DRAFT - Requirements for Sharing of Vulnerability Status Information

Draft version 0.1
2020-10-22

This document is available at:
https://docs.google.com/document/d/10GBfBV5UhRLEHBU3p9EtN6Fw_M9i04MIydRHwFIpBAE

Version 0.1

Preamble, draft, further intro language

Abstract

End-users of products that contain software are using SBOM ("Software Bill of Materials")[1] data to gain insight into the potential risks from any third-party software components that are incorporated in that product. (For the purposes of this paper a "product" can have several meanings including a physical form such as a device, a binary software form, a source code software form, or referring to a larger system of systems). Specifically, users want to know to what extent they are affected by known vulnerabilities, which includes vulnerabilities in upstream software components.

SBOMs provide this insight into the composition of the product, but they do so at a high level that does not convey the extent to which a known vulnerability can be exploited in the product. This lack of "exploitability" results in many "false positives" being represented in the SBOM data and thus obscuring the high risk exploitable vulnerabilities.

Software components may contain vulnerabilities, yet an attacker may not be able to exploit these vulnerabilities in the product due to the manner of utilization of the component or the configuration of the product.

In strict terms, for use in this discussion, a vulnerability is a set of conditions or behaviors that have a security impact, typically violating some security policy, possibly an implicit one (e.g., I do not want my software to run arbitrary code at someone else's behest). Vulnerabilities exist in or affect components. Vulnerabilities by their nature are exploitable, however a vulnerability can, in theory, exist in a component, and the vulnerability may not be practically exposed or exploitable. The goal of "VEX" is to convey this degree of exposure or exploitability, particularly in downstream components.

Commented [1]: Potential Names:
1.) VEX: Vulnerability Exploitability
2.) Vulnerability Status
3.) Vulnerability Disposition
4.) Disposition
5.) VINFO: Vulnerability Information
6.) VulnStat: Vulnerability Status
7.) Vulnerability Status
8.) Kvell: Yiddish for 'bragging' or 'expressing pride'
9.) SVRI: Security Vulnerability Research Information
10.) SVRD: Security Vulnerability Research Data

Commented [2]: I suggest striking this. Consumers just want to know. 1st, 3rd, hybrid, they don't care really.

Commented [3]: It is correct that consumers don't care if the vulnerability is upstream or not but in conjunctions with SBOMs, we are dealing exclusively with 3rd party (upstream) components because, except for unsupported products like STRUTS-1, there are rarely published vulnerabilities for 2nd party (supplier) products. I would rather leave in the upstream 3rd party components clause and provide a footnote that this is more general and would include 2nd party vulnerabilities, as well.

Commented [4]: Edited

Commented [5]: I suggest changing this to " "vulnerable versus exploitable" detail "

Commented [6]: We have terminology issues at this time. We are using the term "vulnerability" for security defects in components, such as Apache Batik, as they stand alone. Of course generally these vulnerabilities are not exploitable until the including component is included in a product.

In these discussions, we are using the term "exploitability" to indicate that the security defect can actually be exploited in the context of the product (or component, etc) that includes that vulnerability. I don't care what terms we use but we need to distinguish the two cases.

Commented [7]: we need to decide, at least for this document/VEX/SBOM, about vulnerability, exposure, and exploitability. I'll propose vulnerability is the thing (conditions) that exist, and exploitability is that an attacker can reach the vulnerability. I'm also OK calling the whole thing "vulnerability" but we may need to explain in the model that 1 vuln still exists yes but 2 can't be exploited.

Commented [8]: suggest changing: which actions should be taken to address

Introduction

As vulnerabilities in products become more apparent through mechanisms such as Software Bills of Materials (SBOMs) and software composition analysis (SCA) tools, this paper explores when a producer desires to inform the consumer of the impact of a given vulnerability upon a product. This document is meant to give guidance on what interfaces and information elements are necessary as part of the technical solution to describing the state of potential vulnerabilities in a product.

Commented [9]: suggest change to "work" - a "work" is representative of a component, and SBOM VEX doesn't generally represent the final product, it represents the "work" of the product; and multiple items that create the 'product'.

Commented [10]: ahh well that is not always the case. Bruce convinced me there is a valid use case where the VEX is product centric. Also we should wait for the glossary to harmonize terminology.

Commented [11]: A glossary is high priority

Commented [12]: suggest change to "which"

Commented [13]: suggest "work"

Situation

Increasingly products and their associated systems are making use of a wide assortment of both open source and commercial packages. These systems may incorporate software that is being utilized for purposes the original upstream authors never considered.
When a vulnerability is discovered in a component, it is eventually announced through a well-established CERT process, using the common vulnerability & exposure (CVE®) framework.[1]

While a CVE assigned vulnerability is present in the original implementation, that does not necessarily mean that it is exploitable or even present in final products or systems that contain the software component. This may be the case because of several different situations including:

- The vulnerable software is not executed
- The vulnerable code has been excluded from the executable due to conditional compile statements
- Existing security controls make it impossible to exploit
- The vulnerable code has been fixed by a downstream developer.

