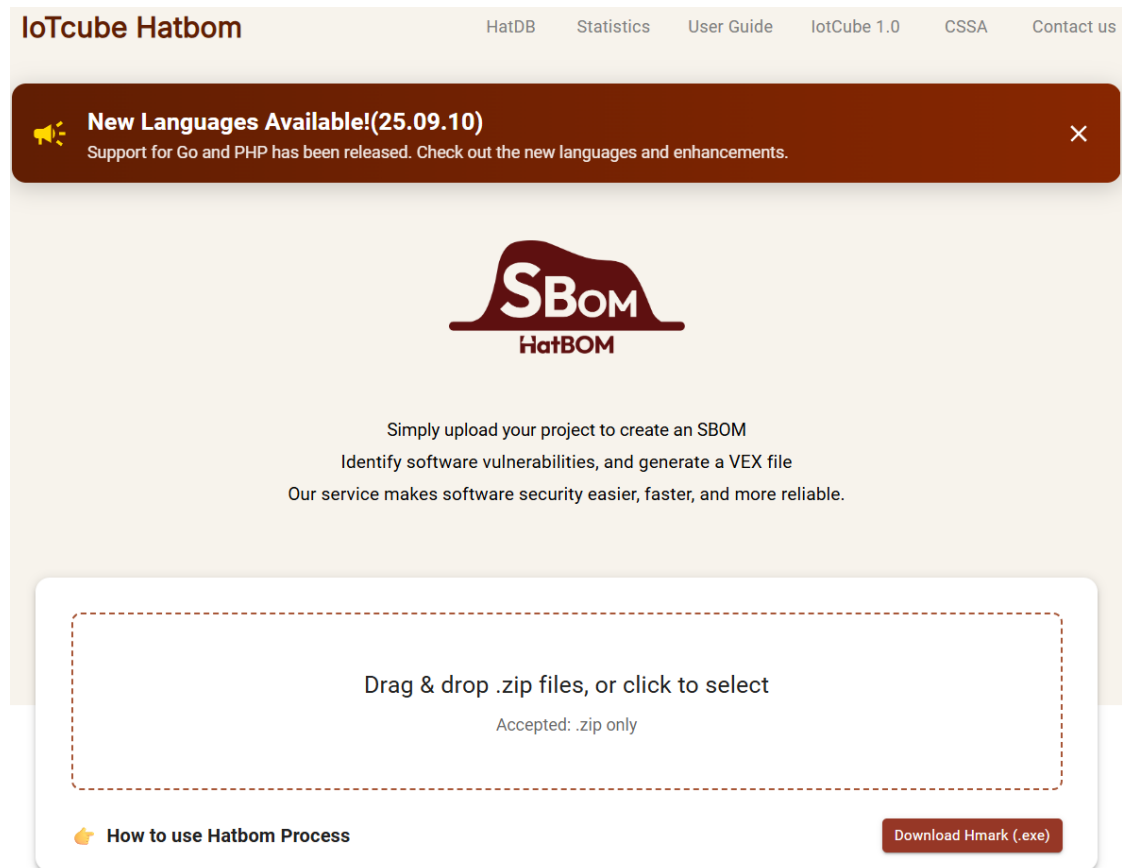
In the current version, only source code is accepted as input; support will be gradually extended to binaries and Docker images.

- Supported languages: C/C++, Java, Python, Go, PHP (more languages to be added).

※ To preserve software privacy, Hatbom also provides a local hashing(hashing-on-client) tool that hashes the software locally and uploads the hashed package to the platform (Hmark program).

[Procedure of analysis]

Go to : <https://iotcube.net> 을 입력



- 1) Compress the project folder that contains the project code into a .zip file and Drag & Drop it (①, ②, ③).



2) When analysis is complete, the following results are displayed.

Source Code

SBOM
VULN
VEX

This is the Source Code Upload Type SBOM/Vulnerability List/VEX Generation Page.
Our Core Engines are highly optimized and efficient in generating SBOM, Vulnerability Lists, and VEX Generation Pages from source code.

