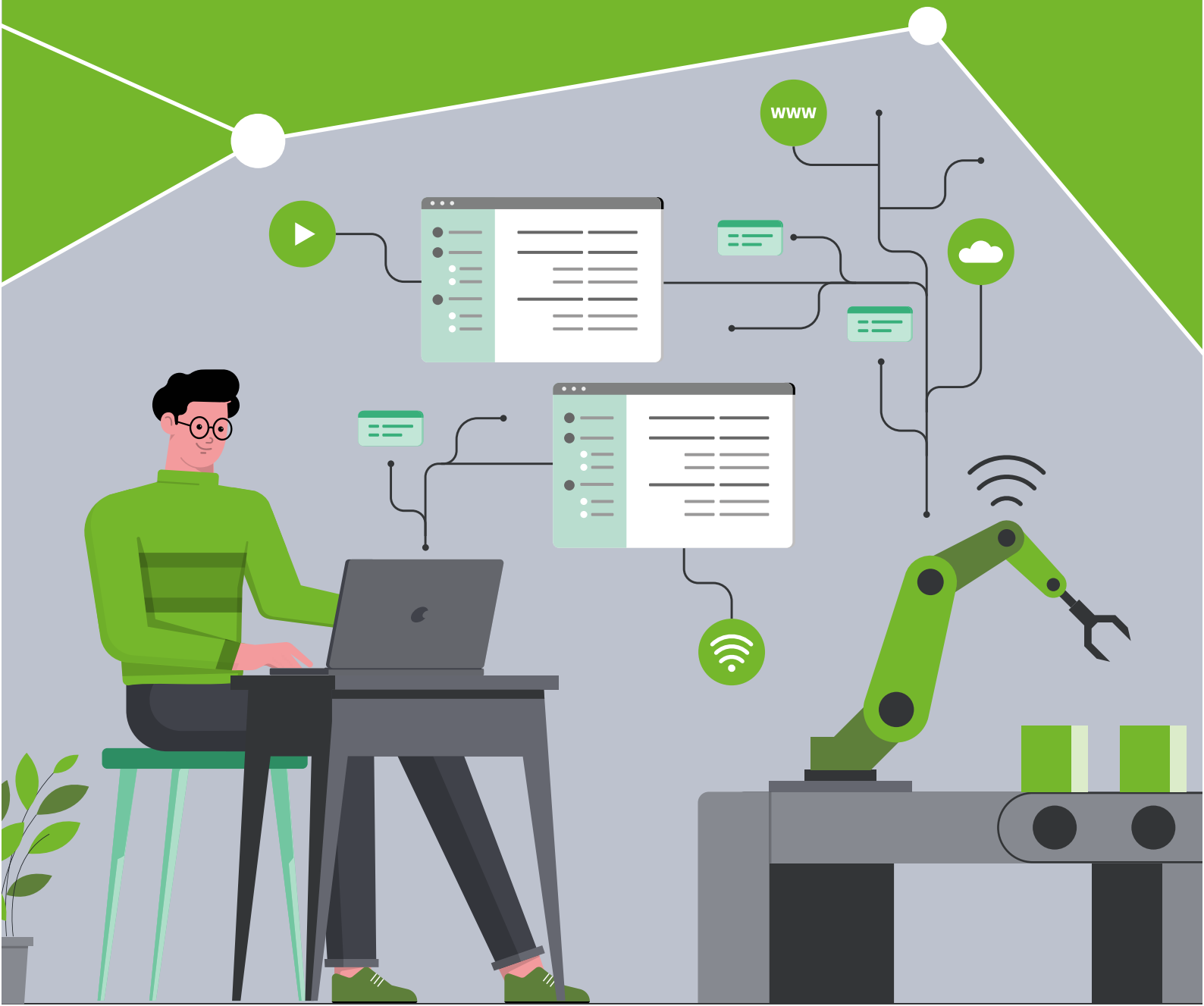
OPEN INDUSTRY 4.0 ALLIANCE

Case Study

Vulnerability Management Using Software Bill of Materials (SBoM)



