

Learn from Log4shell

Using SBOMs for Zero-Day Preparedness



Hello World



Paul Novarese
Principal Solutions Architect
Anchore, Inc.
pvn@anchore.com - @pvn
@pvn@mas.to

Agenda

00

Supply Chain Attacks

01

Log4Shell – What Actually Happened

02

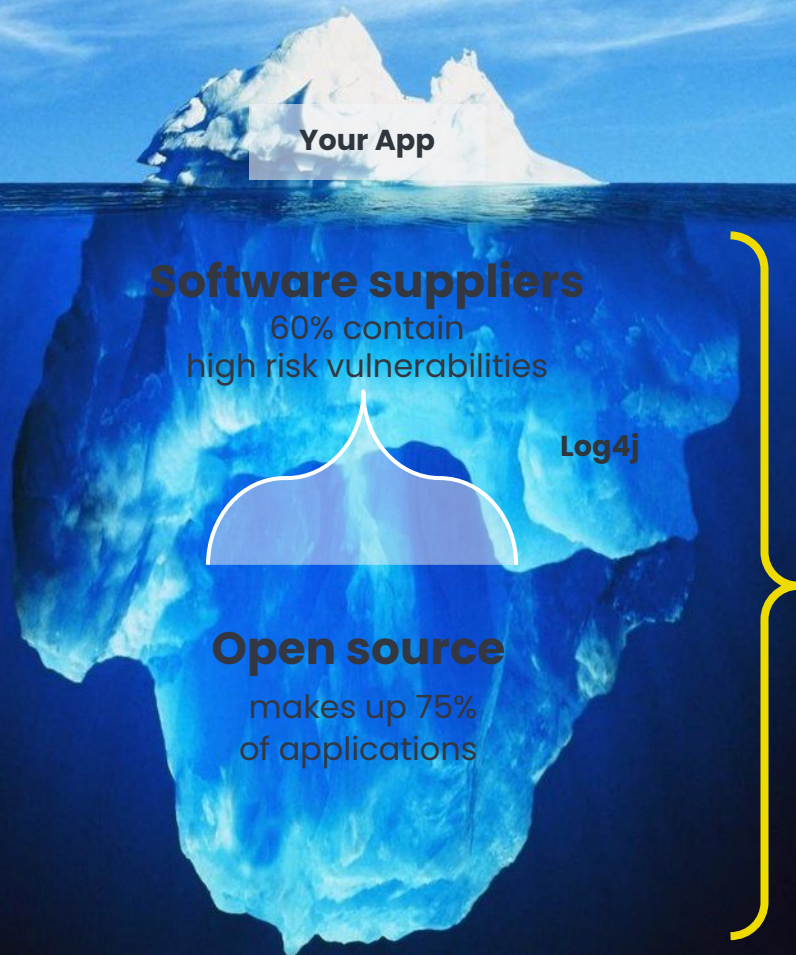
Software Bill of Materials

03

Now what?

What are Supply Chain Attacks?

Risk in the Software Supply Chain



Popular Cryptocurrency Exchange dYdX Hacked Its NPM Account

Application Security

🕒 4 MIN READ 📰 NEWS

Supply Chain Attack Pushes Out Malware to More than 250 Media Websites

TA569 has modified the JavaScript of a legitimate content and advertising engine used by news affiliates, in order to spread the FakeUpdates initial access framework.

W4SP Stealer Stings Python Developers in Supply Chain Attack

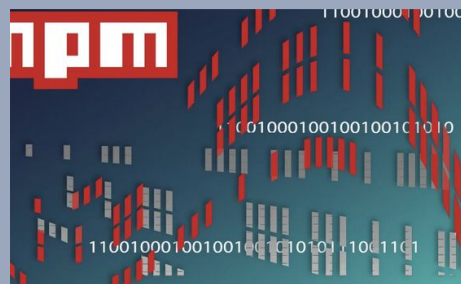
Threat actors continue to push malicious Python packages to the popular PyPI service, striking with typosquatting, authentic sounding file names, and hidden imports to fool developers and steal their information.



