



Practical Guide to SBOM

As more software is ingrained into almost every organization, software developers are under more pressure to rapidly deliver revenue-impacting applications. With speed of delivery as a requirement, developers use third-party libraries to perform actions that would otherwise take days – or weeks – to develop. The issue, however, is with this benefit comes the introduction of security risks. Risk of vulnerabilities is a concern as organizations depend more on third-party libraries that must be fully maintained and updates must be frequently deployed to remediate security issues.

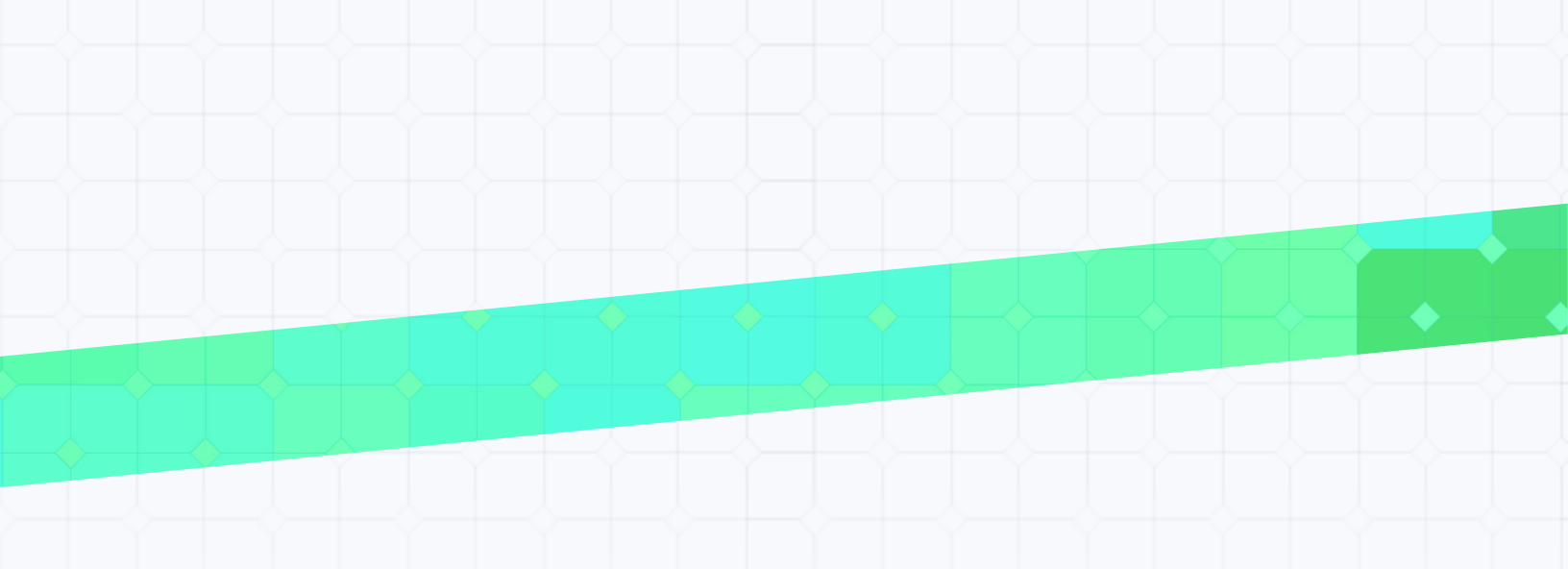
It's not uncommon for a corporate application to have dozens of third-party libraries, and they must be tracked so that any updates can be tested and installed as part of the software development lifecycle. This led to the development of a Software Bill of Materials (SBOM), and a recent [US Executive Order on Improving the Nation's Cybersecurity](#) requires any government entity to have one.

What is an SBOM?

Software Bill of Materials Lifecycle



An SBOM is similar to a supply chain document in manufacturing and product development. In product development supply chains, the manufacturer uses parts from specific vendors, installs components to build the product, and then tracks a product's travel history from the manufacturer to the retail store where you purchase it.



As an example, server machines in a network environment are built using vendor parts delivered to the manufacturing plant, the server is built, and then it travels from one location to another until it arrives at a data center where it's installed. Every step in this process is a part of the supply chain.

