



OWASP 2023
GLOBAL
AppSec

WASHINGTON
DC
OCT 30 - NOV 3

From SBOMs to F-Bombs: Vulnerability Analysis, SCA Tools, & False Positives & Negatives

Kevin W. Wall <kevin.w.wall@gmail.com> , Oct 31, 2023

Copyright © 2023 – All rights reserved.



This work is licensed under a
Creative Commons Attribution-NonCommercial-
ShareAlike 4.0 International License
as specified at
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

Who is This Knucklehead?

- 20+ years systems programming, 20+ years in AppSec.
- Previously developed several *proprietary* systems libraries, including one for AppSec.
- OWASP ESAPI involvement (ESAPI dates back to 9/2007)
- Involved in ESAPI since June, 2009.
- Redesigned and re-implemented symmetric encryption
- Have been ESAPI co-lead since 2011.
- Buzzword free talk. No mention of ChatGPT or quantum computing! (Unless you consider SBOM a buzzword! :)

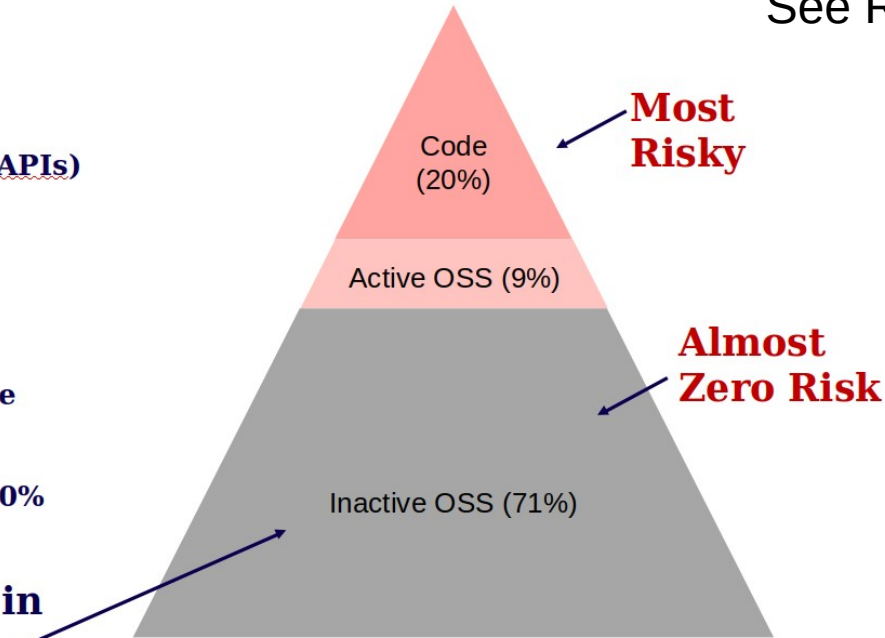
Why *This* Talk?

- I predate Software Composition Analysis (SCA) tools:
 - Manually analyzing patches to check if they allow exploitable paths since 2010.
- Jeff Williams (CTO of Contrast Security) gives talk confirming my suspicion that majority of SCA findings are false positives.
 - Vulnerabilities in dependencies exploitable < 29% of the time.
 - Matched my observation of my analysis doing secure code reviews for previous employers.
 - Agreed with my experience in writing up 11 ESAPI Security Bulletins.
- CVE-2023-24998 – my breaking point. *Lots* of F-bombs. 😡

SCA Bandages for Open Sores Security?

Open Source Security

- **Average app/API has...**
(based on 100K+ real world apps/APIs)
 - 9% active library code
 - 71% inactive library code
(never loaded, never invoked)
 - 62% of libraries are totally inactive
with zero active code
 - Most of the risk comes from the 20%
custom code
- “Vulnerabilities” reported in
inactive code are **false
positives**



* Contrast State of Open Source (OSS) Security Report
2021

Where We've Gone Astray

- We treat vulnerabilities in libraries same as we do in end products.
 - Try to measure the severity the same way for both (i.e., CVSS).
- We conflate severity with risk. CVSS measures severity, not risk.
- We treat risk as one-size-fits-all.
 - Not referring to risk appetite or risk acceptance here.
- Act as only option is patch *everything* now or accept risk and try to patch later.
 - Why not patch selectively? (Only patch what the application *uses*!)
 - Why not sandbox the application or use virtual patching with WAF, etc.?