Robert Lemos

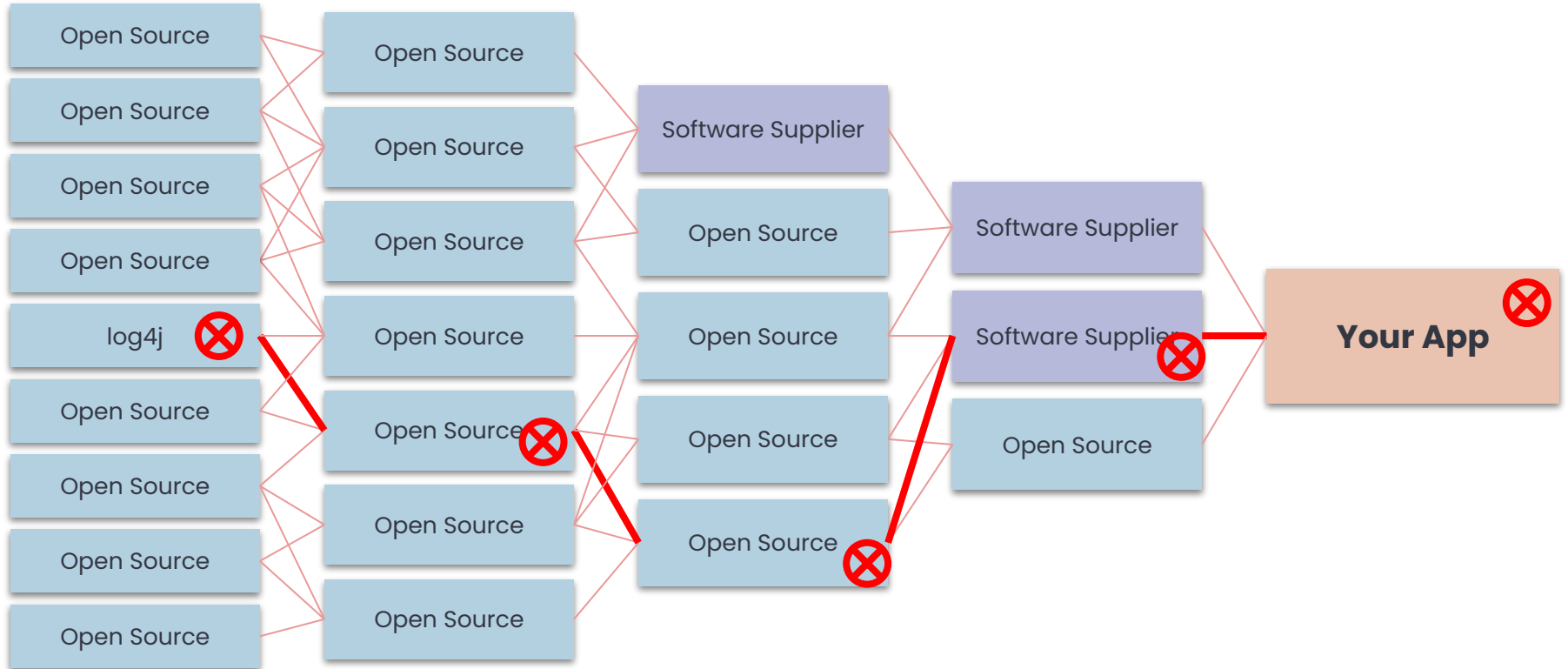
Contributing Writer, Dark Reading

November 03, 2022

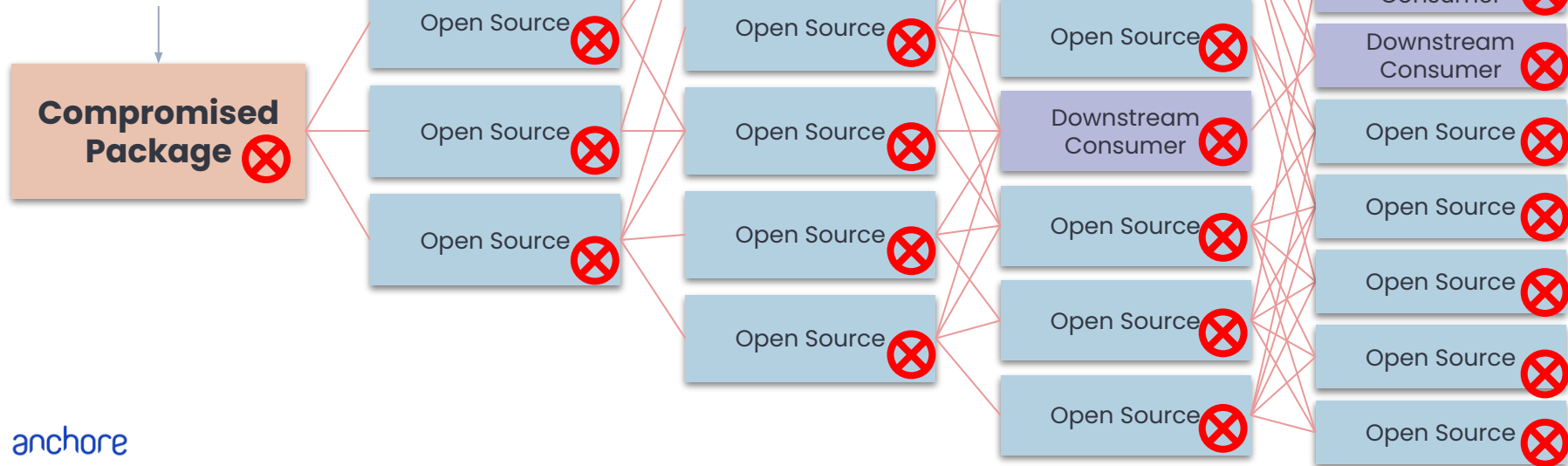


November 04, 2022

Software Supply Chain: The Iceberg Funnel

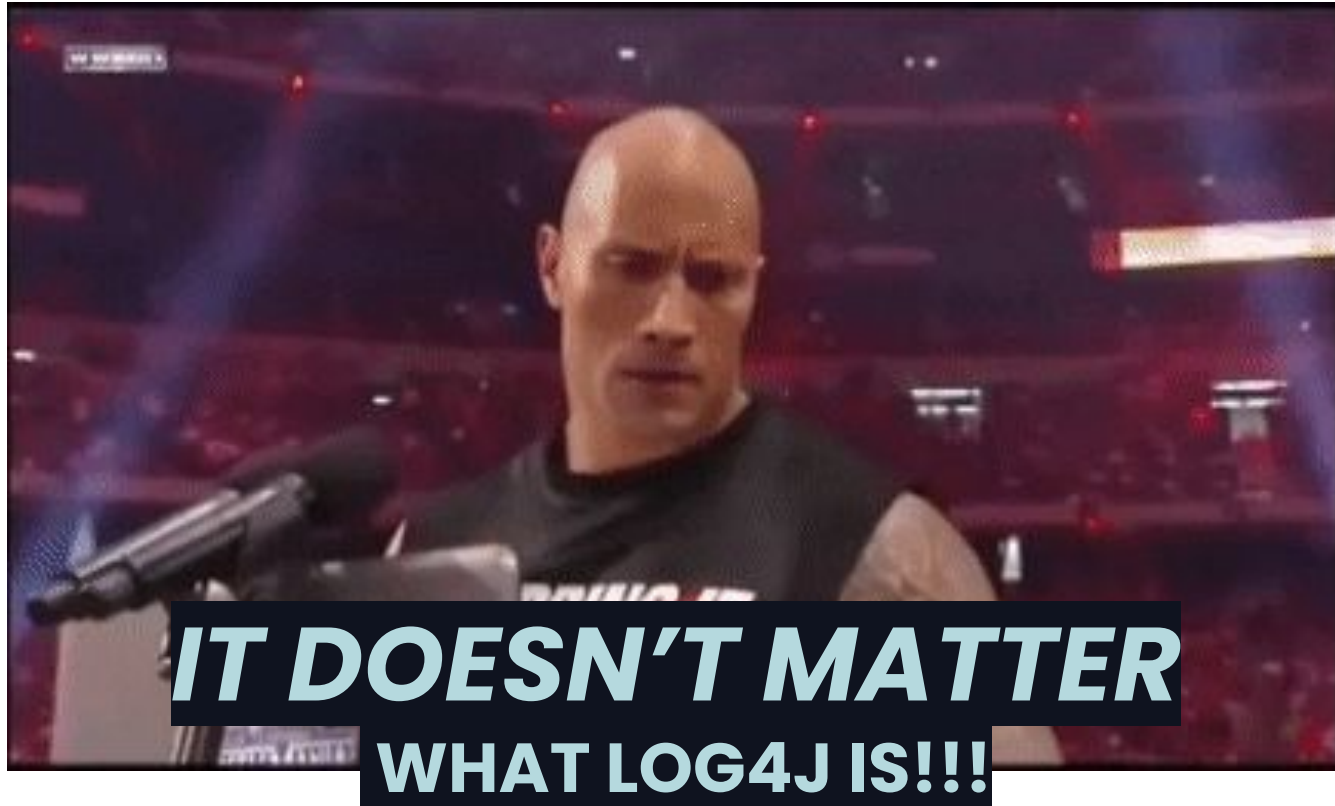


The Reverse Funnel



Log4Shell Impact

What is log4j? What is log4shell?



log4j Timeline - Background

- 2001 - Initial release
- 14 Sep 2013 - Vulnerability is introduced 2.0-beta9
- 3 Aug 2016 - Potential exploit presented at Black Hat