1 SBOM
 2 Vulnerability
 3 VEX
 4 Result

SBOM

Next

14 components identified in target software.

binutils-2.34

- 2 subhook v0.4.2
- 1 turtlebot_apps rocon_demo_201911
- 1 node-mapnik 0.7.9
- 14 ObEngine v0.4.2
- 2 build prerewrite-develop
- 1 cronie cronie-1.6.1
- 11 glibc glibc-2.26.9000
- 9 gnupg V1-9-9
- 5 dyld dyld-97.1
- 1 vpn wmhtgui-sensitivity
- 6 zlib-ng v1.2.9
- 5 workrave v1_8_3
- 1 platform_system_core android-wear-n-preview-1
- 1565 binutils-gdb users@@@hjl@@@linux@@@released

4
Dependency Graph

Result Details

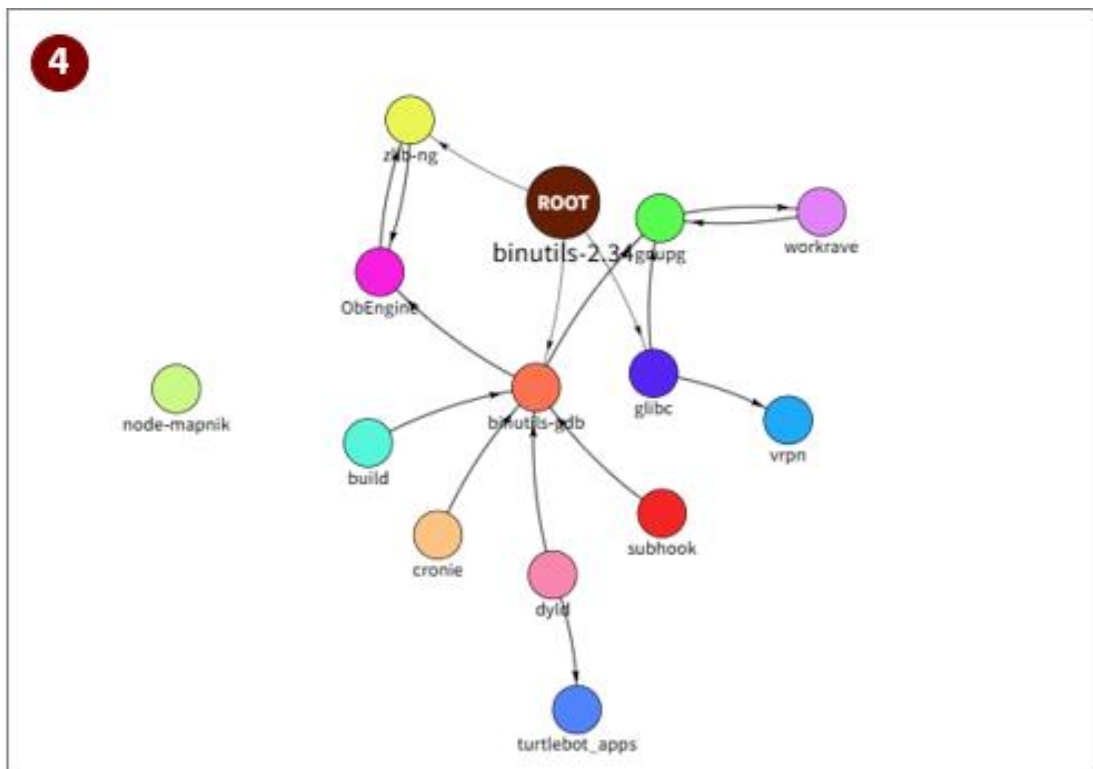
File Name	binutils-2.34
Files	1624
Dependencies	14
Input Format	ZIP File
Output Format	CycloneDX format SBOM

5
SBOM Download

- The binutils-2.34 project package contains 14 open- source components.
- Clicking each of the 14 component nodes shows the list of reused files.
- Result Details :

- File Names binutils-2.34
- Files 1624
- Dependencies 14
- Input Format ZIP File
- Output Format CycloneDX format SBOM
- Click the Dependency Graph(④) to view the detected dependency relationships among components.