IMPRINT

Publisher

Open Industry 4.0 Alliance

Christoph Merian-Ring 12, 4153 Reinach, Switzerland

https://openindustry4.com

info@openindustry4.com

Status

File: Vulnerability Management Using Software Bill of Materials (SBoM)

October 9th, 2024 – Version 1.0

Editors

Lucas Wolf (Open Industry 4.0 Alliance)

Authors

Roland Erk (Conplement AG)

Dr. Florian Probst (Software AG)

Daniel Bitzer (ifm electronic)

Vitaliy Volevach (Siemens)

Abstract

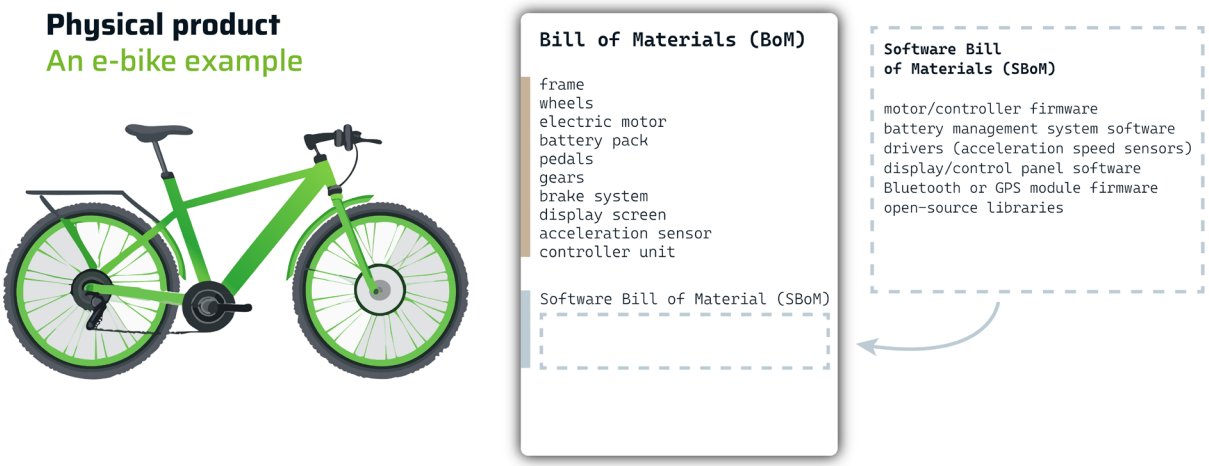
This case study explores the challenges and solutions related to managing software vulnerabilities in embedded devices and general-purpose software within industrial environments. It highlights the limitations of traditional methods and emphasizes the importance of adopting a Software Bill of Materials (SBoM) to enhance security and streamline the vulnerability management process. By examining both the "old world" approach, reminiscent of the log4j crisis, and the "new world" approach utilizing SBoMs and Asset Management Systems (AMS), the study illustrates how modern practices can lead to more effective, proactive security measures. Readers will gain insights into how manufacturers, integrators, and users can implement these strategies to improve their security posture, reduce maintenance costs, and ensure the reliability of their systems.

Motivation

Currently, security in embedded devices and general-purpose software is implemented in a very much ‘ad hoc’ manner, if at all. The fact that this isn't optimal is becoming apparent as cybercriminals score major hits in regular intervals, with headline news at least once a month, sometimes many more. This situation costs victims time, money, and customer confidence for some software vendors. However, since the impact is spread across multiple vendors, it doesn't create enough pressure to drive significant change. If it did, we'd see gradual improvements in software security instead of the current trend.

Since this problem isn't being resolved in an unregulated market, the EU and its member states are pursuing legislation to enhance security for embedded devices and general-purpose software. The goal is to empower users with the information needed to assess product security and hold vendors accountable for weak security measures and lack of timely updates to address vulnerabilities. While industrial devices are not currently subject to these regulatory requirements, manufacturers would be wise to consider implementing the same measures. This would future proof their processes in case the exemptions are removed. More importantly, it can already grant a competitive edge by demonstrating that their product design prioritizes security.

