CS 460/ECE 419 Fall 2025
Module 8: Lab Assignment: Inside the Stack: Mapping Hardware and Software Bill of Materials

HBOM vs. SBOM Explained

What's the difference? A **Software Bill of Materials (SBOM)** lists everything inside the code: The open-source libraries, packages, and dependencies that make up software components. A **Hardware Bill of Materials (HBOM)** lists everything on the board: The chips, sensors, microcontrollers, and modules that make up the physical device. Together, they provide end-to-end visibility across the digital and physical supply chain, which is essential for identifying vulnerabilities, counterfeit parts, and hidden dependencies.

Aspect	SBOM	HBOM
Purpose	Tracks software components, versions, and known CVEs.	Tracks hardware components, part numbers, and suppliers.
Focus	Software supply-chain risk (vulnerable libraries).	Hardware provenance and tampering risk (counterfeit or insecure chips).
Contents	Package names, versions, dependencies, licenses, CVEs.	Chipsets, sensors, SoCs, communication modules, firmware IDs.
Common Formats	SPDX, CycloneDX, Syft JSON.	IPC-1752A, IPC-1754, or vendor-specific CSV/XML formats.
Key Users	Developers, vulnerability managers, software assurance teams.	OEMs, supply-chain analysts, hardware security engineers.
Tagline	*"What's in the code."*	*"What's on the board."*

Assignment Objective

Students will learn to generate, analyze, and interpret both SBOM and HBOM artifacts. By the end of this lab, students should be able to:

1. Generate an SBOM from a software repository using Syft or Trivy.
2. Perform a vulnerability analysis using Grype.
3. Interpret and document CVE findings in the context of a specific hardware device (e.g., M5Stack Cardputer, M5Stamp Fly, M5StampPLC)
4. Compare SBOM results across tools and formats (SPDX vs CycloneDX).
5. Identify potential HBOM components and describe how hardware provenance affects software risk.

CS 460/ECE 419 Fall 2025
Module 8: Lab Assignment: Inside the Stack: Mapping Hardware and Software Bill of Materials

Report Deliverable

Throughout this lab, you will capture your commands, screenshots, and findings in a short report (2–3 pages total). This report will compile your SBOM generation results, vulnerability analysis (top 5 CVEs), and HBOM component summary. The final report, along with the required output files, will be submitted in your Codespace's deliverables/folder and pushed to GitHub.

Firmware Targets

1. M55Stack Official Cardupter with M5stampS3 v1.1 Development Kit: https://shop.m5stack.com/products/m5stack-cardputer-kit-w-m5stamps3?srsltid=AfmBOoqYlj6xwdijrXebQvER4xcF4CFlfx0xOF53qY1oaUlZ7jcJa945
2. Mini-Drone: M5Stamp Fly with M5StampS3 and Remote control for Drone: https://docs.m5stack.com/en/app/Stamp%20Fly
3. M5StamPLC (industrial-style I/O/PLC-style controller): https://shop.m5stack.com/products/m5stamp-plc-controller-with-m5stamps3

Resources

- Idaho National Lab (INL) SBOM: https://sbom.inl.gov
- National Vulnerability Database (NVD): https://nvd.nist.gov/vuln/search

Part 1: SBOM Generation (in Codespaces)

All commands in this assignment can be executed inside your GitHub Codespace. No local installations are required.

1. In your Codespace terminal, clone the firmware repo for your chosen device (choose **one**):

   ```
   git clone https://github.com/m5stack/M5Cardputer.git
   ```

   # or

   ```
   git clone https://github.com/m5stack/M5Stamp-Fly.git
   ```

   # or

   ```
   git clone https://github.com/m5stack/M5StamPLC.git
   ```

2. Change into that folder, for example:

   ```
   cd M5Cardputer
   ```

CS 460/ECE 419 Fall 2025
Module 8: Lab Assignment: Inside the Stack: Mapping Hardware and Software Bill of Materials

3. Generate an SBOM in SPDX format using Syft:

```
syft . -o spdx-json > ../deliverables/sbom_syft_spdx.json
```

4. Generate a CycloneDX SBOM using Trivy:

```
trivy fs . --format cyclonedx --output ../deliverables/sbom_trivy_cdx.json
```

5. Record:
   - How many components each tool reported (Syft vs. Trivy),
   - One difference you notice between the SPDX SBOM and the CycloneDX SBOM (format, fields, component count, etc.), and
   - Screenshots of your terminal output.