④ Visualization graph of **dependencies** between detected components



⑤ Generated SBOM(CycloneDX)

5 SBOM(CycloneDX)

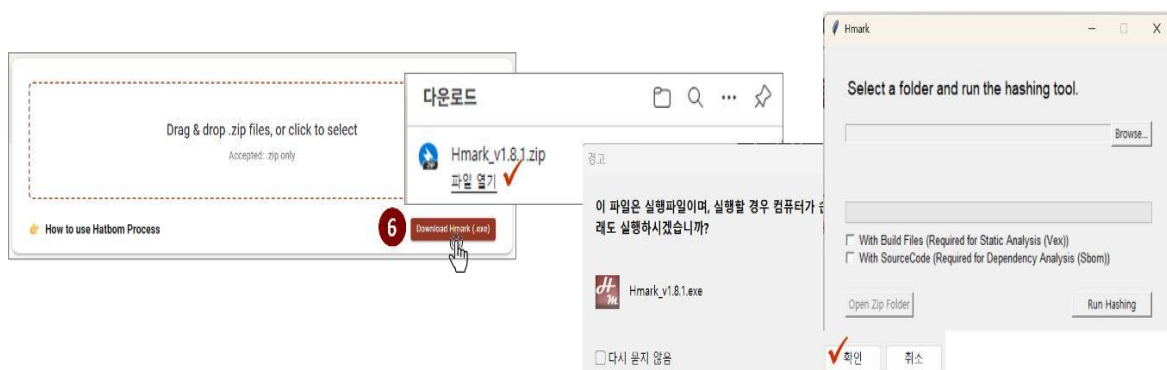
```
{
  "sbom": {
    "bomFormat": "CycloneDX",
    "specVersion": "1.4",
    "serialNumber": "urn:uuid:fca4f21b-9660-8b0b-0b47-fe0d5ea9e126",
    "version": 1,
    "metadata": {
      "timestamp": "2025-09-26T05:52:42.627685+00:00",
      "authors": [
        {
          "name": "IoTcube - https://iotcube.net"
        }
      ],
      "component": {
        "group": "",
        "name": "binutils-2.34",
        "version": "",
        "type": "application",
        "bom-ref": "pkg:generic/binutils-2.34",
        "purl": "pkg:generic/binutils-2.34"
      }
    },
    "dependencies": [
      {
        "ref": "binutils-2.34",
        "dependsOn": [
          "subhook v0.4.2",
          "turtlebot_apps rocon_demo_201311",
          "node-mapnik 0.7.9",
          "ObEngine v0.4.2",
          "build prerewrite-develop",
          "cronie cronie-1.6.1",
          "glibc glibc-2.36.9000",
          "gnupg V1-9-9",
          "dyld dyld-97.1",
          "vrpn wmhtgui-sensitivity",
          "zlib-ng v1.2.9",

```

3) Hashing the project file for **code privacy**

Instead of uploading the project files as-is, you can download the hashing program, perform hashing locally, and upload the hashed file.

- Download the Hmark tool below and run hashing locally.



Hmark

Select a folder and run the hashing tool.

Browse...

☐ With Build Files (Required for Static Analysis (Vex))
 ☐ With SourceCode (Required for Dependency Analysis (Sbom))

Open Zip Folder

Run Hashing

☐ **With Build Files** : If click, binary files required for static analysis are included(for C/C++ projects that have completed the build).

☐ **With SourceCode** : If click, the source code required to build the dependency graph is included.

- Drag & Drop the .zip file produced by hashing.



- After this, the process is the same as the previous order.

