Using SBOMs to Secure Industrial IoT Devices

2022-07-27

Authors:

Isaac Dangana
Red Alert Labs
isaac.dangana@redalertlabs.com

Tom Alrich
Consultant to Red Alert Labs
tom.alrich@redalertlabs.com

CONTENTS

1 INTRODUCTION

Software bills of materials (SBOMs) are an important tool for ensuring the cybersecurity of Industrial Internet of Things (IIoT) and Internet of Things (IoT) devices. While much has been written about using SBOMs for user-managed software (the normal software use case), it is important to understand that SBOMs can also be used to describe the software and firmware installed in an IoT device, enabling better device security and maintenance.

In this article, the authors draw on their experience with both SBOMs and IoT/IIoT devices to identify the main ways in which an SBOM for an IoT device differs from an SBOM for a "user-managed" software product – i.e. what most of us think of when we think of "software". We describe how SBOMs for devices differ from those for software, and we explain the primary challenges facing SBOMs for devices. We close by describing how those challenges can be addressed.

2 SBOMS AND VEX DOCUMENTS

It is hardly an exaggeration to say that today's software products are built mostly out of *components*: code or compiled libraries that perform particular functions. Software would be much more expensive and take much longer to develop (if it were developed at all) if developers were not able to take advantage of components. The average software product includes over 100 components; 90% of those are open source.[1] Many software products contain thousands of components. Some are obtained as source code and others as binaries.

However, with more components, the risk of vulnerabilities from each component increases. The recently discovered Log4j vulnerabilities showed that a single library can be widely used, including in other libraries. It therefore is often buried under many "layers" of other components. This can make it literally impossible to discover every instance of that library. This also makes it impossible to patch each instance. Vulnerabilities like those in Log4j will probably be with us as long as we rely on computers.

An SBOM is at heart very simple: It is a machine-readable listing of the components in a software product or intelligent device (collectively referred to as a "product" in this article).[2]

Having an SBOM allows a software developer (supplier) to learn of risks posed by components included in one of their products, so they can take steps to mitigate those risks by for example replacing vulnerable components with less vulnerable ones. Having an SBOM also allows a

[1] Sonatype, "2020 State of the software supply chain report", *https://www.sonatype.com/resources/white-paper-state-of-the-software-supply-chain-2020*

[2] A good two-page discussion of SBOMs is available at *https://ntia.gov/files/ntia/publications/sbom_at_a_glance_apr2021.pdf*

software using organization to "look over the shoulder" of the supplier by identifying potentially risky components in a product, then having a conversation with the supplier about how they will mitigate those risks through patching or other means. Even if the supplier does not provide a patch for a vulnerability, in many cases there are other mitigations a user organization can apply to reduce the risk due to the vulnerability.

There are currently two mature, full-featured SBOM formats: CycloneDX[3] and SPDX[4]. Both can be represented in multiple forms, including spreadsheets. However, the two most widely used representations of both formats are JSON and XML.

There is another, even more recently developed, document type called VEX (an acronym for Vulnerability Exploitability eXchange), which is tied closely with the SBOM concept. VEX documents were developed to solve a specific problem that became apparent only as SBOMs became more widely produced and used for software vulnerability management purposes.

The problem is that a high percentage (perhaps over 90%) of vulnerabilities that are found in components of a software product are not in fact *exploitable* in the product itself. This means that an attacker will not be able to utilize that vulnerability to compromise the product. It also means that someone scanning an instance of the product for that vulnerability will not find it.

Of course, most security professionals will ask, "Why is it a 'problem' that a vulnerability isn't exploitable in a product? Isn't that a good thing?" While it is a good thing, there's also a problem: The software user currently has no way of knowing, when they find that a component of a product contains a serious vulnerability, whether or not the vulnerability is exploitable in the product itself. This means that software users may waste a lot of time scanning for and patching vulnerabilities that aren't exploitable.