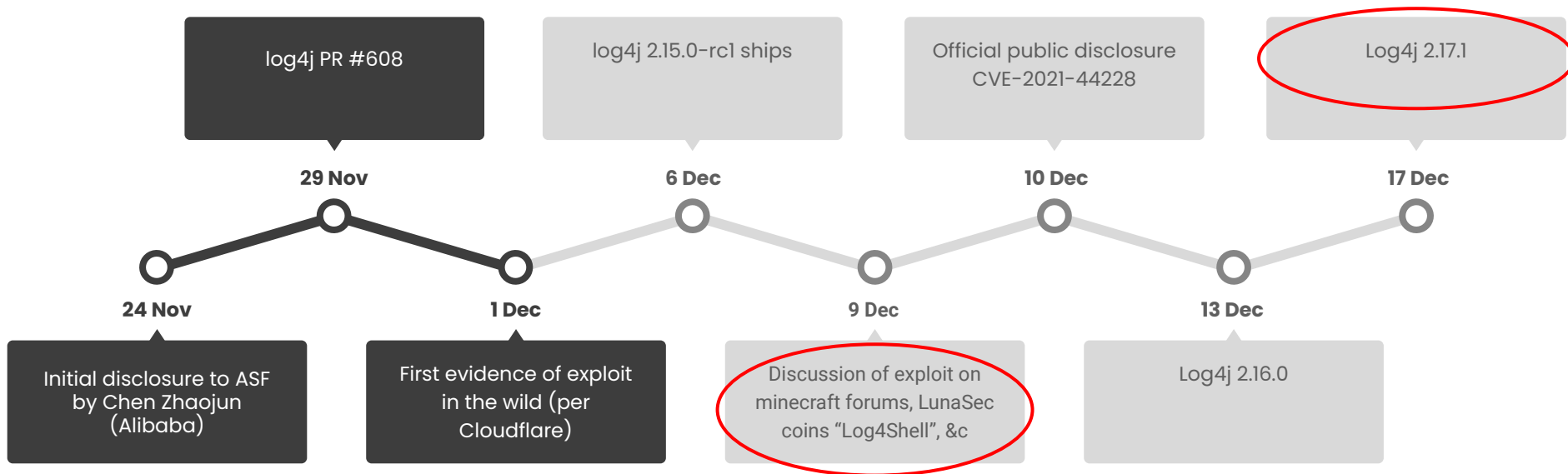


 **blackhat** USA 2016

BlackHat Sound Bytes

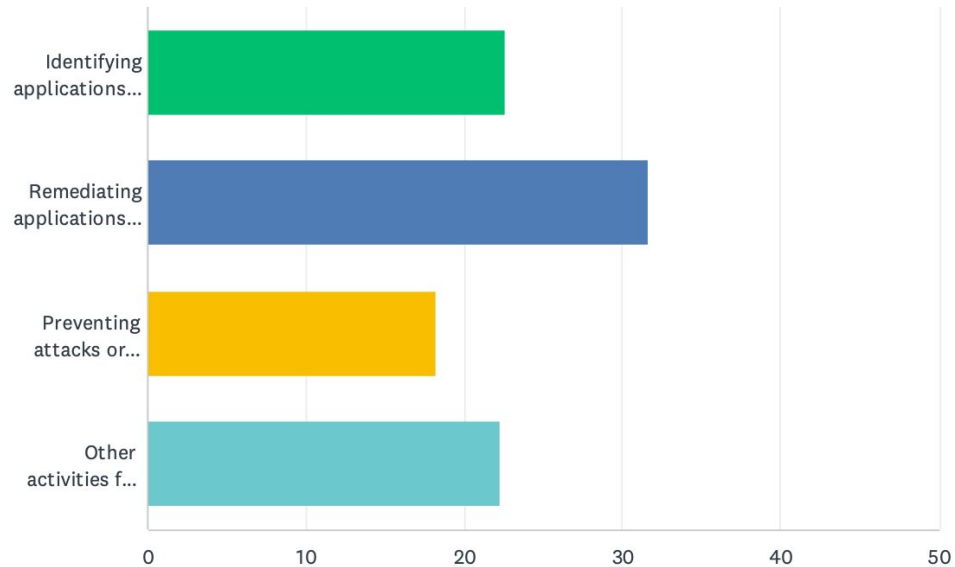
- Audit your Applications for two new vulnerability classes:
 - JNDI Injection
 - LDAP Entry Poisoning
- Carefully protect and periodically audit your LDAP backends; they contain the keys to your kingdom!

log4shell Timeline



Q12 Estimate how many hours you personally have spent to date on each of the following activities.

Answered: 195 Skipped: 15



What is a Software Bill of Materials?

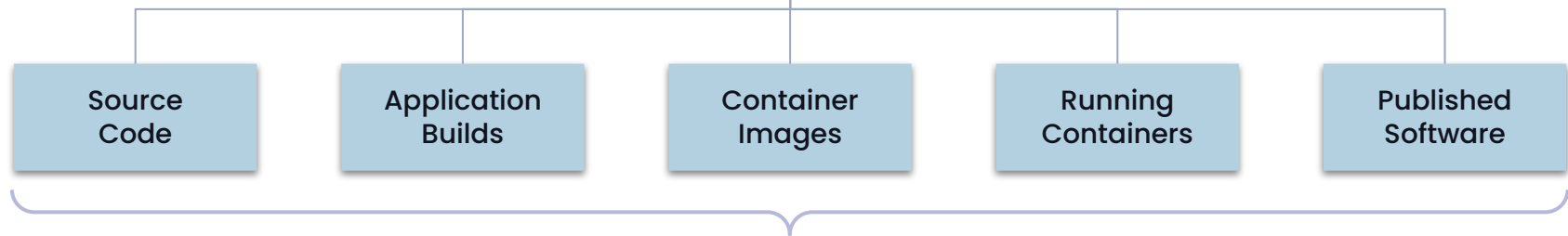
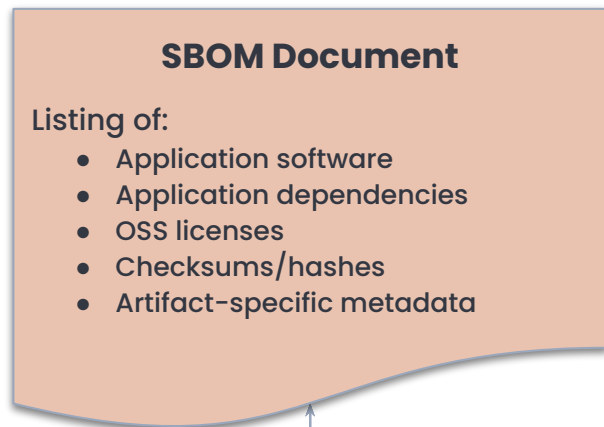
What is an SBOM?