Notice! :

Differences when analyzing with source code files vs. hashed files.

Process	SBOM		Vulnerability	VEX	Remark
	SBOM	Dependency Graph	Vulnerability	Static Analysis	
Source Code	○	○	○	X	VEX document editing is always provided regardless of static analysis.
Hmark	○	X	○	(When a binary build is available) ○	

2.2 Vulnerability Step

This stage is the vulnerability detection step. You can see which vulnerabilities are present in the submitted project. Click ⑨ or ⑩ as shown below to run vulnerability detection.

Source Code

NotepadVULDBVEX

This is the Source Code Upload Type SBOM/Vulnerability List/VEX Generation Page.
Our Core Engines are highly optimized and efficient in generating SBOM, Vulnerability Lists, and VEX Generation Pages from source code.

1 SBOM9 Vulnerability3 VEXResult

SBOM

14 components identified in target software.

binutils-2.34root

2

subhook v0.4.2

1

turtlebot_apps rocon_demo_201811

1

node-mapnik 0.7.9

14

ObEngine v0.4.2

2

build prewrite develop

1

crone crone-1.6.1

11

glibc glibc-2.36.9000

9

grupeg V1-0-0

5

dyld dyld-97.1

1

vgn vmhgul-sensitivity

6

zlib-ng v1.2.0

5

workrave v1_8_3

1

platform_system_core android-wearn preview-1

1565

binutils-gdb users @@@@linux@@@released

Dependency Graph

Result Details

File Name	binutils-2.34
Files	1624
Dependencies	14
Input Format	ZIP File
Output Format	CycloneDX format SBOM

SBOM Download

Vulnerability

Next

Detected 45 vulnerable code clones (2 kinds of CVE) in your package.

#Detected vulnerable code clones
45

#Detected unique CVEs
2

Rank of Top 3 Vulnerable Files

Rank	Name	Count
1	binutils/arlex.c	9
2	binutils/deflex.c	9
3	binutils/syslex.c	9

Rank of Top 3 CVE

Rank	Name	Count
1	CVE-2019-16866	25
2	CVE-2019-18934	20

UDDY Vulnerable Files

id	File Path	CVE	CVSS ▲	KEY 🔑	<input type="checkbox"/>
1	binutils/deflex.c	CVE-2019-18934	Medium	None	<input type="checkbox"/>
2	binutils/arlex.c	CVE-2019-16866	Medium	None	<input type="checkbox"/>
3	binutils/arlex.c	CVE-2019-18934	Medium	None	<input type="checkbox"/>
4	gas/itbl-lex.c	CVE-2019-16866	Medium	None	<input type="checkbox"/>
5	binutils/arlex.c	CVE-2019-16866	Medium	None	<input type="checkbox"/>
6	binutils/syslex.c	CVE-2019-18934	Medium	None	<input type="checkbox"/>
7	gas/itbl-lex.c	CVE-2019-16866	Medium	None	<input type="checkbox"/>
8	binutils/syslex.c	CVE-2019-18934	Medium	None	<input type="checkbox"/>
9	binutils/syslex.c	CVE-2019-16866	Medium	None	<input type="checkbox"/>
10	binutils/deflex.c	CVE-2019-16866	Medium	None	<input type="checkbox"/>

Description
<ol style="list-style-type: none"> 1) Vulnerabilities were detected in a total of 45 files. 2) The detected vulnerabilities are two types: CVE-2019-16866 and CVE-2019-18934. 3) The top 3 files containing the most occurrences of the above two CVEs are arlex.c, deflex.c, and syslex.c. 4) CVSS (Common Vulnerability Scoring System) rates vulnerability severity on a scale from 0 to 10: <ul style="list-style-type: none"> -None (0) -Low (0.1-3.9) -Medium (4.0-6.9) -High (7.0-8.9) -Critical (9.0-10.0) 5) To generate a VEX document, you must select the detected CVEs to include.