Moreover, if users don't know which vulnerabilities aren't exploitable, they are likely to overwhelm the support lines of their suppliers with calls about non-exploitable vulnerabilities. This will increase the suppliers' support costs, as well as the level of frustration of support staff members. If nothing were done about this, it could ultimately result in SBOMs no longer being distributed or used.

There are many reasons why a component vulnerability might not be exploitable in the product itself. A common example is that, even though a developer included one or two modules of a library that contains four modules in their product, they did not implement the module that is subject to the vulnerability. Therefore, an attacker trying to exploit that vulnerability to compromise the product will never succeed.

[3] *https://cyclonedx.org/*

[4] *https://spdx.dev/*

VEX documents are designed to alleviate this problem. A VEX essentially states, in a machine-readable format, "Even though this vulnerability (e.g., CVE-2021-12345) has been identified in one of the components of our product A, it is not in fact exploitable in product A itself."[5]

There are currently (as of June 2022) two VEX formats, both machine-readable. One is based on the OASIS CSAF vulnerability notification format.[6] The other is based on the OWASP CycloneDX SBOM format.[7]

3 SBOMs for Devices vs. SBOMs for User-Managed Software

Today, two types of SBOMs are regularly developed, although they are not typically distinguished one from the other. One type is for *user-managed* software products – i.e., software that is intended to be installed on a standard hardware platform such as an Intel™ X86-type server[8] operated under the control of a standard operating system, such as Microsoft Windows™ or Linux. The decision about what user-managed software to install on a particular server or workstation and when it will be installed is almost always left up to the user organization. By the same token, the user organization is required to perform all maintenance on the software, including patching.

The other type of SBOM is for the software and firmware installed in a device, typically an IoT or IIoT device. A device differs from a user-managed software product, primarily because all software necessary for the operation of the device is pre-installed by the manufacturer. The device is usually provided as a sealed box that cannot be opened by the end user. Patches and other updates to the software are usually remotely applied by the supplier, often without being subject to any control by the user.

The manufacturer of a device often does not manufacture the hardware or develop the software included in the device. Instead, they put the pieces together to achieve the purpose of the device. While the software products installed on a device are usually developed, sold and supported by different suppliers, legally and practically the manufacturer of the device is solely responsible for choosing, installing, patching and updating those software products.

---

[5] The VEX document can be used to make almost any notification about vulnerabilities, not just a notification that a particular vulnerability is not exploitable in a product. However, unlike most "traditional" vulnerability notifications, the VEX is machine-readable and the data it contains can be incorporated into many automated processes.

[6] *https://docs.oasis-open.org/csaf/csaf/v2.0/csd01/csaf-v2.0-csd01.html*

[7] *https://cyclonedx.org/capabilities/vex/*. Note that a CycloneDX VEX can be used with, and refer to, either a CycloneDX or an SPDX SBOM.

[8] For the purposes of this discussion, it makes no difference whether the server is physical or virtual.

Of course, this means that a device typically utilizes many individual software products, developed by different organizations (including open source communities). How does the software running on the device differ from user-managed software? The biggest difference is that a device is built to fulfill a specific purpose; all of the software installed on the device is only there because it plays a role in fulfilling that purpose. Therefore, it makes sense to treat all the software and firmware installed on the device as a single unit.

On the other hand, user-managed software products each have their own purpose. The server on which they run is simply a platform; a given software product can usually be moved from one server to another without any loss of functionality. Moreover, many user-managed software products can be installed on a single physical or virtual server yet have no connection to each other besides the fact that they run under the same instance of the operating system. The choice of which user-managed software products run on a particular server is entirely up to the user organization.

Therefore, in the case of user-managed software, the device owner needs to receive an SBOM for every software product (including operating systems and firmware) that they utilize. An SBOM for the server itself (which would have to change whenever any of the software products on the server was updated, added, or removed) would be meaningless[9]. Such an SBOM would simply provide a list of the software products currently installed on the server, which could just as easily be obtained from Windows Explorer.