If history can serve as prologue, then only a very small percentage of published vulnerabilities can be exploited.[2] [3] Despite such analysis, raw percentages or averages are not necessarily useful risk measurements. A small percent of a large number may still be a meaningful number. Also, any single vulnerability can have a high potential impact and pose a significant risk. Many vulnerability management approaches aim to avoid false negatives (i.e. choosing not to respond to a vulnerability that turns out to be exploitable).

Therefore, the results of a vulnerability scan of an SBOM may result in many unexploitable vulnerabilities. This could obscure the risk of exploitable vulnerabilities. The VEX is an additional source of information from the Downstream Developer to further inform the End User and increase the "signal to noise" ratio of such an automated scan to provide useful and actionable intelligence.

If a developer is aware of the vulnerability, then they can choose to fix the vulnerability, request an upstream fix, or remove the impacted component from their product.
If end customers are aware of the vulnerability, they can mitigate it through upgrades, replacements, remove the product from service, or other means.
At any point in the supply chain, there is the potential for someone to create a mitigation that may not improve the security posture of the product and could introduce new vulnerabilities. Accurately knowing whether a particular product or system is vulnerable is critical to that evaluation.

[1] "CVE" is a trademark of MITRE Corporation.
[2] http://www.rsaconference.com/industry-topics/presentation/how-understanding-risk-is-changing-for-open-source-components
[3] https://www.sonatype.com/2020ssc

Commented [14]: Is VEX internal or external to SBOM? To what extent? Even if external need to map vuln to component.

Commented [15]: not vulnerable because 1. the SBOM tells me I do not include a vulnerable component, or

Commented [16]: or 2. although I do include a vulnerable component, not vulnerable for other reasons <-- this is VEX?

Commented [17]: I think both VEX and SBOM are in terms of the product (i.e. supplier, product_name, version, hash). One difference is the SBOMs need not be updated if the supplier knows what they are doing (admittedly rare) but VEX will need to be updated as vulnerabilities are discovered in 3rd party components. In a diagram sense the product "includes" or "is associated with" an SBOM but "is associated with" a number of VEX documents, which must be dated.

Commented [18]: this is ambiguous - open and closed sourced, 1st and 3rd party, commercial and non-commercial is the matrix. Trimming down is probably better then stepping into this matrix at all. suggest ... [1]

Commented [19]: We really need a glossary. In any ... [2]

Commented [20]: suggest "source, object code, and ... [3]

Commented [21]: suggest "other than those originally ... [4]

Commented [22]: suggest: "announced." strikeout ... [5]

Commented [23]: We need a glossary. The group ... [6]

Commented [24]: suggest "vulnerability identifier or ... [7]

Commented [25]: We need a glossary (I'll stop saying ... [8]

Commented [26]: suggest strikeout - product vs. code

Commented [27]: suggest "code or object"

Commented [28]: Software is better because it is more ... [9]

Commented [29]: suggest adding ", including code ... [10]

Commented [30]: citation needed

Commented [31]: I think a different statement would be ... [11]

Commented [32]: suggest strikeout - upstream may ... [12]

Commented [33]: suggest "refine"

Commented [34]: suggest strikeout

Commented [35]: suggest "fidelity"

Commented [36]: suggest "a"

Commented [37]: suggest remove ", "

Commented [38]: suggest strikeout

Commented [39]: suggest postpend "and upstream the ... [13]

Commented [40]: suggest "or remove, reconfigure or ... [14]

Commented [41]: suggest "a" vs "the"

Commented [42]: suggest "through hotfixes, patches ... [15]

Commented [43]: suggest "removal of" vs "remove"

Commented [44]: suggest ", off product hardening and ... [16]