Why Libraries and End Products Should be Treated Differently

- Some definitions:
 - Libraries: Think of them as any SDKs (e.g., DLLs, shared libraries, jars, etc.)
 - Can treat web services (e.g., SOAP and RESTful APIs) as libraries.
 - End products: executables, web servers, DBMS, appliances, etc.
- Dynamic vs static behavior.
- Extensible vs (mostly) non-extensible.
- Vulnerability severity estimates:
 - CVSS sucks, but IMO,
 - It's treated as a risk score even though First.org and NVD states that should not be.
 - It sucks *worse* for libraries: **actual worst case scenario CVSS portrays is rare for libraries.**
 - Conjecture: Average CVSS scores for libraries increasing much faster than they are for end products resulting in an increasing rate of CVEs that are rated “High” by NVD.

Special Instructions for CVSS and Libraries

- Quoting the FIRST.org's notes from § 3.7 of their CVSS User Guide:

When scoring the impact of a vulnerability in a library, independent of any adopting program or implementation, the analyst will often be unable to take into account the ways in which the library might be used. While specific products using the library should generate CVSS scores specific to how they use the library, **scoring the library itself requires assumptions to be made. The analyst should score for the reasonable worst-case implementation scenario.**

- As such, this tends to skew the CVSS values for libraries higher than end-products.

Detour: Comparing CVSSv3 scores

- CVE-2023-24998 – DoS in Apache Commons FileUpload
 - Type: Library
 - Vector: CVSS:3.1/AV:N/AC:L/PR:**N**/UI:N/S:U/C:**N**/I:**N**/A:H
 - Base score: 7.5 (High)
- CVE-2023-0669 – Pre-Authentication RCE in product Fortra GoAnywhere MFT
 - Type: Product
 - Vector: CVSS:3.1/AV:N/AC:L/PR:**H**/UI:N/S:U/C:**H**/I:**H**/A:H
 - Base Score: 7.2 (High)

PR is “Privileges Required”; ‘N’ means ‘None’ and ‘H’ means ‘High’.

SBOMs: How They Can Help

- Software Bill of Materials (SBOM) mandated for federal government in Executive Order 14028, signed by President Biden on 5/17/2021.
 - Required to be in electronically accessible format.
- Can be used to construct ledger of all the “ingredients” (dependencies) of how software product is constructed.
 - Motivation: Assist in identifying which software artifacts are affected by a specific CVE.
- Help you pass required compliance checklists, e.g., FedRAMP

SBOM Promise: What We Expect Them To Do

“The idea behind such a thorough inventory is that companies can better track the nuts and bolts of their software—including whether it houses security vulnerabilities like the Log4j software flaw—and more quickly respond to them.”

— *The Wall Street Journal*, “AI Is Generating Security Risks Faster Than Companies Can Keep Up”, 2023-08-10,
<https://www.wsj.com/articles/ai-is-generating-security-risks-faster-than-companies-can-keep-up-a2bdedd4>

SCA & SBOMs: Where They Fall Short

- Lots of false positives:
 - Leads to wasted resources patching things that are not exploitable.
 - Leads to waivers that could come back to bite you.
- Some false negatives (example later).
 - Unsure how common this is. Needs research!
- Incomplete solution:
 - Tracks the affected software artifact but not (by itself) where the artifact is deployed.
 - Would like to know what servers and installation paths for efficient patching.


Approaches Analyzing CVEs: In Open Source

- Look what's been fixed:
 - Often specific commit IDs are mentioned in CVE notes or its references.
 - Or use 'git diff' across previous and patched release branches or tags
- Check if PoC exploit is available and test against your code, tweaking as needed.
- Follow call tree of affected components through your dependencies:
 - Lots of recursive greps.
 - Gradually go up the dependency tree.
 - When you get to top, you have list of potentially vulnerable components in your dependencies.
 - Then look for exploitable paths through these vulnerable components.