At any point in the manufacturing process, an attacker has opportunities to manipulate parts, inject their own malicious component, steal sensitive or proprietary information (might it be text, code, tokens etc.), or otherwise damage the integrity of a product. Similar to the manufacturing process, attackers have opportunities to inject their own malicious content into software. Attackers could hack the developer and inject their own code or silently take over an open-source repository. In open-source libraries, a malicious threat actor could contribute code with obfuscated malicious functions that the repository owner does not catch during a pull request review.

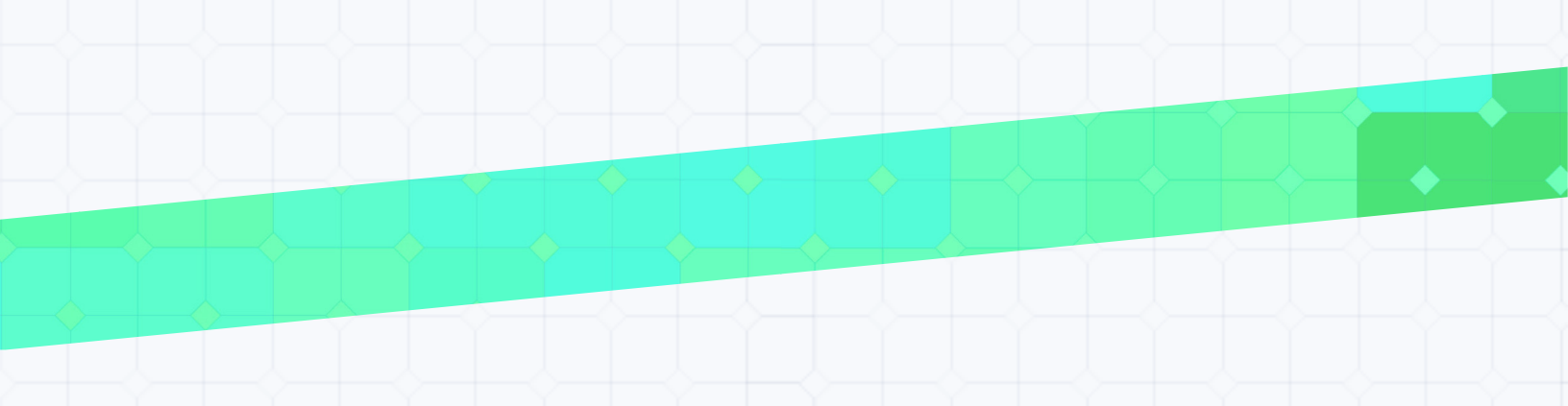
An SBOM is a document that tracks the supply chain in software development. It keeps track of every third-party library, script, CI/CD application, artifact (e.g., Docker), license, and version integrated into your applications. For small businesses with just one application, it might seem like tracking the supply chain is simple, but it can soon become overwhelming as your software development lifecycle adds several more moving parts. An SBOM ensures that every element integrated into your software including dependencies are tracked and can be audited for security issues.

What are SBOM's Benefits?

You might wonder why you would want to add SBOM generation to your software development lifecycle if you don't have government customers and don't plan to work in any public sector. Having an SBOM does more than just keep an organization compliant with government requirements. An SBOM tracks the supply chain in software development, so it has many benefits for software security and license tracking.

Key benefits of having an SBOM include:

- ◆ **Faster incident response:** with the SBOM document, security teams can quickly identify if any third-party software might be responsible for a data breach and contain it more quickly to limit damages.
- ◆ **Due diligence evidence for stakeholders:** any organization that keeps detailed audit trails of third-party tools will satisfy stakeholders and investors concerned with security of your software.
- ◆ **Convenient migration to other platforms:** whether you're migrating to the cloud or changing infrastructure, an SBOM will provide a way for operations to ensure that new platforms support all application dependencies.
- ◆ **Research into improvements:** some organizations perform yearly reviews to identify legacy applications and improve operational efficiency. An SBOM will provide easier identification of improvement opportunities for performance, better technology, security, and other benefits.
- ◆ **Penetration testing:** having a list of third-party tools makes it easier to automatically identify and scan for vulnerabilities across the environment.
- ◆ **Cryptographically sign your software:** use an SBOM to cryptographically sign applications to verify that users are not downloading altered binaries.

