BETTER LIVING THROUGH SBOMS!

Tom Alrich

RF Tech Talk

March 21, 2022

What's an SBOM for?

- I'm sure your children have all asked you this question at one time or another.
- You might tell them that, if they had a software bill of materials for each of the video games they run, they might know which games contain Log4j.
- If a game does contain Log4j, they might not want to use it until the developer provides a patch.
- In general, an SBOM is a machine-readable list of the components in a software product.

Components – why are they a big deal?

- Why do software developers use components? Because they save the developer a huge amount of time (which = money, by the way).
- For example, Log4j performs logging functions for software. All software needs logging, but it would take a lot of time for a developer to write this code. The developer can download Log4J for free and insert it in their software. Then they'll have logging!
- The average software product has 135 components. Many products have thousands of components. 90% are open source.
- In some software products, more than 90% of the code is components. You could say the developer's job nowadays is just tying components together to make a new product.

Where do SBOMs come from?

- It's up to the software developer to develop SBOMs and distribute them to their customers. Most developers are producing them now for internal use, but they're not distributing them.
- Last May, President Biden issued Executive Order 14028 regarding cybersecurity in the federal government. One of the requirements was that all agencies request SBOMs from their software suppliers. Compliance is due this August.
- Even though the EO only applies to the government, it's likely that SBOMs will become a quasi-requirement for all software, no matter who it's sold to.

Machine readability

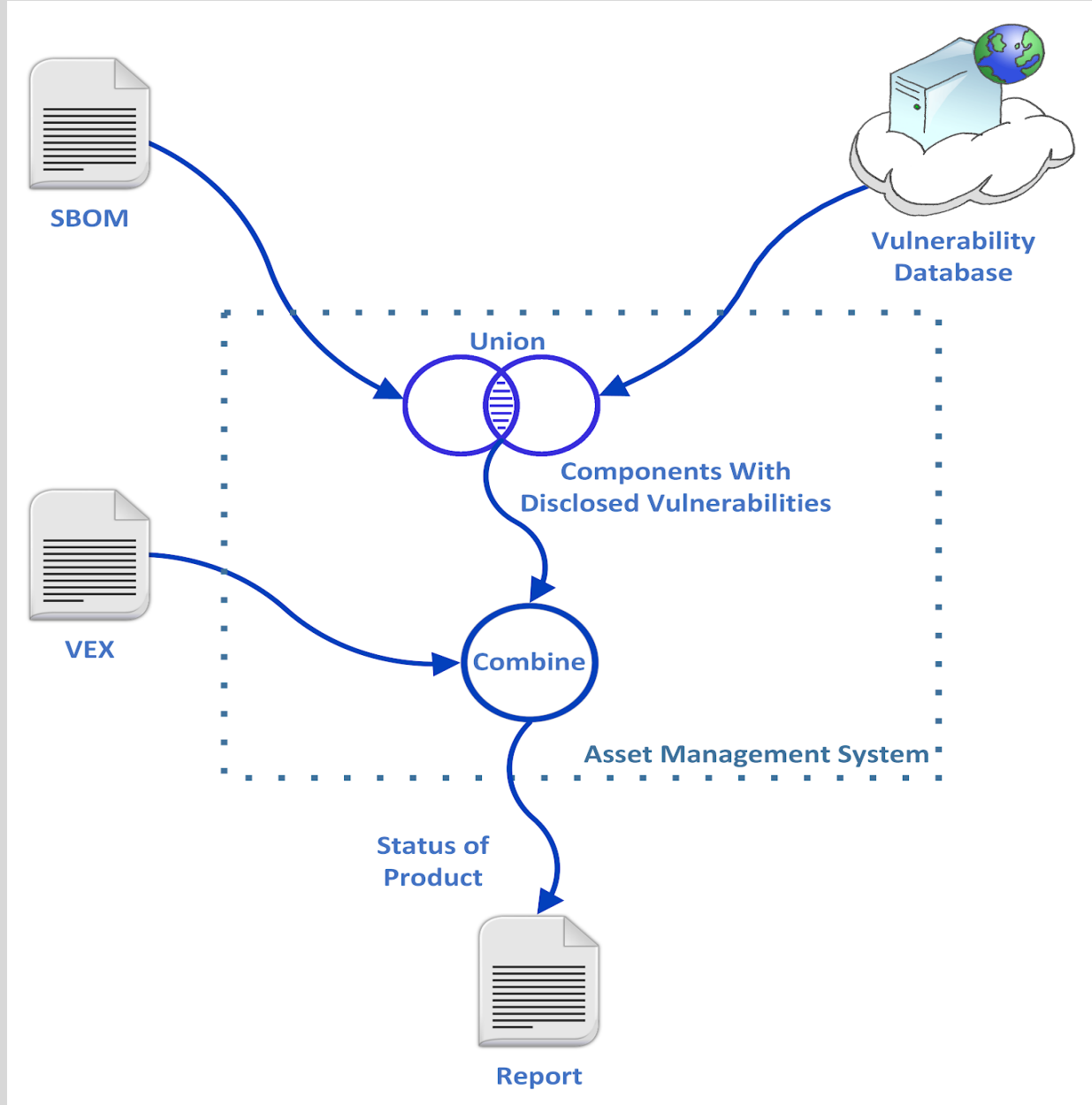
- Because of the huge (and growing) number of components in software today, it's impossible for a user to track all of the component vulnerabilities themselves.
- This is why SBOMS come in two machine-readable formats: SPDX and CycloneDX. With the right tool, you'll be able to "read" SBOMs and find, using the NVD, a list of vulnerabilities that apply to the components, then pass it on to tools for configuration or vulnerability management.
- One open source tool that does that now is Dependency-Track. You load an SBOM and then see all the vulnerabilities that apply to components. Plus, D-T will track new vulnerabilities for components as they're identified (D-T only reads CycloneDX SBOMs).