2.3 VEX Step

Developers can analyze and validate the selected CVEs themselves. For C/C++ projects, if the project has been built, static analysis is performed automatically and the results are provided together.

Source Code

Medium
Vulner
VEX

This is the Source Code Upload Type SBOM/Vulnerability List/VEX Generation Page.
Our Core Engines are highly optimized and efficient in generating SBOM, Vulnerability Lists, and VEX Generation Pages from source code.

1 SBOM
2 Vulnerability
11
3 VEX
4 Result

The number of functions to be analyzed exceeds 10,000. If you require a more extensive analysis, please contact us or reach out to the CBSA office.

Vulnerability

Detected 45 vulnerable code clones (2 kinds of CVE) in your package.

#Detected vulnerable code clones

45

#Detected unique CVEs

2

Rank of Top 3 Vulnerable Files

Rank	Name	Count
1	binutils/arexx.c	9
2	binutils/deflex.c	9
3	binutils/syslex.c	9

Rank of Top 3 CVE

Rank	Name	Count
1	CVE-2019-16866	25
2	CVE-2019-18934	20

UDDY Vulnerable Files

Id	File Path	CVE	CVSS	KEY	
1	binutils/deflex.c	CVE-2019-18934	Medium	None	<input checked="" type="checkbox"/>
2	binutils/arexx.c	CVE-2019-16866	Medium	None	<input checked="" type="checkbox"/>
3	binutils/arexx.c	CVE-2019-18934	Medium	None	<input checked="" type="checkbox"/>
4	gas/ftbHex.c	CVE-2019-16866	Medium	None	<input checked="" type="checkbox"/>
5	binutils/arexx.c	CVE-2019-16866	Medium	None	<input checked="" type="checkbox"/>
6	binutils/syslex.c	CVE-2019-18934	Medium	None	<input checked="" type="checkbox"/>
7	gas/ftbHex.c	CVE-2019-16866	Medium	None	<input checked="" type="checkbox"/>
8	binutils/syslex.c	CVE-2019-18934	Medium	None	<input checked="" type="checkbox"/>
9	binutils/syslex.c	CVE-2019-16866	Medium	None	<input checked="" type="checkbox"/>
10	binutils/deflex.c	CVE-2019-16866	Medium	None	<input checked="" type="checkbox"/>

- 1) Before clicking ⑪ or ⑫ to enter the VEX Step, developers refer to the information(⑬) of the CVEs detected in the vulnerability Step and select the CVEs(⑭) to be transferred to VEX..
- 2) Typically, CVEs with Medium or higher CVSS scores (Medium, High, Critical) are prioritized for analysis.

VEX

[Next](#)

Here is your VEX report based on the selected packages and identified vulnerabilities.

VEX Document Preview

15☒ Result Lock

```
1 {
2   "@context": "https://openvex.dev/ns/v0.2.0",
3   "id": "https://openvex.dev/docs/example/vex-fb18c08f-f212-4e07-b2e6-b4e177e21403",
4   "author": "matbom",
5   "role": "Document Creator",
6   "timestamp": "2025-09-26T00:42:02.478Z",
7   "version": 1,
8   "statements": [
9     {
10      "vulnerability": {
11        "name": "CVE-2019-16866"
12      },
13      "products": [
14        {
15          "id": "binutils-2.34@v0.1.null - yyensura_buffer_stack"
16        }
17      ],
18      "status": "affected",
19      "justification": "-",
20      "impact_statement": "-"
21    },
22    {
23      "vulnerability": {
24        "name": "CVE-2019-16866"
25      },
26      "products": [
27        {
28          "id": "binutils-2.34@v0.1.null - yyensura_buffer_stack"
29        }
30      ],
31      "status": "affected",
32      "justification": "-",
33      "impact_statement": "-"
34    }
35  ]
36 }
```

190

Total vulnerabilities (vulnerable code clones)