- 
- ◆ **Help customers remediate vulnerabilities:** any currently installed libraries are documented with remediation guidance to help internal or external clients with their security improvements.
 - ◆ **Validation for data centers:** use an SBOM to show clients that any infrastructure within the data center is secure and updated with the latest patches.
 - ◆ **License tracking and verification:** any licensing errors can be costly for enterprise organizations. An SBOM can be used to track licensing compliance and validate that every application installed across the organization's environment has proper licensing and unlicensed applications can be uninstalled or licenses purchased.
 - ◆ **Compliance with US government mandates:** if you plan to work with the US government or currently have US government contracts, you need an SBOM to avoid issues with non-compliance.

Anatomy of an SBOM

Most organizations need an SBOM for compliance and security, but where do you start? Even if you already have means to generate an SBOM, it's important for you and your developers to understand the anatomy of an SBOM to identify issues and make decisions based on output from the document.

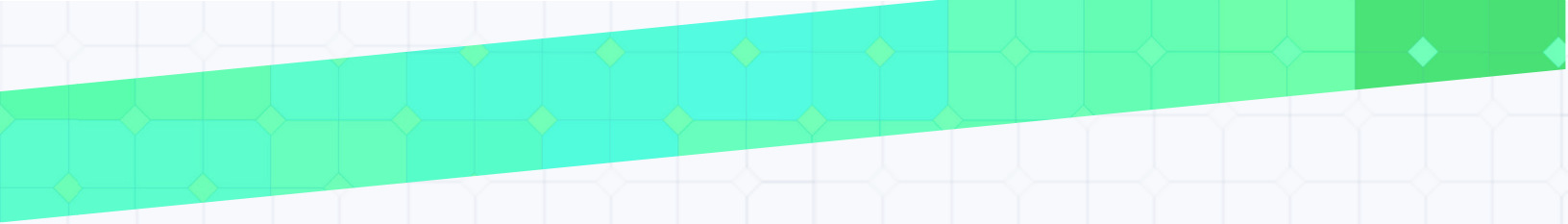
The National Institute of Standards and Technology (NIST) lays out some [guidelines](#) for an SBOM document, but organizations can define the document in whatever way works best for their security. Although organizations can format the document in any way that they want, they still must have specific attributes in the SBOM document to stay compliant.

The data fields that must be in an SBOM are:

- ◆ **Component name:** for example, the library or third-party tool integrated into your software or lifecycle.
Component version: the version currently in use and installed.
- ◆ **Unique identifier for the software:** a unique identifier can be a [CPE](#), [PURL](#), or [SWID](#). All three unique identifier formats have their own specifications for the way an identifier is generated, so organizations must choose the format that's right for their business.
- ◆ **Relationship with other dependencies:** a brief description of the way the component interacts with the software, including if it interacts with other dependencies.
- ◆ **Developer name:** the individual's name or business name of the developer that created the component.
- ◆ **Name of tool used to create the SBOM document:** if you're an Apiiro customer, this data field would be Apiiro.
- ◆ **Document creation date and time:** the timestamp for when the SBOM document was generated.

Your developers can determine the way this document and its information are laid out and organized, but they must be present for the SBOM to follow compliance and stay resourceful for the development lifecycle. A well organized SBOM document makes it much easier for developers and security teams to get the information they need in the event of an update/patch, determine if any component was recently compromised, and determine if a review of infrastructure is needed to identify if the organization is at risk.

A good SBOM is more than just data fields. It also has automation behind its generation and your organization should put practices and procedures in place for it. Many development and security teams focus on the data fields, but attention should be paid to the other components in SBOM anatomy.



You could build your own automation tool by using the output from the various generation tools, but products such as Apiiro will perform the automation for you, making it much more convenient and efficient to add automation to the SBOM's document generation. An SBOM will evolve as more libraries and software are added to your infrastructure, so automation makes the process much more efficient and reduces the risk of mistakes or oversights.

SBOM output can also be in standardized data formats such as JSON or XML. These documents are used by your own software tools to analyze and report information to developers, security teams, and operations people. Output in standard data formats can then be used to build reports in formats best for stakeholders such as Word documents or Excel spreadsheets, but this option must be configured and built by your own developers.