How could SBOMs help with Log4j?

- The most important use case for SBOMs is software vulnerability management. When a serious vulnerability is found in a component like Log4j, SBOMs will help you identify where that component is found in your network.
- But even when there's not a high-profile vulnerability, a software user should track at least high and critical component vulnerabilities in their software and contact their suppliers to find out when they'll patch them.

VEX

- One of the biggest problems facing SBOMs is that a huge percentage (perhaps 90% or even more) of component vulnerabilities aren't in fact *exploitable* in the final product. So, if Dependency-Track lists 20 component vulnerabilities in a product, 18 or 19 of them might not be exploitable.
- This is good news, except you need to know which component vulnerabilities are exploitable and which aren't. To do this, a supplier will issue a different machine-readable document, called a VEX (Vulnerability Exploitability eXchange).
- Your tool will need to process both SBOMs and VEXes. That way, you'll only see vulnerabilities that are exploitable – i.e., the 5% you should care about, not the 95% that you shouldn't care about.


SBOM
Vulnerability Database
Union
Components With Disclosed Vulnerabilities
VEX
Combine
Asset Management System
Status of Product
Report

What should you do about exploitable component vulnerabilities?

- Very few software users can patch vulnerabilities in the software they use – the supplier has to do that for you. What you can (and should) do is contact the supplier when you learn about a serious exploitable component vulnerability and ask when they'll patch it.
- You can also try to include contract language saying that, for any serious component vulnerability that is exploitable, the supplier will develop a patch within X business days (X is up to you. 5? 10? Certainly not 60…).

How can SBOMs help with procurement?