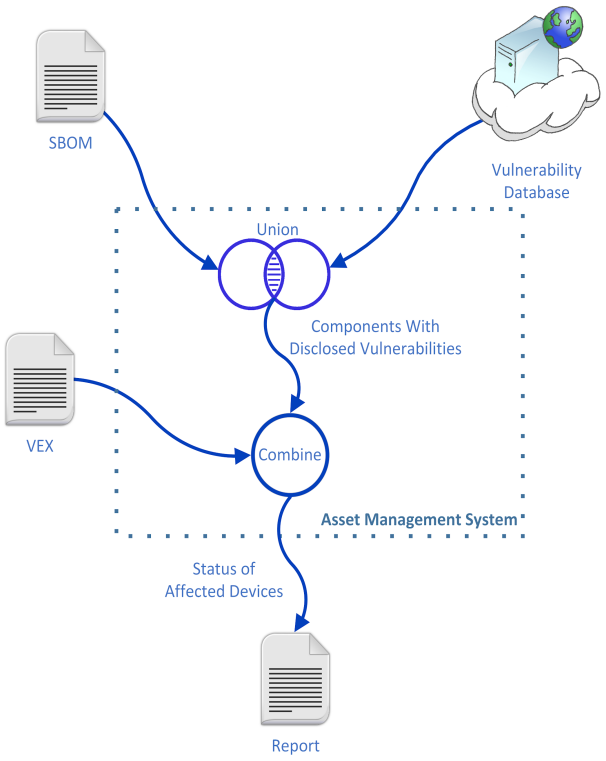
Commented [45]: suggest insert new paragraph line ... [17]

Commented [46]: suggest delete of ,

Ecosystem Participants

Art to fix all the wrong terminology I'm using.

Commented [47]: I'm going to need you to come in on Saturday.. and Sunday.. mmm... yeaahhh, ok? Good.



*Cloud server image courtesy of Josh Twist (http://www.thejoyofcode.com/Cloud_Artwork.aspx)

CERT / Vulnerability Database

Our story begins when a vulnerability has been reported to a CERT somewhere around the world for some software. At that time, a CVE is created and a CVE-ID is assigned by the supplier. At some later point, the information is shared with the public along with a reference to some remedial advice.

As part of the CVE information, a package name is identified. This package name may relate to a package name found in an SBOM or otherwise determined to be in a product via SCA analysis.

Commented [48]: IMO we can use this as an 'example' but shouldn't 'bind this' to SBOM intrinsically, Yet again we delve into use case specifics for existing processes that are not globally used by all. IMO SBOM VEXes should also exist independently of centralized sources and be able to stand independently of any centralized governing body of controls and systems, but, should be able to leverage such capabilities for existing open frameworks as the developer/company decides to do so - their own systems, other countries (CNVD), or CERT/VUDB, etc. at their discretion. This is why the 'key id' for the VEX itself is critical; because it provides its own direct identifier and is signed / hashed to ensure authenticity (to a reasonable degree).

Commented [49]: I think the "key id" is the (supplier, product name, version, hash) we defined in the framing document.

Is that agreed?

Commented [50]: Vulnerability advisories, NVD entries -- VEX helps these be more accurate/cheaper to produce?

Upstream Developers and Suppliers

The upstream developer typically controls a component that is used by others. This party will generally create CVEs (although others may perform this as well) and assign CVE-IDs.
This upstream developer may release software patches or new versions intending to correct the vulnerability and may update the CVE based upon that information.
Upstream developers may be under contract restrictions to support downstream developers, or they may provide the software on an “as is” basis, as would typically be the case with open source components.

Downstream Developers

These developers make use of software components developed and/or maintained by others (i.e. “upstream developers” or “suppliers”), and deliver their products to end-users or additional downstream developers.
Downstream developers:

- May or may not be required to make statements as to whether or not a particular version of their product contains a vulnerability.
- May want to issue patches for specific versions of their product.
- May be required by customers to make claims about the security posture of their product. Developers will want to avoid making false claims
- May want to be able to communicate that they are not vulnerable to a particular CVE across a range of products
- Will want to assure the attestation of any statements or claims.

Developer Tooling

Developer tooling is used to build and package products. As part of this activity, tools may be able to identify that a particular component has a vulnerability, based on a CVE related to an included package. Tightly integrated developer tooling, such as CI/CD, can produce additional information to better identify products which are or are not impacted by a CVE and which ones have been corrected.

End Users

End users are the people who use a finished product. End users do not further distribute a product to another user. Their interests are that the product runs securely in the end user’s environment, cannot maliciously be caused to malfunction, and cannot be caused to attack others. (e.g., cannot be exploited).

- End users have varying degrees of risk tolerance, based on where products are utilized, how they are used, and what harm can be caused by an exploit to them or others.
- Some environments such as exploit testing labs may even be risk-seeking, while others such as critical infrastructure providers will be highly risk-averse.
- End users may want a developer’s justification (from a list of potential justifications defined by this working group) for an exploitability posture declaration.

- End users may be required by regulators to receive vulnerability information and to maintain their products, or they may desire to do so on their own.
- End users will develop different levels of trust in the validity of Downstream developers' declarations

End User Tooling

End user tooling may include network and asset management systems, firewalls, mobile device managers, cloud orchestration systems, and many others.
Tooling in this context can be used to identify which vulnerabilities are present on which devices within an environment, to report and remediate the risk. For this to work, the tooling must be able to accurately state whether a vulnerability is present, exploitable, and harmful to the end user.