After both commands complete, run:

```
ls ../deliverables/
```

to confirm your SBOM files were created (sbom_syft_spdx.json and sbom_trivy_cdx.json). In your report, note how many components each tool detected and describe one key difference you observed.

Part 2: SBOM Vulnerability Analysis

1. Feed the SBOM into Grype to discover CVEs (on one line):

```
grype sbom:../deliverables/sbom_syft_spdx.json -o table >
../deliverables/vuln_analysis_grype.txt
```

NOTE: If Grype reports zero vulnerability matches, that is still valid. Embedded firmware often doesn't declare traditional package versions, so automated scanners may not map any CVEs. You should still include this result and discuss why the scan might return zero findings.

2. In your report, include a table for the top 5 vulnerabilities that includes the following:

CVE	Severity	Component	Version	Comment
CVE-2023-0286	High	OpenSSL	3.0.2	TLS certificate validation bypass

CS 460/ECE 419 Fall 2025
Module 8: Lab Assignment: Inside the Stack: Mapping Hardware and Software Bill of Materials

3. To preview your results in the terminal before opening the full file, run:

   ```
   head -20 ../deliverables/vuln_analysis_grype.txt
   ```

   Copy the top 5 rows into your report table. Then select one CVE, locate it in the NVD Database, and summarize its cause or impact in one sentence.

Part 3: HBOM Exploration

Use official device documentation (links below) to identify three major hardware components (e.g., ESP32-S3 microcontroller, MPU6886 IMU, 2.0-inch LCD):

a. M5Cardputer Specs
b. M5Stamp Fly Docs
c. M5StamPLC Docs

1. For each component, find:
   a. Manufacturer (e.g., Espressif, Texas Instruments, Epson)
   b. Part number or chip family (e.g., ESP32-S3, INA226 current sensor, RX-8130 RTC)
   c. Known vulnerabilities or operational risk (look up the chip name in NVD, or describe functional/security impact - e.g., CAN bus tampering, wrong RTC timestamping)
2. Summarize how hardware dependencies can influence firmware/software risk (supply-chain discussion)

Part 4: Deliverables

1. Submit a brief report (2-3 pages).
2. Upload all required files to the deliverables/folder in your Codespace and commit/push to GitHub (use all lowercase filenames as shown below to simplify grading):
   a. `sbom_syft_spdx.json`
   b. `sbom_trivy_cdx.json`
   c. `vuln_analysis_grype.txt`
   d. `hbom_summary.md`
   e. `reflection.md`

CS 460/ECE 419 Fall 2025
Module 8: Lab Assignment: Inside the Stack: Mapping Hardware and Software Bill of Materials

Grading Rubric

Criterion	Excellent (Full Points)	Partial Credit	Points
SBOM Generation	Both SBOMs (Syft SPDX + Trivy CycloneDX) generated correctly for the chosen firmware repo; component count and format differences are discussed; screenshots included.	One file missing/no comparison.	10
Vulnerability Analysis	Grype scan is run against the Syft SBOM; either top 5 CVEs are summarized or (if 0 matches) the student explains why embedded firmware may show no CVE hits. One CVE or risk scenario is described using NVD or functional/security reasoning.	Minor inaccuracies/missing CVE table.	10
HBOM Exploration	Three components documented with manufacturer, part #, vulnerability info, and analysis.	Incomplete component data.	10
Report Quality	Clear organization, labeled screenshots, concise reflection linking software-hardware risk.	Formatting or clarity issues.	8
Professionalism	Proper file naming, citations, GitHub commit/push complete.	Minor submission errors.	2
TOTALS			**40**