This case study focuses on a specific aspect of a product's security lifecycle: the Software Bill of Materials (SBoM). The SBoM is similar to the bill of materials that is well known for physical products and contains a list of all hardware components that are included in the product itself. Now, let's make this more tangible by looking at a real-world example.



For instance, the BoM for an e-bike would include the frame, wheels, electric motor, battery pack, pedals, gears, and so on. The SBoM for the e-bike will likewise contain a list of all

software components that are included in the device and is part of the BoM. For example, it includes the battery management system software, drivers for the acceleration and speed sensors, Bluetooth or GPS module firmware and any open-source libraries used for connectivity or sensor data processing. This list can be very small, like for an embedded sensor that only contains its manufacturers firmware, but devices with more complex functionality will likely contain hundreds of components, especially if they are based on the Linux or Windows operating systems. Most of these components are likely to be not written by the device manufacturer but are either off-the-shelf or open-source components.

To explain why this SBoM is important, let us look back to the log4j crisis. At that time, a lot of software that was developed using Java programming language was using an open-source component, the log4j library, for writing output. This library had a flaw that let an attacker run any software by altering its inputs. This fault made millions of software installations and devices vulnerable to attacks. The main problem was, when the error was discovered, the users of these software or devices did not know that they were vulnerable and therefore could not protect themselves, because they did not know that log4j was running inside their software. Manufacturers were also slow to announce that their products did include log4j before a fix was available for their software. This led to some installations still being vulnerable months after the fault was widely known.

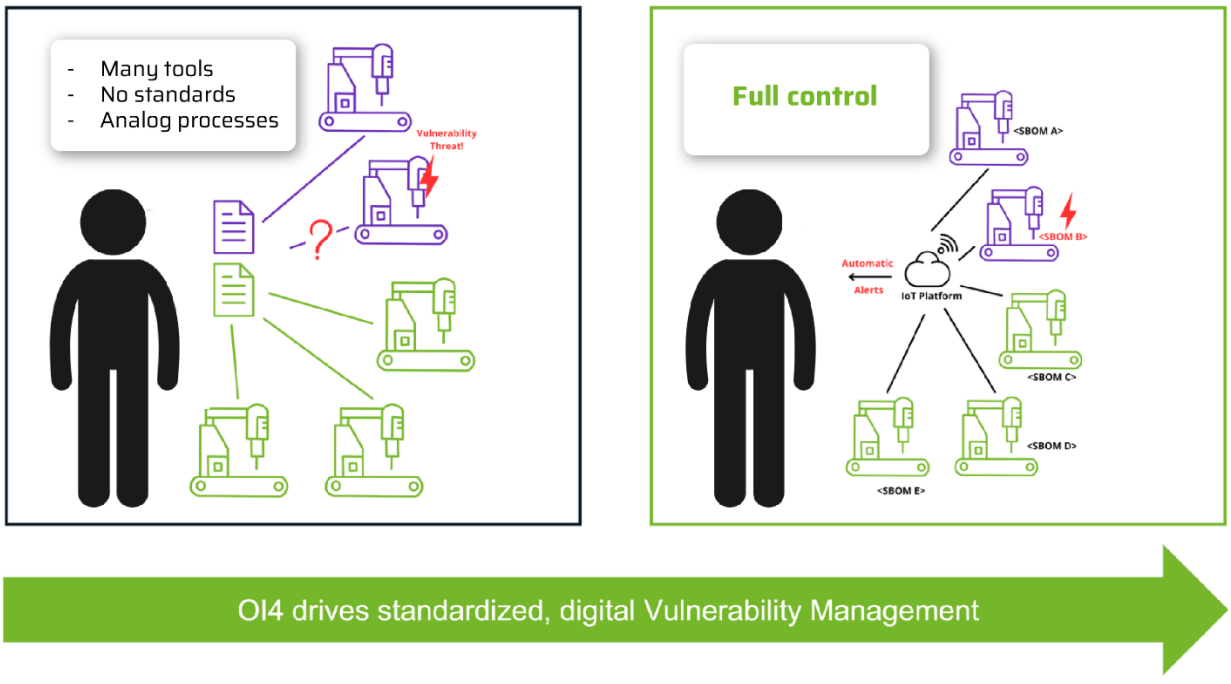
This incident demonstrated clearly that information about software vulnerabilities is currently not available to most end users and therefore the users are not able to determine what to do to protect themselves from attacks. To solve this, regulators are demanding that software and device manufacturers and integrators provide a SBoM to the user. This has multiple advantages:

- The users of a device can perform their own security analysis when vulnerabilities in the device software become known, taking their own scope of use and how this affects their security posture into consideration. This can be done immediately when knowledge of the vulnerability is released, as opposed to waiting for a manufacturer reaction that may be days or sometimes even months late.

- The manufacturers of an industrial device like a sensor or a motor controller usually do not know the device's end users, so contacting them in case of a significant vulnerability would require collecting contact information for all users, details an integrator or reseller of a product not necessarily wants to provide. Additionally, simply publishing the vulnerability on the manufacturer's website will leave a wide gap open for exploitation of the vulnerability and places an undue burden on the user to manually browse this information and verify whether the vulnerability applies or not.

- The manufacturers of machines typically utilize a multitude of industrial devices from different vendors. Now they can automatically notify the end-users about potential security vulnerabilities, based on the information received from the device manufacturers.

As often seen in the media, creating flawless hardware and software from the outset is an extremely challenging, if not unattainable goal. Despite this, users naturally expect software to be secure and reliable. While such expectations are legitimate, they are also somewhat idealistic, considering that delivering software that is proven to be robust and fault-resilient is associated with a significant increase in cost. Costs, that consumers are often reluctant to shoulder. A more pragmatic approach would be to recognize that some imperfections will occur and to implement strong procedures to address and rectify these faults efficiently as they come to light. This case study aims to demonstrate how this can be effectively accomplished by device manufacturers, integrators, and users, with the discussion being limited to devices utilized within a factory setting.



Manufacturer

Building and Updating Devices

The device manufacturer will have to generate a SBoM and provide it with each software update for all parts of the software a device includes.

For this case study, let's assume the device runs embedded Linux built by the Yocto build environment, a well-established environment by the Linux Foundation. Now, the device runs one or many applications in the form of docker containers. Base firmware and applications can update on their own schedule, so it makes sense for the manufacturer to provide SBoMs for each component separately.

In particular, the SBoM of the firmware can be generated automatically from the Yocto build by including the meta-dependencytrack layer into the build configuration. This will generate a CycloneDX format SBoM that is suitable for processing with a vulnerability analysis software or service like the open-source dependency trackservice (https://dependencytrack.org/). For the applications, the container images can be analyzed using the open source syft tool (https://github.com/anchore/syft/blob/main/README.md), which also generates CycloneDX format SBoMs.

Obviously, this would be done by the manufacturer in their continuous integration and continuous delivery (CI/CD) environment, and an updated SBoM would be accompanying each software release that gets built. For these SBoMs to be effective, they must reach the device's integrator or end user. While it is possible for the manufacturer to provide the SBoM on a physical medium or as a web link, this does have drawbacks:

- Manufacturers would need to know who to send the medium or link to. Contacts will change frequently and keeping contact information current is a burden.

- When the devices are integrated into bigger systems and sold to end-users down the chain, the contacts of the end-users are typically not known to the device manufacturer.

These drawbacks can be avoided by publishing the SBoM as part of the Asset Administration Shell (AAS) for this device. For each device provisioned with a combination of firmware and applications, the manufacturer would create a Type or Instance AAS for this device, including the SBoM for each component. A Type AAS describes a family of products with the same configuration, while an Instance AAS has a one-to-one link to one physical device. Which of those options or even both together is chosen does not matter, if

each available software component version has an entry in the AAS. Optimally, this would also include a link to the respective software update file.