However, the opposite is true for an IoT or IIoT device: the only SBOM that makes sense is one for the entire device. The end user is not directly involved in the decision whether to install, update or remove any software product in the device; therefore, they cannot play a direct role in managing any risks posed by individual software products installed on the device. For versioning purposes, the manufacturer usually treats the entire set of software and firmware in the device as a single software product, even though it is often composed of many individual products that are also sold as user-managed software.

Regarding updates and patching, the device manufacturer normally groups updates and patches of multiple software products into a new release of the device software with a unique version number; usually, the new release is "pushed out" to all the devices in use. The manufacturer of the device does not usually push out to their customers every update or patch for an individual software product contained in the device as soon as they receive it from the product's supplier; rather, they combine all the individual product updates and patches received since the last device update into a new device update with its own version number.

[9] Note that the operating system of the device – whether an IoT device or an Intel-standard server running different types of software – is always treated as just one software product running on the device, for SBOM purposes.

The one exception to this rule is if a serious vulnerability has been identified in one of the software products included in the device, which requires an immediate patch; in such a case, it is hoped that most device suppliers will quickly issue a special patch or update, without waiting for the next scheduled full device update.

Therefore, device manufacturers should provide a single SBOM for all the software (including the real-time operating system if present) and firmware in the device. They should re-issue the SBOM whenever they "push out" a new device software update. However, even though the device software is composed of multiple software and firmware products, the manufacturer will not usually provide SBOMs for those individual products, although their customers would certainly welcome that.

4 IDENTIFYING VULNERABILITIES IN DEVICES VS. USER-MANAGED SOFTWARE

SBOMs are a tool for managing risks of two primary types: component licensing risks and component cybersecurity risks. While managing component licensing risks is important primarily for software developers, managing component cybersecurity risks is important for any organization that uses software to achieve their organizational goals – i.e., just about every public or private organization on the planet, whether a developer or an end user of software.

In fact, probably the most important type of software cybersecurity risk today arises from the presence of vulnerabilities in software components. Vulnerabilities appear frequently, but they are difficult to identify and qualify. This makes component vulnerability risk management one of the hardest tasks confronting any organization that needs to manage security risks posed by its use of IoT or IIoT devices.

4.1 USER MANAGED SOFTWARE RISK MANAGEMENT

How does an organization manage component vulnerability risks using SBOMs? For user-managed software products, the steps include those below. Note that all steps after step A require use of an automated tool that can ingest SBOMs and VEX documents, or else a third party service that performs these same tasks on behalf of the organization. Currently (July 2022), there are no tools or subscription-type services available that perform all of these steps, although it is likely that situation will change soon.

A. For every important software product used by the organization (other than cloud-based software[10]), obtain the supplier's agreement to provide a new SBOM whenever any change has occurred in the software.

[10] Cloud-based software probably contains as many vulnerabilities as on-premises software, so it should be tracked just as closely. However, as of the writing of this article (April-May 2022), there has been little discussion in the SBOM community about how to manage component vulnerability risks for cloud-based software, so it is impossible now to provide guidelines for that process.

B. Parse SBOMs as they are received, yielding the names of components included in the software product. Using those names, look up each component in a vulnerability database like the US National Vulnerability Database (NVD)[11].

C. Retrieve a list of any open vulnerabilities found in the NVD, for each of the components in a software product. In the NVD, vulnerabilities are identified with a CVE record[12].

D. Compile these vulnerabilities into a list of open component vulnerabilities for each software product for which the user organization receives SBOMs. These lists should be updated regularly (preferably daily) by checking with the NVD. Initially, every vulnerability on the list should be assumed to be exploitable in the product itself, until a VEX document from the supplier changes that designation. Note that the list will change from version to version of the product, as new vulnerabilities are identified, and existing ones are patched. Therefore, if an organization operates multiple versions of one software product, they should maintain this list for each version.[13]

E. Parse VEX documents received from suppliers. The primary purpose for which VEXes were developed is to identify component vulnerabilities that are not exploitable in the full product, even though they are exploitable in the component by itself. Therefore, whenever the user receives a VEX stating that a particular vulnerability isn't exploitable in one or more versions of the product, the tool should remove that vulnerability from the list of open component vulnerabilities in the appropriate version(s) of that product. This way, the list of vulnerabilities for a product will always include only vulnerabilities that are known to be exploitable in the product.