55

Affected

90

Not Affected

-
















Fixed

45

Under Investigation

Detected CVE List

16

Index	CVE	Products	Status	Actions
1	CVE-2019-16866	binutils-2.34@v0.1.null - yyensura_buffer_stack	affected	  
2	CVE-2019-16866	binutils-2.34@v0.1.null - yyensura_buffer_stack	affected	  
3	CVE-2019-16866	binutils-2.34@v0.1.null - yyensura_buffer_stack	affected	  
4	CVE-2019-16866	binutils-2.34@v0.1.null - yyensura_buffer_stack	affected	  
5	CVE-2019-16866	binutils-2.34@v0.1.null - yyensura_buffer_stack	affected	  

[+ New CVE Document](#)[CVE Download All](#)[Recover CVE List](#)

Rows per page: 5 1-5 of 190

VEX file download

17

OpenVEX  **Recommended**

Structured VEX Format



VDR 

Vulnerability Data Repository



CSAF 

Security Advisory Framework



CycloneDX 

SBOM Standard Format

Description

```

15 {
16   "vulnerability": {
17     "name": "CVE-2019-16866"
18   },
19   "products": [
20     {
21       "@id": "binutils-2.34@v0.1null - yyensure_buffer_stack"
22     }
23   ],
24   "status": "affected",
25   "justification": "-",
26   "impact_statement": "-"
27 },
28 {
29   "vulnerability": {
30     "name": "CVE-2019-16866"
31   },
32   "products": [
33     {
34       "@id": "binutils-2.34@v0.1null - yyensure_buffer_stack"
35     }
36   ],
37   "status": "affected",
38   "justification": "-",
39   "impact_statement": "-"
40 }

```

⑮ The detected-vulnerabilities list includes status fields for the CVEs that have been selected.

After developers analyze and validate each selected vulnerability, they set the final status value for each CVE.