Most development and security teams have their own procedures in place to make the software development lifecycle efficient, and the generation and automation of an SBOM should be no different. Here are a few items to consider when you put your SBOM procedures in place:

- ◆ **Frequency of document updates:** any time a new application is installed, library is integrated into your software, or infrastructure applications change, you must generate a new SBOM to track the software's information.
- ◆ **Dependencies:** you might have dependencies within dependencies, so make sure that you drill down into all components to include them in your SBOM documentation.
- ◆ **Identification of no dependencies:** some applications have no dependencies, and if this is the case for your application, it should be noted that the software has no dependencies. If the number of dependencies is unknown, then the SBOM should specify it.
- ◆ **Generation and storage:** An SBOM document should be generated in a timely manner after changes to software, and it should be stored in a place where anyone who needs to read it has permission to do so. These documents contain sensitive information, so only necessary staff members and vendors should have access to an SBOM document.

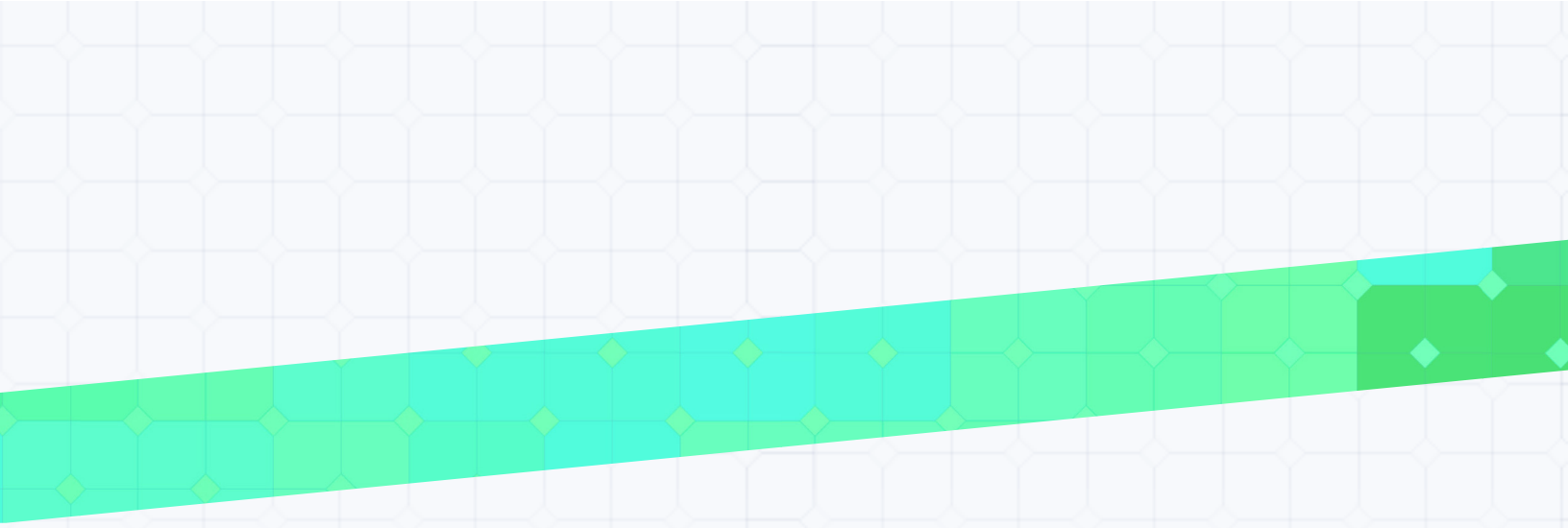
- ◆ **Access control:** in enterprise environments, elements of an SBOM document are made available to only specific people. Granular access must be defined and the policies should specify which groups should have access to specific portions of the SBOM information.
- ◆ **Tolerance for inaccuracy:** the process of creating an SBOM is still relatively new, so improvements will be necessary for any organization. In most cases, a bit of trial-and-error and experimentation are necessary until the best solutions are identified.

SBOM Formats

To stay compliant, the US government supports three formats, each of which serves a different purpose. The format you choose will depend on the purpose of the document. Most companies use SBOMs for supply chain security, but they also track licensing and any changes to the software development lifecycle.