F. If the above steps are performed regularly (preferably every day or even continually), the user organization will be able to obtain, at any time, an up-to-date list of exploitable component vulnerabilities in each of the user-managed software products it operates. This

---

[11] *https://nvd.nist.gov/*. In order to identify vulnerabilities for a particular component in the NVD, the user must have a CPE (Common Platform Enumeration) name for the component. The format for CPE names is described here: *https://cpe.mitre.org/specification/*. To eliminate the need for SBOM users to go through the often-laborious process of searching for a CPE name on their own, it is the authors' opinion that the supplier of the product described in the SBOM should attempt to provide the CPE name for every component listed in the SBOM.

If the developer of a component has never reported a vulnerability for it, there may not be a CPE name for the component. If the supplier of the product cannot find a CPE name for a component in the product, they should create one according to the MITRE specification and include that CPE name in the SBOM. The supplier should try never to leave the component CPE name field empty.

[12] Described at *https://cve.mitre.org/cve/identifiers/*

[13] This list is very sensitive information. It might well provide a "roadmap for the attacker" if the wrong party got ahold of it. It should never be transmitted over the internet unencrypted.

list can be "fed into" configuration or vulnerability management tools, so that the user organization can track and apply patches for exploitable component vulnerabilities in the software products installed on their network.

G. The lists can also be used to coordinate with the supplier of each product, to learn when they will patch each important exploitable vulnerability[14]. That is, for each of these exploitable vulnerabilities (or at least those with higher CVSS[15] scores), the user will apply gentle (but firm) pressure on the supplier to develop a patch quickly.

The last step points to one of the main reasons why SBOMs are promoted as a product security tool. It is likely that many software suppliers do not patch vulnerabilities found in components of one of their products with the same promptness that they (hopefully) patch vulnerabilities found in their own code. The reason this might happen is simple: Customers don't bother suppliers about vulnerabilities they don't know about. The NVD does not track vulnerabilities in *components* of a software product, but only in the product itself. Thus, the customer has no way of learning about component vulnerabilities, unless they have an SBOM to tell them what the components are.

With an SBOM, the customer can learn about both components and - through the NVD or another vulnerability database like OSS Index[16] - the vulnerabilities found in them. When a customer learns that the NVD lists a serious vulnerability in a component of a software product they use, and when they believe this vulnerability is exploitable in the product, they can and should contact the supplier immediately, to ask when they will develop a patch.

4.2 Device Software Guidance

The above discussion relates to user-managed software. When it comes to IoT and IIoT devices, the situation changes, making it harder for a user to identify component vulnerabilities. Here are the challenges involved:

- Because the supplier releases updates to all the software in the device as a single unit, the most efficient way for a device user to learn about current vulnerabilities applicable to software in the device is to look up the device itself – using the version number of the most recent device update – in the NVD.
- However, many device suppliers have never obtained CPE names for their products, meaning they are not reporting vulnerability information for the device to the NVD. This is not because the software and firmware in the device have no vulnerabilities; it is almost

[14] Since no supplier can patch every vulnerability that applies to their products, the supplier should consult with their customers to prioritize vulnerabilities to patch.

[15] More information is available at *https://nvd.nist.gov/vuln-metrics/cvss*.

[16] *https://ossindex.sonatype.org/*

certainly because the device manufacturer isn't looking for vulnerabilities in the first place.[17]

- Of course, if the user knows the CPE name and current version number for at least some of the software products installed in the device, they can look up each of those products in the NVD. However, the user will usually not have this information, unless they have a complete, up-to-date SBOM for the device.
- However, even if a user does know at least some CPE names for software products installed in the device, and even if they discover a serious vulnerability for one of those products, they will not be able to install any patch they receive from the software supplier. This is because a device user is normally not able to apply patches directly to software installed in their device. The user must rely on the device manufacturer to provide the patch as part of their next full device update.