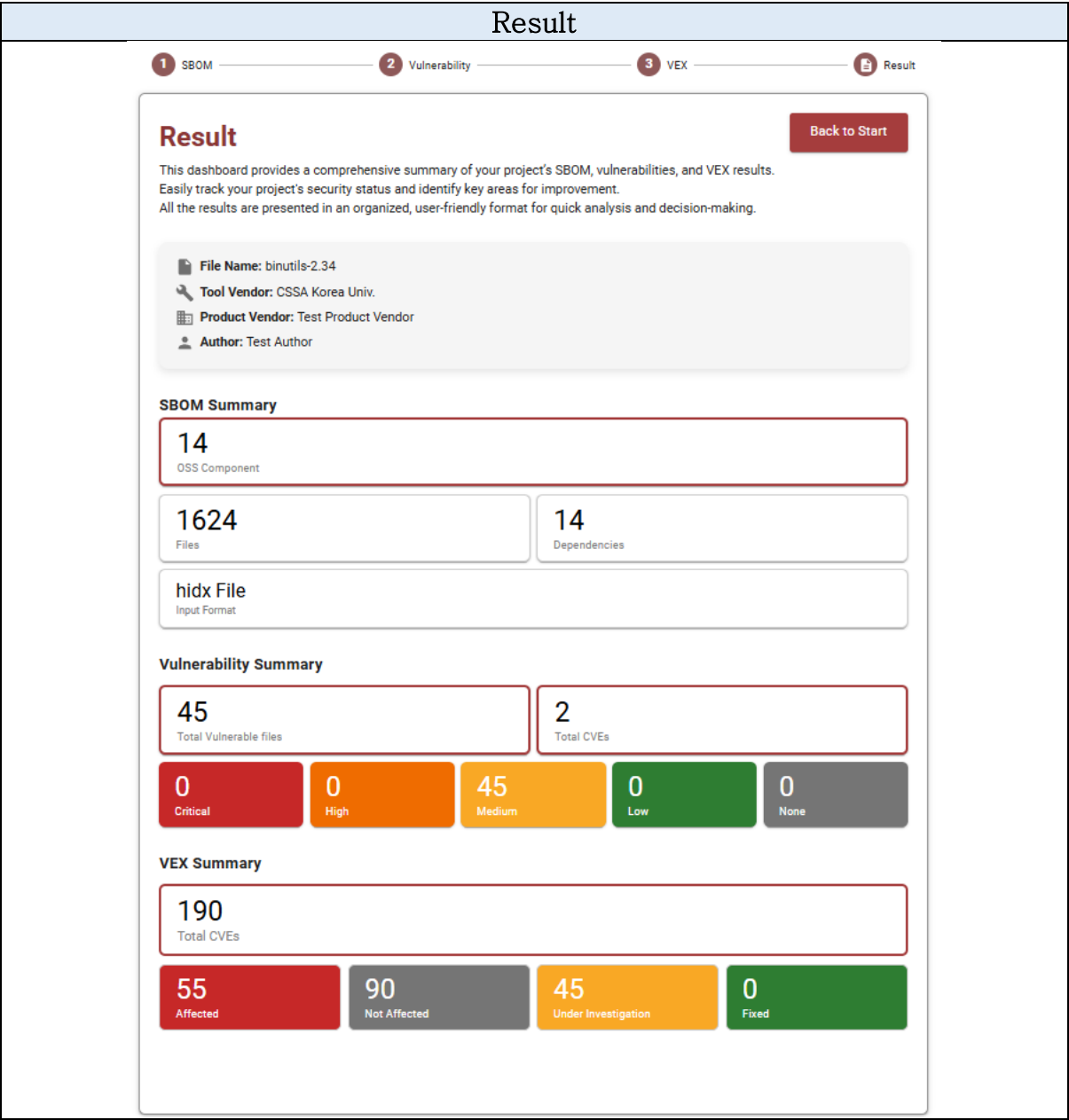
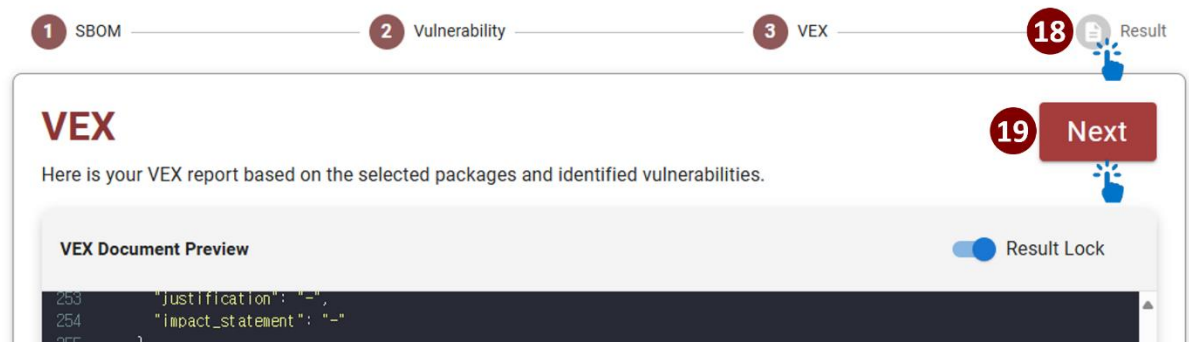
If you decide that it is necessary to change the status values for each CVE, you can click the Modify icon (⑮) for the CVE and select from the four status lists as shown on the left to modify it.

⑰ Once all CVE status values have been adjusted, the VEX document can be downloaded.
※Currently, OpenVEX format is supported first.

2.4 Result Step

This feature provides a summary of the analysis performed so far.

Click ⑱ or ⑲ to proceed to the Result step.



3. Future Plan

- Add language support to Hatbom: TypeScript and Ruby (by the end of 2025)
- Release a binary-based SBOM generation and vulnerability detection tool
- Release a container-image-based SBOM generation and vulnerability detection tool

※ Contact: cssa@korea.ac.kr