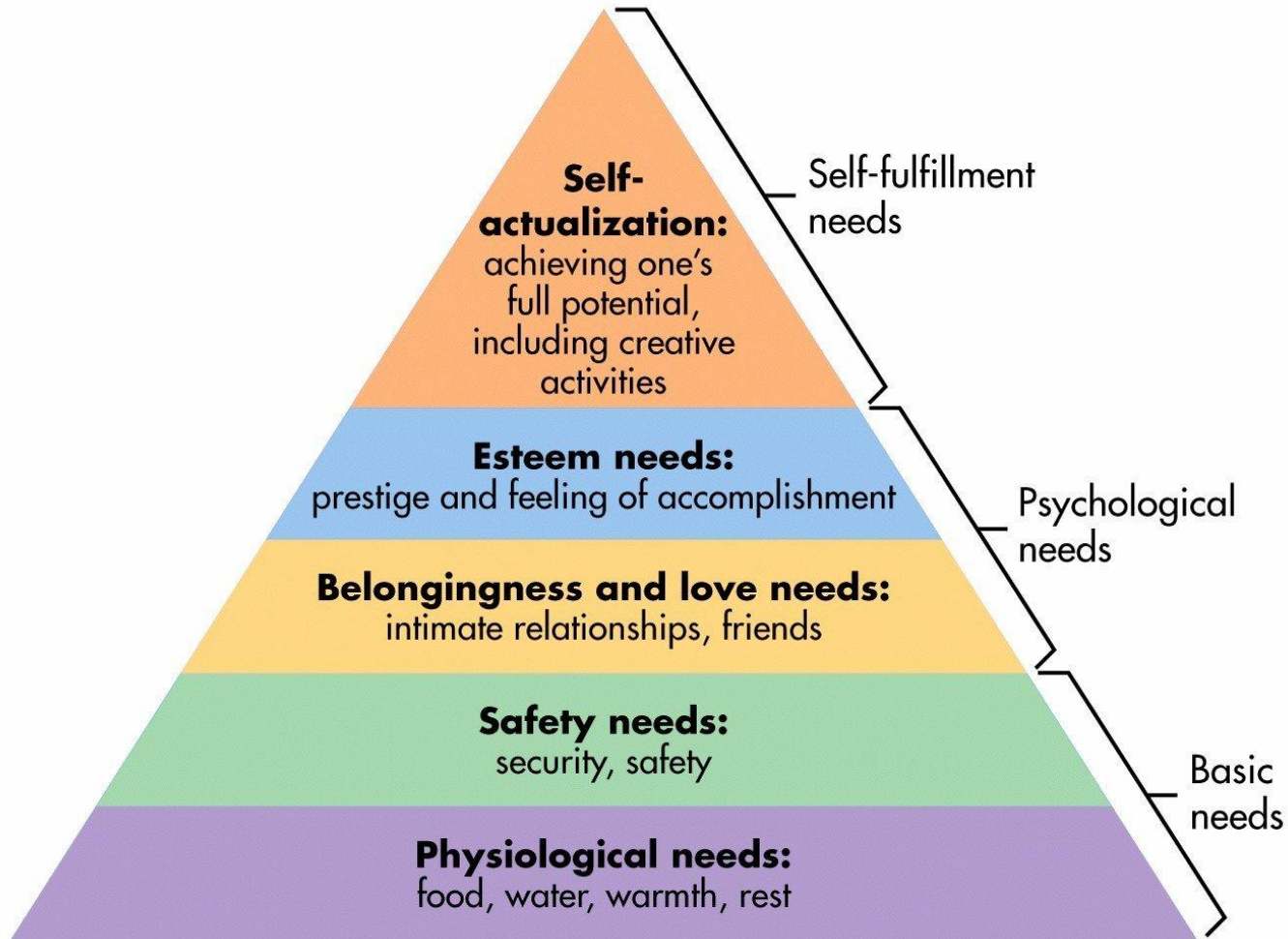


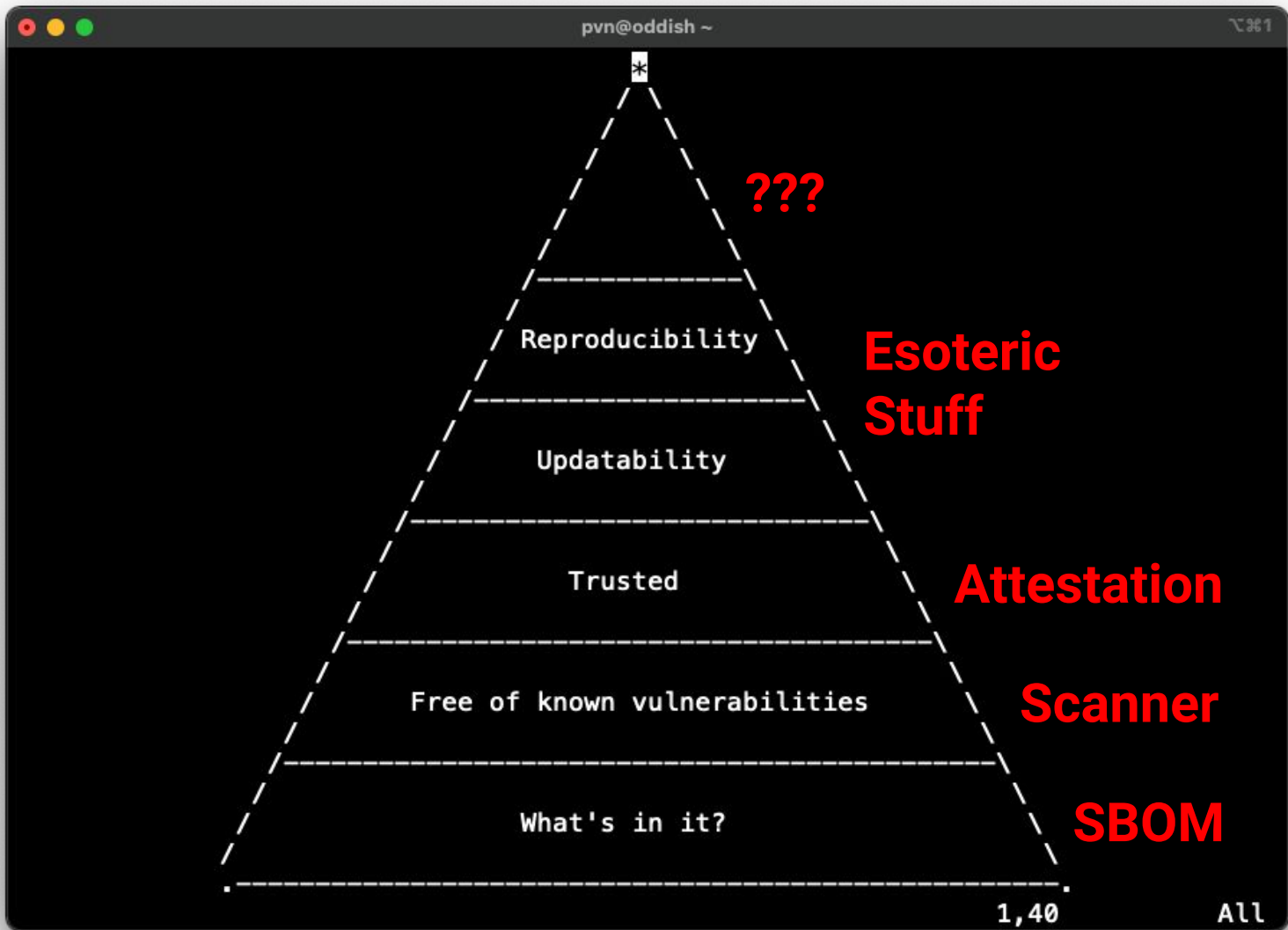
Nutrition Facts	
Serving Size 6 rolls (85g)	
Servings Per Container 2.5	
Amount Per Serving	
Calories 210	Calories from Fat 80
% Daily Value*	
Total Fat 9g	14%
Saturated Fat 2g	11%
Trans Fat 1.5g	
Cholesterol 10mg	3%
Sodium 390mg	16%
Total Carbohydrate 25g	8%
Dietary Fiber 2g	7%