The manufacturer publishes the AAS for the device on its public AAS repository and provides a way to find this AAS, for example by attaching a QR code to the device. This is also a good opportunity to also include additional documentation into the AAS, like user manuals, data sheets, maintenance instructions or certificates of compliance.

Handling Vulnerabilities

If a vulnerability becomes known in one of the components, the manufacturer provides themselves (as opposed to open source or third-party software dependencies), it is their responsibility to publish this information. While the first impulse may be to only reveal the vulnerability and advise the users directly or simply publish the vulnerability announcement on the manufacturer's website, it would be more prudent to release this information through the already established vulnerability reporting facilities for Common Vulnerabilities and Exposures (CVEs), so available vulnerability tracking systems can be used to detect the new vulnerability and alert the users. This eliminates the need for the manufacturer to maintain lists of former customers for notifications or for customers to frequently check the manufacturer's website for updates - both of which are manual, cumbersome, and prone to errors.

If, on the other hand, a vulnerability is revealed in one of the dependencies of the product, the responsibility of the manufacturer would be to assess the severity and consequences of the vulnerability and provide either a new firmware or application that fixes the problem if it is deemed to be exploitable. The new firmware will create a new version entry in the AAS instance of the device.

If a vulnerability is not exploitable, for example because it is in a container based image and the affected component is not used in a way that exposes it, the manufacturer can just update the SBoM and publish a new AAS version with this assessment result to potentially save the users time to do the analysis on their own and also demonstrate that the manufacturer reacts to security findings in a timely manner.

Integrator

The integrator will receive the Type and/or Instance AAS' for the devices used in the production of a machine and will include or reference these AAS' as components when delivering the machine with its own Instance AAS (just distributing a Type AAS seems less useful here).

If the integrator changes or updates the software on the device, they are responsible for updating the SBoM in the AAS to ensure it accurately reflects the software currently running on the machine. They also should establish a vulnerability handling process similar to the manufacturer above.

User

This case study assigns the primary responsibility for determining and enforcing the factory's security posture to the party most impacted by an attack - the user - who is therefore most motivated to seek improvements. The user is, in this industrial scenario, likely the IT department that is tasked with integrating a machine into the factory and providing an environment suitable for secure production and not the actual operator of the machine.

The IT department will likely be no stranger keeping a detailed inventory of deployed machines and software already, if only to facilitate maintenance and inspections. The following not only demonstrates how the IT department (in the following, just "the user") can attain a better security posture, but also automate most of the required tasks and therefore make net gains in ease and productivity.

Integrating a New Machine

When a new machine is procured, the user currently is already likely to receive a plethora of documentation with the machine. If the manufacturer has supplied these in the AAS with the device, the user gains access by just scanning the provided QR code. To make the most of this, the user will want to deploy an Asset Management System (AMS) that helps keep track of all the contents of the AAS. Adding the machine to the inventory then becomes as simple as scanning the QR code. If the manufacturer did not already provide an Instance AAS, this would be the time that one would be created by the AMS from the Type AAS supplied to keep track of further changes of the machine.

At this time, this AMS also extracts the SBoMs from the AAS and provides the list of components to a continuously running vulnerability tracking system like the open-source Dependency Track software, which can also easily be integrated into the AMS. Any already present vulnerabilities will be indicated to the user and can be evaluated before the machine is installed and operating, minimizing the window of attack. The user will maintain ongoing vulnerability tracking, continuously testing the SBoMs against known vulnerabilities throughout the machine's lifetime. They will also set up alerts for any new vulnerabilities identified by the system's data sources, which are updated with CVE releases from manufacturers or other entities that discover vulnerabilities.

In addition, the AMS will continuously monitor the AAS published by the manufacturer and integrator to detect changes in the SBoM (in case new information was added there) and merge some of these changes into the AAS stored locally. For example, as suggested above, if the manufacturer provides the assessment of the consequences of a vulnerability in a timely manner, the user can trust this information and save the time for the analysis. This is also an excellent channel for the manufacturer to indicate new software updates to the user.