Example – Analyzing CVE-2023-24998 in ESAPI (1/2)

```
wallk@feynman:~/work/esapi-work/kww-2.5.2.0-release$ mvn dependency:tree | grep -v :test
```

```
...
[INFO] -----< org.owasp.esapi:esapi >-----
[INFO] Building ESAPI 2.5.3.0-SNAPSHOT
[INFO] -----[ jar ]-----
...
[INFO] --- maven-dependency-plugin:3.5.0:tree (default-cli) @ esapi ---
[INFO] org.owasp.esapi:esapi:jar:2.5.3.0-SNAPSHOT
[INFO] +- javax.servlet:javax.servlet-api:jar:3.1.0:provided
[INFO] +- javax.servlet.jsp:javax.servlet.jsp-api:jar:2.3.3:provided
[INFO] +- xom:xom:jar:1.3.8:compile
[INFO] +- commons-beanutils:commons-beanutils:jar:1.9.4:compile
[INFO] | +- commons-logging:commons-logging:jar:1.2:compile
[INFO] | \- commons-collections:commons-collections:jar:3.2.2:compile
[INFO] +- commons-configuration:commons-configuration:jar:1.10:compile
[INFO] +- commons-lang:commons-lang:jar:2.6:compile
[INFO] +- commons-fileupload:commons-fileupload:jar:1.5:compile
[INFO] +- org.apache.commons:commons-collections4:jar:4.4:compile
[INFO] +- org.apache-extras.beanshell:bsh:jar:2.0b6:compile
[INFO] +- org.owasp.antisamy:antisamy:jar:1.7.3:compile
[INFO] | +- org.htmlunit:neko-htmlunit:jar:3.1.0:compile
[INFO] | +- org.apache.httpcomponents.client5:httpclient5:jar:5.2.1:compile
[INFO] | | \- org.apache.httpcomponents.core5:httpcore5-h2:jar:5.2:compile
[INFO] | +- org.apache.httpcomponents.core5:httpcore5:jar:5.2.1:compile
[INFO] | +- org.apache.xmlgraphics:batik-css:jar:1.16:compile
[INFO] | | +- org.apache.xmlgraphics:batik-shared-resources:jar:1.16:compile
[INFO] | | +- org.apache.xmlgraphics:batik-util:jar:1.16:compile
[INFO] | | | +- org.apache.xmlgraphics:batik-constants:jar:1.16:compile
[INFO] | | | \- org.apache.xmlgraphics:batik-i18n:jar:1.16:compile
[INFO] | | \- org.apache.xmlgraphics:xmlgraphics-commons:jar:2.7:compile
[INFO] | +- xerces:xercesImpl:jar:2.12.2:compile
[INFO] | \- xml-apis:xml-apis-ext:jar:1.3.04:compile
[INFO] +- org.slf4j:slf4j-api:jar:2.0.6:compile
[INFO] +- xml-apis:xml-apis:jar:1.4.01:compile
[INFO] +- commons-io:commons-io:jar:2.11.0:compile
[INFO] +- com.github.spotbugs:spotbugs-annotations:jar:4.7.3:compile
[INFO] | \- com.google.code.findbugs:jsr305:jar:3.0.2:compile
[INFO] -----
[INFO] BUILD SUCCESS
```



Example – Analyzing CVE-2023-24998 in ESAPI (2/2)

- 1) 😐 - Oh bother. (Snyk tells me I need to upgrade FileUpload jar.)
- 2) 😄 - Woohoo! ESAPI doesn't call FileUploadBase.
- 3) 🤔 - Really? FileUploadBase is an *abstract* base class? Couldn't you enumerate *all* the affected classes instead of making me look?
- 4) 😡 - Wait? It's not enough to just upgrade? Grrr.
- 5) 🤬 - [After analyzing their fix] What, this doesn't fix the DoS, it just makes it slightly harder to exploit. F-bombs ensue.
 - Rather than limiting # of files uploaded per HTTP *session*, they just limited the maximum # uploaded per HTTP *request*.

Combating the SCA Noise: Prevent your SBOMs from Becoming F-bombs

- Noise in the small – the individual project's perspective.
 - For libraries, may require more than just updating to some version.
 - Difficult to squeeze into a 2-week sprint without dropping something else.
- Noise in the large (enterprise level squawking) – the blue team's perspective.
 - See a backlog of possibly thousands of unpatched libraries or products.