Here is a brief breakdown of each format:

| | CycloneDX | SPDX | SWID |
|------------------------------|--|-----------------------------------|--|
| Supported by | OWASP | Linux Foundation | NIST |
| Format standard | No standard format, but OWASP defines specifications . | ISO/IEC 5962:2021 | ISO/IEC 19770-2:2015 |
| Unique identifiers supported | SWID, CPE, PURL | CPE, PURL | SWID |
| Target audience | Developers and Security teams | Developers and administrators | Governments and developers working for governments |
| Output formats supported | json, xml, protobuf | xls, rdf, json yml | .xml |



As you can see from the file formats listed above, CycloneDX has much more flexibility. It's one reason why many security teams prefer the CycloneDX format, but SPDX and SWID have their purposes depending on your goals.

A Brief Overview of CycloneDX Format

The CycloneDX format is primarily for documenting the cybersecurity of your software. It's a flexible standard, because it allows the document creator to generate a custom format best designed for their own customers. OWASP provides an open-source SBOM generation tool in several languages.

Primary output from the generation tool is in JSON and XML, but developers can use this output to create their own documentation format. However, the custom documentation should be thorough with all data fields necessary for it to be considered a compliant SBOM.

The generation tool is built for developers, so stakeholders with no development experience will find it difficult to use. Apiiro's SBOM generation tool takes care of the technical side and allows a much easier and convenient way to create an SBOM and ensures that it contains enough information to validate the security of your systems.

The following JSON output is an example of CycloneDX output:

JSON Example

```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.4",
  "serialNumber": "urn:uuid:3e671687-395b-41f5-a30f-a58921a69b79",
  "version": 1,
  "components": [
    {
      "type": "application",
      "name": "Acme Application",
      "version": "9.1.1",
      "cpe": "cpe:/a:acme:application:9.1.1",
      "swid": {
        "tagId": "swidgen-242eb18a-503e-ca37-393b-cf156ef09691_9.1.1",
        "name": "Acme Application",
        "version": "9.1.1",
        "text": {
          "contentType": "text/xml",
          "encoding": "base64",
          "content": "PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGludXRmLTgiID8+"
        }
      }
    },
    {
      "type": "library",
      "group": "org.apache.tomcat",
      "name": "tomcat-catalina",
      "version": "9.0.14",
      "purl": "pkg:maven/org.apache.tomcat/tomcat-catalina@9.0.14"
    }
  ]
}
```

Image source: <https://cyclonedx.org/use-cases/>

A Brief Overview of the SPDX Format

Introduced by the Linux Foundation, SPDX formats are used to scan applications and validate that your organization has proper licensing to run them. Licensing violations can be severe for large enterprises that use applications without knowing if they can be used for commercial purposes. Some applications allow individuals to use them for free, but they must be paid for if used for commercial purposes.

As more software licenses structures are introduced, it becomes a challenge for organizations to ensure that they follow regulations properly. An SBOM discovers applications and helps organizations determine if the installed product is compatible with licensing.

Linux Foundation has a Git repository with open-source [SPDX generation tools](#), but again you need to understand code and how to execute it. The following is a simple example of SPDX output:

Note: You can sort by each column by clicking on the column header.

| Full name | Identifier | OSI Approved? | Text |
|--|-------------|---------------|------------------------------|
| 3dfx Glide License | Glide | | License Text |
| Abstyles License | Abstyles | | License Text |
| Academic Free License v1.1 | AFL-1.1 | Y | License Text |
| Academic Free License v1.2 | AFL-1.2 | Y | License Text |
| Academic Free License v2.0 | AFL-2.0 | Y | License Text |
| Academic Free License v2.1 | AFL-2.1 | Y | License Text |
| Academic Free License v3.0 | AFL-3.0 | Y | License Text |
| Academy of Motion Picture Arts and Sciences BSD | AMPAS | | License Text |
| Adaptive Public License 1.0 | APL-1.0 | Y | License Text |
| Adobe Glyph List License | Adobe-Glyph | | License Text |
| Adobe Postscript AFM License | APAFML | | License Text |
| Adobe Systems Incorporated Source Code License Agreement | Adobe-2006 | | License Text |
| Affero General Public License v1.0 | AGPL-1.0 | | License Text |
| Afmparse License | Afmparse | | License Text |
| Aladdin Free Public License | Aladdin | | License Text |
| Amazon Digital Services License | ADSL | | License Text |
| AMD's plpa_map.c License | AMDPLPA | | License Text |
| ANTLR Software Rights Notice | ANTLR-PD | | License Text |
| Apache License 1.0 | Apache-1.0 | | License Text |
| Apache License 1.1 | Apache-1.1 | Y | License Text |
| Apache License 2.0 | Apache-2.0 | Y | License Text |

Image source: <https://spdx.dev/resources/use/>

A Brief Overview of the SWID Format

The SWID format, which was originally created in 2009 and later updated in 2015, is generally used in development environments to validate that software has not had any unauthorized changes, prevent installation of unauthorized software, detect unauthorized changes to installed software, and identify vulnerabilities in software that must be patched. SBOMs with this format are generally integrated with the software development lifecycle especially in large enterprise organizations that have several developers working within a trusted environment.