In other words, the organizations responsible for patching vulnerabilities found in software products installed in an IoT or IIoT device are the suppliers (which might be open source communities) of the installed products. But, because the users of the device are not the direct customers of the software suppliers, they cannot make requests (or demands) of those suppliers; the manufacturer of the device needs to do that. If for some reason, the supplier of one of the software products installed in the device does not patch a serious vulnerability, it is the device manufacturer's responsibility to patch the vulnerability themselves, or else replace the product altogether.[18]

---

[17] One firmware security specialist identified about 2200 unpatched vulnerabilities in just a single firmware product included in an IIoT device used in critical infrastructure environments. Yet, because the device didn't have a CPE name, a search for it in the NVD would yield no vulnerabilities, leaving the user with a false sense of security. In fact, the device manufacturer, which sold around 50 different devices, had never registered a single one of their devices in the NVD. This means that a customer searching for any one of the devices in the NVD might be led to believe that it has no vulnerabilities. The researcher stated that this is a common problem with intelligent devices.

[18] Of course, if the "supplier" of a product is an open source community, there may be nobody to fix a vulnerability. This could happen if the community has dwindled in recent years, and nobody is producing patches for the product anymore. This is why, for every open source software product included in their device, the device manufacturer needs to monitor the community supporting the product. The manufacturer should remove the product from the device if it becomes clear that the community has disappeared or is in the process of disappearing (e.g. no commits on GitHub within for example the last six months); this is analogous to removing a proprietary product from the device when its vendor has stated the product is in "end of life" (EOL) status.

If the community has disappeared or the product supplier has gone out of business, but the device manufacturer believes the product cannot be removed from the device, they need to be prepared to address any new serious vulnerabilities by immediately developing a patch, or even fixing the code

5 WHAT SHOULD THE USER ORGANIZATION REQUIRE OF THE MANUFACTURER?

The customer of an IoT or IIoT device should require the device manufacturer to develop and follow a Component Security Management Program, including the following provisions[19]:

1. Register the device in the National Vulnerability Database with its own CPE name.
2. Provide, to each device customer that has signed an appropriate non-disclosure agreement (NDA), a complete software bill of materials (SBOM) for the device, in either the CycloneDX or SPDX format. The SBOM must list every software or firmware product installed in the device, including that product's CPE name and version number. A new SBOM must be issued – along with a new version number – whenever a software update is issued for the device.
3. Whenever a vulnerability in one of the software or firmware products installed in the device is determined not to be exploitable in one or more versions of the device, the manufacturer should provide a VEX document to each customer, informing them of this fact.[20]. All other component vulnerabilities will be considered by the customer to be exploitable; the customer will hold the manufacturer responsible for ensuring they are patched or otherwise mitigated.
4. Forward all SBOMs and VEX documents received from suppliers of software or firmware installed in the device to customers of the device, as long as the customer has signed whatever NDA is required by the software or firmware supplier.

themselves. It is unacceptable for a manufacturer to leave a product in their device if it contains a serious unpatched vulnerability.

The above applies to open source software products, but a similar risk applies to proprietary software: since proprietary software is almost always provided as compiled binaries, if the supplier of a software product installed in the device goes out of business, the device manufacturer will have no way to patch a serious vulnerability. One important risk-reduction measure a manufacturer can take is to require that the product supplier (perhaps for an extra cost) provide the source code for any product provided as binaries. That way, if the supplier goes out of business, the manufacturer can fix the product themselves.

[19] These provisions are not intended to be legal language, although they can form the basis for contract terms.

[20] A single VEX document can list the exploitability status of multiple vulnerabilities in one product or in multiple products. However, given how new the VEX concept is, it is better now to confine one VEX to one product, although addressing multiple vulnerabilities for one product in a single VEX is completely acceptable.