We need to do better! What will that require?

Gaps in SCA Tools

- Credibility gap: “The sky is falling”. Sowing FUD.
 - Overwhelmed with false positives.
 - Most from transitive dependencies, so very little control.
 - Most have their privately researched vulnerabilities they report as well.
- The false negative gap from partial remediation.
 - Can happen when a patch that is not “secure-by-default” is applied.

Example of False Negative in SCA Tools:

CVE-2023-24998

- From <https://nvd.nist.gov/vuln/detail/CVE-2023-24998>
 - **Current Description:** Apache Commons FileUpload before 1.5 does not limit the number of request parts to be processed resulting in the possibility of an attacker triggering a DoS with a malicious upload or series of uploads. Note that, like all of the file upload limits, the new configuration option (FileUploadBase#setFileCountMax) is not enabled by default and must be explicitly configured.
 - **Base Score:** 7.5 HIGH
 - **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Example of False Negative in SCA Tools:

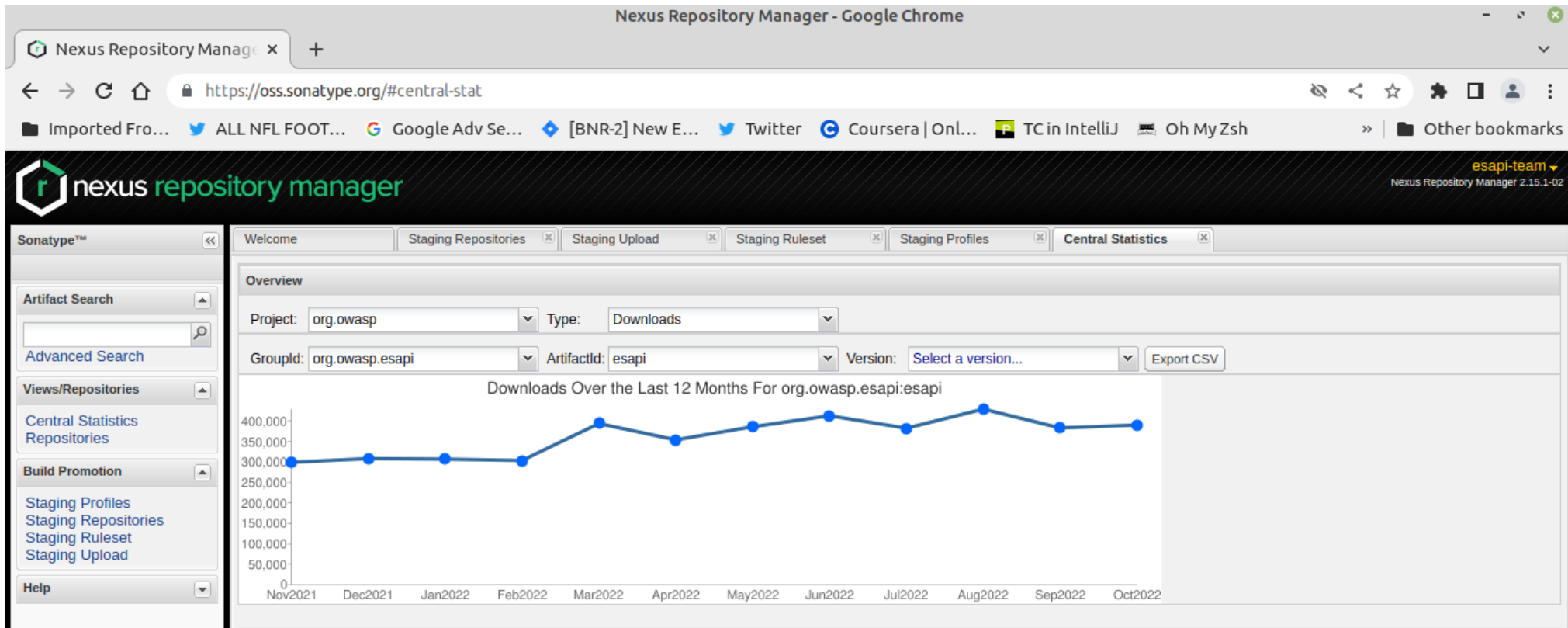
Highlighted CVE-2023-24998

- From <https://nvd.nist.gov/vuln/detail/CVE-2023-24998>
 - **Current Description:** Apache Commons FileUpload before 1.5 does not limit the number of request parts to be processed resulting in the possibility of an attacker triggering a DoS with a malicious upload or series of uploads. Note that, like all of the file upload limits, the new configuration option (FileUploadBase#setFileCountMax) is ***not enabled by default*** and must be explicitly configured.
 - **Base Score:** 7.5 HIGH
 - **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