NIST introduced NISTIR 8060 guidelines as well as modifications that lay on top of SWID to make it useful for security use cases and to show accountability across development.

The following is an example of SWID tagging output:

SWID file for ACME Infusion pump

```
<Softwareidentity
  name="Infusion pump"
  tagId="pkg:supplier/ACME/Infusionpump@2.0"
  version = "2.0"
  tagVersion="1">

  <Entity name="ACME Corporation"
    regid = "acme.com" role="tagCreator softwareCreator"/>
  <!--Authoritative tag. SBOM of this component is KNOWN -->
  <!--For dependent complete, check their respective SWID file -->

  <Link rel="component" href="swid: pkg:supplier/Microsoft/Windows10@1909" use ="required"/>
  <Link rel="component" href="swid: pkg:supplier/Microsoft/SQLServer2016@1.0" use ="required"/>
</Softwareidentity>
```

SWID file for Windows 10 V1909

```
<Softwareidentity
  name="Windows 10"
  tagId="pkg:supplier/Microsoft/Windows10@1909"
  version = "1909"
  tagVersion="1">

  <Entity name="Microsoft Corporation"
    regid = "microsoft.com" role="tagCreator"/>
  <!--Non-authoritative tag. SBOM of this component is UNKNOWN -->
</Softwareidentity>
```

SWID file for SQL Server 2016

```
<Softwareidentity
  name="SQL Server 2016"
  tagId="pkg:supplier/Microsoft/SQLServer2016@1.0"
  version = "1.0"
  tagVersion="1">

  <Entity name="Microsoft Corporation"
    regid = "microsoft.com" role="tagCreator"/>
  <!--Non-authoritative tag. SBOM of this component is UNKNOWN -->
</Softwareidentity>
```

Image source: https://www.ntia.gov/files/nt...om_generation_v1.pdf



SBOM Use Cases

Since SBOM creation is relatively new, you might wonder where you would fit the automation process into your own systems. An SBOM can help your organization's credibility in several situations and help improve security.

Here are some examples of where an SBOM process can help an organization:

- ◆ **Regulatory compliance:** taking effective measures to ensure security of your software is a requirement in compliance, and an SBOM is necessary for developers to do business with the US Federal Government.
- ◆ **Legacy compatibility and security:** it's not uncommon for legacy system security to be mistakenly overlooked. An SBOM will identify any vulnerable legacy systems and their dependencies so that they can be updated and secured.
- ◆ **Customer requests:** customers with high security standards might request an SBOM before they integrate your products with their own infrastructure. An SBOM ensures compliance and security for your customers, which increases business potential.
- ◆ **Fundraising, mergers and acquisitions, startups, and IPOs:** to get investors to financially support an organization, they need reassurance that your organization is protected from severe revenue-impacting events. Compliance violations, data breaches, and lawsuits from licensing negligence are all incidents that could lead to expensive damage to business continuity. An SBOM can be included in your organization's business plan and investor presentations to ensure that infrastructure is secure from exploits and licensing fines.



How Apiiro Can Help

Defining an entire procedure to handle automation and SBOM generation is a huge challenge, but Apiiro makes it much easier to generate a precise SBOM document. Indeed, our platform will automatically discover your assets and lower the time to remediate risks throughout the software supply chain.

Our inventory functionality is highly valuable for organizations that need automation in SBOM generation. Whether you need to create an SBOM to show to investors or a large enterprise organization that needs better licensing and security control of third-party applications, Apiiro will provide you with a fully automated way to generate it.

Discovering dependencies and software within your organization is the first step in reducing your attack surface. With Apiiro, your developers and security teams will have more visibility into all aspects of your software, open-source dependencies, and installed applications.