- When you're buying software, SBOMs can help you in two important ways. You should always request a recent SBOM for any product you're considering purchasing, as well as any VEXes that apply to it. You should ask the supplier to list all current open *exploitable* vulnerabilities for components. If they won't or can't give you an SBOM, that could be a strike against them in your evaluation.
- If there are any open serious exploitable component vulnerabilities, you should require in contract language that they be patched soon after the contract is signed.
- You can also ask questions like, "Are there any components in the product that are more than – say – 3 or 4 versions behind the current version?" or "Are there any components that are in end-of-life status?"

```
<?xml version="1.0" encoding="utf-8"?>
<bom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  serialNumber="urn:uuid:3e671687-395b-41f5-a30f-a58921a69b71" version="1"
  xmlns="http://cyclonedx.org/schema/bom/1.3">
  <metadata>
    <authors>
      <author>
        <name>Acme</name>
      </author>
    </authors>
    <component type="application">
      <name>Application</name>
      <version>1.1</version>
      <hashes>
        <hash alg="SHA-1">75068c26abbed3ad3980685bae21d7202d288317</hash>
      </hashes>
      <cpe>cpe:2.3:a:acme:application:1.1:*:*:*:*:*:*:*</cpe>
      <externalReferences />
      <components />
    </component>
    <manufacture>
      <name>Acme</name>
    </manufacture>
    <supplier>
      <name>Acme</name>
    </supplier>
  </metadata>
  <components>
    <component type="library" bom-ref="pkg:maven/org.bob/browser@2.1">
      <publisher>Bob</publisher>
      <group>org.bob</group>
      <name>browser</name>
      <version>2.1</version>
      <hashes>
        <hash alg="SHA-1">94568c26abbed3ad3980685deaf1d7202d268314</hash>
      </hashes>
      <cpe>cpe:2.3:a:bob:browser:2.1:*:*:*:*:*:*:*</cpe>
      <purl>pkg:maven/org.bob/browser@2.1</purl>
    </component>
```

```
<component type="library" bom-ref="pkg:maven/org.carol/CompressionEng@3.1">
      <publisher>Carol</publisher>
      <group>org.carol</group>
      <name>CompressionEng</name>
      <version>3.1</version>
      <hashes>
        <hash alg="SHA-1">63568c26aebad3ad398bb85ce1f1d7202d27731a</hash>
      </hashes>
      <cpe>cpe:2.3:a:carol:compression_eng:3.1:*:*:*:*:*:*:*</cpe>
      <purl>pkg:maven/org.carol/CompressionEng@3.1</purl>
  </components>
  <dependencies>
    <dependency ref="pkg:maven/org.bob/browser@2.1">
      <dependency ref="pkg:maven/org.carol/CompressionEng@3.1" />
    </dependency>
    <dependency ref="pkg:maven/org.bingo/buffer@2.2" />
  </dependencies>
  <compositions>
    <composition>
      <aggregate>complete</aggregate>
      <assemblies>
        <assembly ref="pkg:maven/org.carol/CompressionEng@3.1"/>
      </assemblies>
      <dependencies>
        <dependency ref="pkg:maven/org.carol/CompressionEng@3.1"/>
      </dependencies>
    </composition>
    <composition>
      <aggregate>unknown</aggregate>
      <assemblies>
        <assembly ref="pkg:maven/org.bingo/buffer@2.2"/>
        <assembly ref="pkg:maven/org.bob/browser@2.1"/>
      </assemblies>
    </composition>
  </compositions>
</bom>
```

How can SBOMs help in CIP-013 compliance?

1. CIP-013 requires you to “identify, assess and mitigate” supply chain cybersecurity risks *to the Bulk Electric System (BES).*
2. Probably the biggest such risk nowadays – along with ransomware – is software supply chain security risks. Think SolarWinds.
3. Since about 90% of the code in a software product nowadays consists of components, it follows that about 90% of software supply chain risk is due to components.
4. Without an SBOM, you’re entirely dependent on your supplier to tell you what the component risks are in their software – since you have no idea what components are in there otherwise.
5. Ergo, you need SBOMs, regularly updated.
6. See Fortress Information Security’s white paper on SBOMs and compliance (link on next slide).

Thank you!

Tom's blog: https://tomalrichblog.blogspot.com/

Tom's email: tom@tomalrich.com

To join CISA SBOM list: sbom@cisa.dhs.gov

SBOMs and Compliance paper: https://www.fortressinfosec.com/en-us/enhancing-cybersecurity-best-practices-with-sbom

Dependency-Track: https://dependencytrack.org/