Sugars 3g	
Protein 7g	
Vitamin A 8%	Vitamin A 2%
Calcium 4%	Iron 8%

DISTRIBUTED BY **General Mills Sales, Inc.**
GENERAL OFFICES, MINNEAPOLIS, MN 55440 USA
© 2005 General Mills Pat. Pend. CT LI 9440 3060328104

What is an SBOM?







Now What?

How will this improve my life?

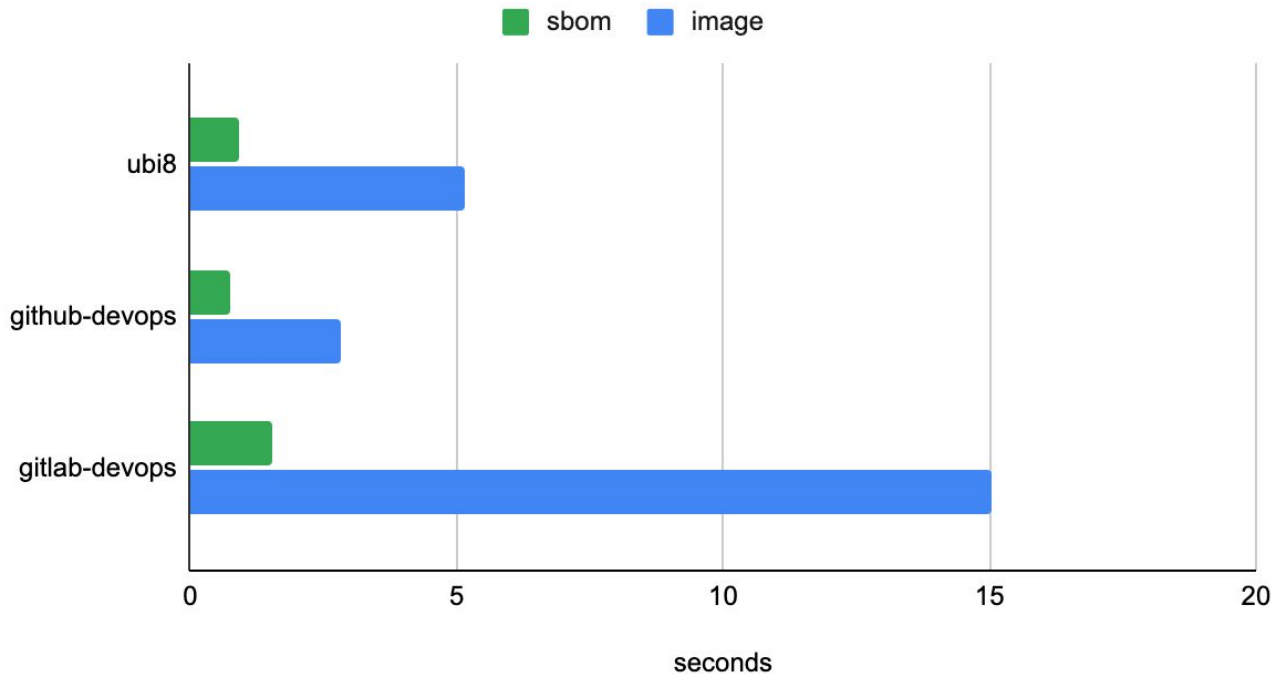
How Do SBOMs Actually Help?