5. Timely report all vulnerabilities, discovered in software or firmware products installed in the device, to the NVD, under the device's CPE name[21].
6. Commit to providing a fix for any device vulnerability, which has been assigned a CVSS score of 9.0[22] or higher, within five days[23] of discovery of the vulnerability.
7. Commit to the following: If the manufacturer modifies any open source product before installing it in the device, the manufacturer will provide appropriate notification of this with the SBOM, so the customer is informed that this component has been modified. The manufacturer should inform customers of any serious vulnerabilities that appear in this modified component.
8. Commit to obtaining source code for compiled software products that are installed in the device if there is a reasonable possibility that the supplier of the product will not be in business within the normal lifetime of the product.
9. Commit to regularly assessing all software or firmware products installed in the device for security issues including:
    a. Unpatched exploitable vulnerabilities that have a CVSS score of 5.0[24] or higher;
    b. One or more unpatched exploitable vulnerabilities with a CVSS score of 7.0 or higher, which have remained in the product for six months or longer[25];
    c. One or more components with an announced End of Life or End of Service date less than six months[26] in the future;

[21] When performing due diligence before purchasing an IoT or IIoT device, the user should check the NVD to ascertain whether a) a CPE name is registered for the device, and b) at least a few vulnerabilities have been reported for it. If the answer to either of these questions is negative, that should raise the suspicion that the manufacturer does not report vulnerabilities at all. Before purchasing from this manufacturer, the user should discuss this issue with the manufacturer. The user should obtain their commitment to register their device in the NVD and start reporting any exploitable vulnerability identified in any software or firmware product installed in the device.

[22] The exact score for this provision can be negotiated between the device customer and the device manufacturer.

[23] The exact number of days can be negotiated between the device customer and the device manufacturer.

[24] The exact score for this provision can be negotiated between the device customer and the device manufacturer.

[25] The exact CVSS score and time period can be negotiated between the device customer and the device manufacturer.

[26] The exact time period can be negotiated between the device customer and the device manufacturer.

d. One or more components that are more than two years[27] behind the current version;

e. One or more open source components that have not been updated or patched for six months[28] or more;

f. No CPE name registered for the product in the NVD; or

g. Open source components that appear to have had a single maintainer[29] for one year[30] or more.

6 CONCLUSION

An SBOM for an IoT or IIoT device differs in important ways from an SBOM for a "user-managed" software product. This is because the device user does not have a direct relationship with the suppliers of the software and firmware products installed in the device; the user's only direct relationship is with the device manufacturer. The user must rely on the manufacturer to track and manage vulnerabilities in device software, as well as apply patches developed by the suppliers of the software products installed in the device.

Because of this, the manufacturer cannot take a "hands-off" relationship with their customers; they must take primary responsibility for software vulnerability management and patching. Most importantly, the manufacturer needs to:

1. Provide a complete software bill of materials for the device to their customers, which is updated whenever the manufacturer updates any of the software in the device; and
2. Register the device in the National Vulnerability Database and report vulnerabilities (CVEs) identified in software and firmware products installed in the device as vulnerabilities in the device itself. This allows a device user to track device vulnerabilities and make sure they are properly addressed, through patching or other means.

7 ACKNOWLEDGEMENTS

The views expressed in the IIC Journal of Innovation are the author's views and do not necessarily represent the views of their respective employers nor those of the Industry IoT Consortium®.

[27] The exact number of years or months can be negotiated. In some cases, it may be preferable for the customer not to require an upgrade after a certain period of time, but instead set out certain criteria for upgrading, such as a serious unpatched vulnerability.

[28] The exact time period can be negotiated between the device customer and the device manufacturer.

[29] See *https://en.wikipedia.org/wiki/Software_maintainer*.

[30] The exact time period can be negotiated between the device customer and the device manufacturer.

➢ Return to *IIC Journal of Innovation landing page* for more articles and past editions.