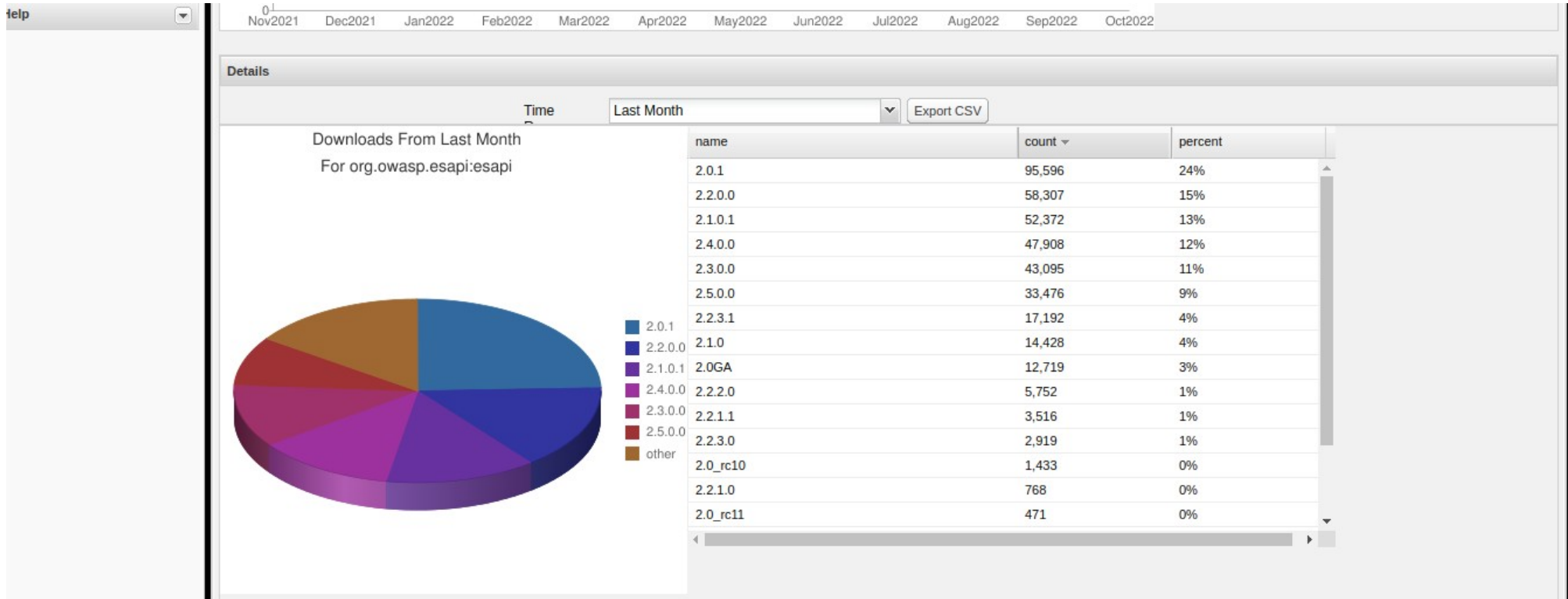
Suggestions and Takeaways: What Can We Do As Developers?

- Lazy vs. diligent approaches.
- Take pride in doing things right.
 - Be precise in your CVE descriptions.
 - Include *all* the affected classes, not just the base-level (i.e., super) classes.
 - ***Remediation should be secure-by-default.***
 - If not possible because of backwards compatibility issues, then at least try to show a very prominent notice and/or provide a test for developers to confirm they've remediated it correctly.
 - Breaks OS level patching of libraries if not secure-by-default.
- Partner with an AppSec engineer or security researcher.