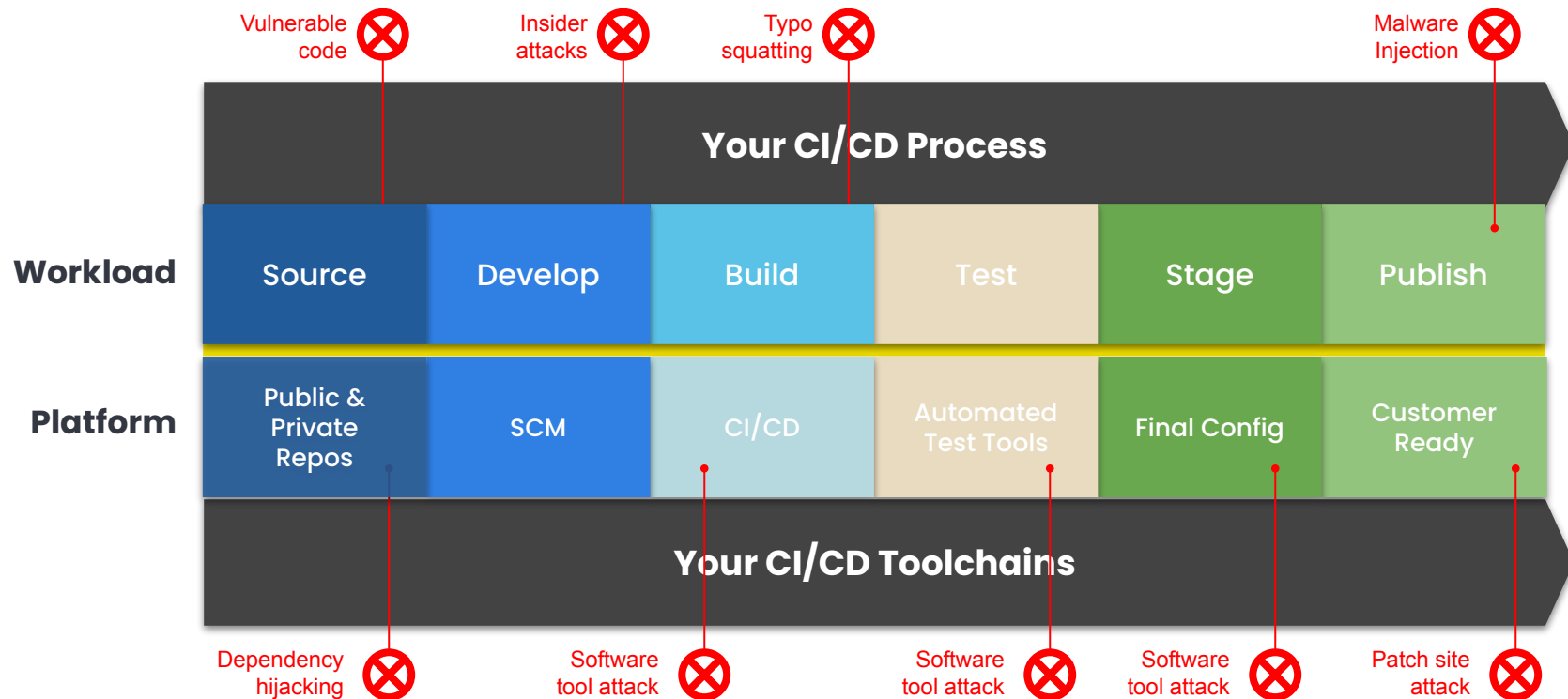
Grype Scan Timing

grype vulnerability check

	ubi8	github-devops	gitlab-devops
sbom	0.917	0.751	1.55
image	5.159	2.839	15.031



SBOMs Enable Continuous Evaluation



SBOM: Current State

Obscured
Dependencies

OpenSSL 3.x

???