Value Chain

- Delivers components for use by others.
- Creates CVEs
- May create SBOMs

- Consumes and utilizes components created by others.
- May deliver a product to the end user or another downstream developer
- Creates CVEs
- Consumes upstream CVEs
- May create SBOMs
- May indicate Product vulnerability to upstream vulnerabilities
- Wants to minimize end user calls for non-affected Products

- Makes use of the Product
- Consumes CVE, SBOM, and VEX information
- Desire to understand their risk exposure
- Desire to minimize false positive vulnerabilities.

Page 3: [1] Commented [18] **mpaulsen Paulsen** **10/15/20 3:47:00 PM**

this is ambiguous - open and closed sourced, 1st and 3rd party, commercial and non-commercial is the matrix. Trimming down is probably better then stepping into this matrix at all. suggest "software components" and leaving it be.

Page 3: [2] Commented [19] **Bruce Lowenthal** **10/16/20 3:05:00 PM**

We really need a glossary. In any case, there are open source commercial products such as MySQL and Java (yeah, I know there are parts of the cryptographic libraries that are not open source because of licensing issues but nearly all the code is open source)

Page 3: [3] Commented [20] **mpaulsen Paulsen** **10/15/20 3:49:00 PM**

suggest "source, object code, and related" - a subtle difference exists between code and software - code is the work, software is the product.

Page 3: [4] Commented [21] **mpaulsen Paulsen** **10/15/20 3:50:00 PM**

suggest "other than those originally intended" and leaving it outside the focus of OSS TPC or CS TPC.

Page 3: [5] Commented [22] **mpaulsen Paulsen** **10/15/20 3:58:00 PM**

suggest: "announced." strikeout remainder. Why, I believe not everyone uses CVE and not every country is CVE'd and this is a very NA/US centric view. IMO it's just best to leave it to the dev / co, to decide how to do their announcement and not pigeon hole them. Further, CVE may change in the future and tagging it now with a (R) leaves the door open for future work which is unnecessary. Let's close the door now.

Page 3: [6] Commented [23] **Bruce Lowenthal** **10/16/20 3:09:00 PM**

We need a glossary. The group acknowledges there are multiple entities that assign vulnerability identifiers but that, for now, we are using CVE as a short hand for all of them and are not excluding them from SBOMs. We could replace CVE with terms like "vulnerability identifier" but some would confuse that with CWEs, etc. I think CVE is best for now in these kinds of documents but needs to be cleaned up an an actual standard.

Page 3: [7] Commented [24] **mpaulsen Paulsen** **10/15/20 3:59:00 PM**

suggest "vulnerability identifier or weakness categorization"

Page 3: [8] Commented [25] **Bruce Lowenthal** **10/16/20 3:11:00 PM**

We need a glossary (I'll stop saying this now). CVE, or things that act like CVEs is meant, not CWEs.

Page 3: [9] Commented [28] **Bruce Lowenthal** **10/16/20 3:13:00 PM**

Software is better because it is more encompassing. A broken table or config file can result in a vulnerability even though few would call it code or object.

Page 3: [10] Commented [29] **mpaulsen Paulsen** **10/15/20 4:02:00 PM**

suggest adding ", including code which is no longer maintained by the upstream provider."

Page 3: [11] Commented [31] **Bruce Lowenthal** **10/16/20 3:20:00 PM**

I think a different statement would be better because suppliers will not own up to this and this point does not need to be made to validate the need for VEX. We can say that many suppliers are unaware of published vulnerabilities in 3rd party components they include (e.g. because of lack of pervasive use of SBOMs)

Page 3: [12] Commented [32] **mpaulsen Paulsen** **10/15/20 4:05:00 PM**

suggest strikeout - upstream may provide vex, and ingested by downstream natively; or modified, or created by downstream.

Page 3: [13] Commented [39] **mpaulsen Paulsen** **10/15/20 4:08:00 PM**

suggest postpend "and upstream the fix if appropriate,"

Page 3: [14] Commented [40] **mpaulsen Paulsen** **10/15/20 4:09:00 PM**

suggest "or remove, reconfigure or disable"

Page 3: [15] Commented [42] **mpaulsen Paulsen** **10/15/20 4:10:00 PM**

suggest "through hotfixes, patches, workarounds, upgrades,"

Page 3: [16] Commented [44] **mpaulsen Paulsen** **10/15/20 4:11:00 PM**

suggest ", off product hardening and protection e.g. firewall filters"

Page 3: [17] Commented [45] **mpaulsen Paulsen** **10/15/20 4:12:00 PM**

suggest insert new paragraph line and title "supply chain" title of some sort