Evidence That Many Are Not Patching (1/2)



Evidence That Many Are Not Patching (2/2)



AppSec Engineer Perspective

- Take responsibility.
- Less hype: Help the community first, not yourself.
- Help developers who need to remediate; don't worry about divulging information to attackers.
- Eat our own dog food – Don't treat vulnerabilities in your software differently.
- Suggest workarounds (e.g., when is it safe to exclude a transitive dependency, is there some data validation I can perform that might make some function call safe, etc.?).

Aside: Possible Alternatives to CVSS Base Scores

- Consider including Temporal and Environmental components to CVSS rather than only relying on Base scores.
 - Likely need help from SCA tool vendors to support this.
- CISA's Known Exploited Vulnerabilities (KEV) - <https://www.cisa.gov/known-exploited-vulnerabilities>
- FIRST's Exploit Prediction Scoring System (EPSS) - <https://www.first.org/epss/>

Suggestions and Takeaways: What Can We Do As DevOps / DevSecOps?

- Incorporate your SCA scans into your CI/CD pipelines.
 - Especially important for QA environment as you need the heads up.
 - Consider *dynamic* SCA scans to focus on the loaded / used libraries.
- If CVE is not in CISA's KEV Catalog, consider weighting CVSS scores from NVD.
 - Weight 1st level (direct) dependencies the most, then 2nd level transitive dependencies, the 3rd level transitive dependencies, etc., *OR*
 - Consider something like Exploit Prediction Scoring System (EPSS).
- Stop blocking releases that only patch *some* known vulnerabilities.
 - Alternative is no patches at all.

Suggestions and Takeaways: What Can We Do As SCA Tool Providers?

- Stop using your private DB vulnerabilities that have not been publicly and externally vetted unless you only wish to consider them Informational.
 - Is your goal to make sales or help secure the world? FUD does not help secure the world.
- When NVD reports a vulnerability as “withdrawn by the CNA” **stop** reporting it! NVD and the issuing CNA have more context than you. If you insist on still reporting it, do so with appropriate notice or mark it as Informational.

Suggestions and Takeaways: What Can NIST and CNAs Do?

- Distinguish between end products vs reusable components.
 - Libraries need an alternative to CVSS scoring, or consider a multiple values (average vs worst case scenarios). Come up with something!
- NIST should provide justification when overriding CNA recommendations.
- Consider weighting CISA's Known Exploited Vulnerabilities (KEV) Catalog as an aspect of severity scoring.
 - Scores would automatically adjust over time. Could even go down.
- Insist on *precise* descriptions of the attack surface in the Details section.
 - Ideally, standardized machine parsable (e.g., XML or JSON) format.
 - Must mention all affected functions, methods, and/or classes. Be specific!
- Special treatment of patches that are not secure-by-default (for the given CVE).

References

- Jeff Williams; “What about Transitive Dependencies?” LinkedIn post (2023-07-27) - https://www.linkedin.com/posts/planetlevel_what-about-transitive-dependencies-i-hear-activity-7090447433146515456-n4GW
- Question “Can I change a component on Central?”, <https://web.archive.org/web/20201112013316/https://central.sonatype.org/articles/2014/Feb/06/can-i-change-a-component-on-central/> (posted 2014-02-06, retrieved 2023-09-02).
- GitHub Security Advisory: DoS vulnerabilities persist in ESAPI file uploads despite remediation of CVE-2023-24998, <https://github.com/ESAPI/esapi-java-legacy/security/advisories/GHSA-7c2q-5qmr-v76q>

Questions?

- Ask me now, or drop me an email at kevin.w.wall@gmail.com
- Or, if you don't mind waiting a bit, DM me as [@KevinWWall](#) on OWASP Slack or Twitter (although the latter may take a while).

NOTE: Slide deck, with speaker's notes, available at:

[https://github.com/kwwall/presentations/tree/master/
SBOMs_to_F-Bombs](https://github.com/kwwall/presentations/tree/master/SBOMs_to_F-Bombs)



OWASP 2023
GLOBAL
AppSec

WASHINGTON

DC

OCT 30 • NOV 3

THANK YOU