Updating a Machine

Since the AMS already has a complete inventory of available updates, adding a scheduling option for updates is another time saving feature. The AMS can even aid the user in downloading all required updates for pre-screening and (for air-gapped environments) for these updates to be stored on a mass storage device to be carried to the machine to start the update. On a successful update, the AMS will update the local Instance AAS to indicate the new software version that was installed. If the machine is connectable by the AMS, the machine can even supply its own Instance AAS as an option, making even this step automatic.

To Summarize: a World With and Without SBoM

The following sections contrast how vulnerabilities were managed in the past with how they can be addressed today using approaches like the Software Bill of Materials (SBoM). This comparison highlights the shift towards more proactive and efficient methods of ensuring security and managing vulnerabilities in modern IT environments.

The Old World

Without the measures suggested above, this would follow along the lines of the log4j vulnerability. Let's assume a security researcher discovers it and follows responsible disclosure. Major Linux distributors would release updates and shortly after that, the vulnerability will be revealed openly. If the manufacturers are lucky, Yocto will have a fix at the same time as the Linux distributors, so they have a head start to build new software and may be able to supply an update before disclosure hits the media.

Assuming the manufacturer now wants to provide this update to all users of the device. Contacting the users is a time intensive process, and that does not guarantee a substantial number of users actually install the update in a timely manner given that contact information may be out of date or simply wrong and that end users may not understand the danger unless it is told in most dire words. We would very likely see a repeat of the log4j scenario, in that many installations were not secured against exploits even several months after it became known.

The New World (SBoM)

Once the affected device is acquired, the new owner's AMS will have gathered all the software modules on the device and submitted their corresponding SBoMs to the internal vulnerability tracking system. As the vulnerability becomes known, the owner's tracking systems will receive a preliminary CVE from their vulnerability information sources during the responsible disclosure process. This preliminary CVE should already indicate which component and version is vulnerable, although it should not contain specific information about the vulnerability that could lead to exploitation of it. If possible, the manufacturer can indicate in the CVE when a fix will become available.

From this moment on, the owner is already informed about the new vulnerability, knows which devices are affected, and can promptly conduct a risk analysis. The vulnerable devices can be firewalled, isolated or disconnected to prevent exploitation, depending on the security posture and risk tolerance of the owner. This decision can be made even before a software update to fix the vulnerability is available. If the release time is known, the

owner can already schedule a maintenance of the device to apply the fix as soon as possible and afterwards remove the temporary mitigations.

Vulnerability tracking systems are becoming a crucial part of modern IT infrastructure, assisting administrators in keeping up with the ever-growing number of reported vulnerabilities and the related software updates. Today, IT administrators have a company-wide dashboard of vulnerable software and can easily deploy fixes. Tomorrow, the same approach could apply to fleets of devices. An AMS could offer a vendor-agnostic, unified dashboard for vulnerable devices on the factory floor, regardless of the manufacturer, integrator, or location - whether within the company or elsewhere. This would enable responsible personnel to efficiently schedule and execute updates, reducing maintenance costs while simultaneously enhancing security.

Conclusion

In conclusion, the shift from traditional, reactive vulnerability management to a proactive approach using Software Bill of Materials (SBoM) represents a significant advancement in the security of embedded devices and general-purpose software. By adopting SBoMs and integrating them into Asset Management Systems (AMS), manufacturers, integrators, and users can better manage and mitigate security risks. This not only enhances the ability to respond swiftly to vulnerabilities but also improves overall security posture, reducing maintenance costs and increasing reliability. As the industry continues to evolve, embracing these modern practices will be crucial in safeguarding against the growing threats in today's digital landscape.

For this methodology to be effectively adopted, it is essential that different companies collaborate with each other. Consensus regarding the use of standards and common interfaces and trust in exchanging data must be reached. Consequently, organizations and alliances that foster the exchange between different players in the industrial field are crucial in increasing the adoption of technologies for vulnerability management, as discussed in this paper.