Open Source

Node Base Image

dependencies

Your Code

Your App

Nutrition Facts	
Serving Size	100g
Amount Per Serving	
Calories	70
Total Fat 1g	
	2%
Total Fat 1g	2%
Saturated Fat 1.1g	4%
Trans Fat 0.1g	0%
Polyunsaturated Fat 2.1g	4%
Monounsaturated Fat 2.1g	4%
Cholesterol 0mg	0%
Sodium 0mg	0%
Total Carbohydrate 1g	2%
Dietary Fiber 0g	0%
Total Sugars 0g	0%
Includes 0g Added Sugars	0%
Protein 0g	0%

Imperfect
SBOM

Imperfect
Evaluation

SBOM: Better State

Identified
Dependencies

OpenSSL 3.x

dependencies

Open Source

Node Base Image

dependencies

Your Code

Nutrition Facts	
Serving Size (100g)	
Amount Per Serving	
Calories	70
Total Fat 1g	100%
Sodium 1g	0%
Total Cholesterol 1g	0%
Total Carbohydrate 1g	0%
Protein 1g	0%

Your App

Nutrition Facts	
Serving Size (100g)	
Amount Per Serving	
Calories	70
Total Fat 1g	100%
Sodium 1g	0%
Total Cholesterol 1g	0%
Total Carbohydrate 1g	0%
Protein 1g	0%

Improved SBOM

Improved
Evaluation



Paul V. Novarese

@pvn



When I tweak the demo 10 minutes before my presentation



8:41 PM · Aug 26, 2018 · Twitter for iPhone

```

jenkins@pvn.li:~ # find . -regex '.*sbom.json' | wc -l
167
jenkins@pvn.li:~ # time find . -regex '.*cyclonedx-sbom.json$' -exec jq -r '(.metadata.component.name) + " " +
    (.metadata.component.version) + " " + (.components[] | select (.name|contains("log4j"))) |
    select (.version < "2.15") | "\(.name) \(.version)"' {} \;
lab2:1 sha256:bee2f4c8c5065bfcc3ff623d969856b73997353272c8f0e9d55e3b5bba222601 log4j-core 2.14.1
lab4:1 sha256:0e2988c10f592adb4a8d85bc9433b8ef38ce1918e3895115c71bc3ef3b3c0890 log4j-core 2.14.1
lab1:1 sha256:6020b28cc409dc36152e5b9952e8f55ad8e5883291d7896ff237b4975ed38ad8 log4j-core 2.14.1
lab981:13 sha256:108ff602c6212812abbd82e0fa53aa697ed80f9fb2a625e6748c9f0fd9fdc17e log4j-core 2.14.1
lab25:1 sha256:f34262531caf2c6464de7a4aa0daf6afba7b3a2931cde9369ff98931165bca5 log4j-core 2.14.1
lab25:3 sha256:e15833697e1eb71363def7302c2e6da2120f974d88fa3d083a948fade0dcf42e log4j-api 2.14.1
lab25:3 sha256:e15833697e1eb71363def7302c2e6da2120f974d88fa3d083a948fade0dcf42e log4j-core 2.14.1
lab26:1 sha256:44248a0f622e8e7c89048f33613c4f080e3bd81f40a395b7dc72a0097c705691 log4j-core 2.14.1
lab26:3 sha256:ca6349408a468e6a4711af963c005c997d274a5a41d5d52779da1de44510eaa7 log4j-api 2.14.1
lab26:3 sha256:ca6349408a468e6a4711af963c005c997d274a5a41d5d52779da1de44510eaa7 log4j-core 2.14.1
lab29:1 sha256:359a16540c6f8933b6fd633a802055d10b1992606ade7cb695f3ac0ad0e60d23 log4j-core 2.14.1
lab29:3 sha256:50d30cfb2c7730b14e78882c1e80a5380daec44ceb7219b19685125ab0f0e0cd log4j-api 2.14.1
lab29:3 sha256:50d30cfb2c7730b14e78882c1e80a5380daec44ceb7219b19685125ab0f0e0cd log4j-core 2.14.1

real    0m3.661s
user    0m3.250s
sys      0m0.332s
jenkins@pvn.li:~ # time gype ./jobs/2022-devopsdays/jobs/lab29/builds/3/archive/lab29\3-syft-sbom.json
✓ Vulnerability DB [no update available]
✓ Scanned image [12 vulnerabilities]

```

NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
log4j-api	2.14.1		java-archive	CVE-2021-45105	Medium
log4j-api	2.14.1		java-archive	CVE-2021-44832	Medium
log4j-core	2.14.1	2.15.0	java-archive	GHSA-jfh8-c2jp-5v3q	Critical
log4j-core	2.14.1	2.17.0	java-archive	GHSA-p6xc-xr62-6r2g	High
log4j-core	2.14.1		java-archive	CVE-2021-44228	Critical
log4j-core	2.14.1		java-archive	CVE-2021-45046	Critical
log4j-core	2.14.1	2.16.0	java-archive	GHSA-7rjr-3q55-vv33	Critical
log4j-core	2.14.1	2.17.1	java-archive	GHSA-8489-44mv-ggj8	Medium
log4j-core	2.14.1		java-archive	CVE-2021-44832	Medium
log4j-core	2.14.1		java-archive	CVE-2021-45105	Medium

```

real    0m1.410s
user    0m0.932s
sys      0m0.054s
jenkins@pvn.li:~ #

```

DIY Demos and Labs

- **Create SBOMs, Find log4j, Integrate with Jenkins (Difficulty: Easy)**
 - <https://github.com/pvnovarese/2022-devopsdays>
 - Includes instructions on deploying a disposable Jenkins container
- **Additional Labs (Difficulty: Medium)**
 - <https://github.com/pvnovarese/2022-devopsworld>
 - Additional Labs
- **GitHub SBOM Action:**
 - <https://github.com/marketplace/actions/anchore-sbom-action>

Takeaways

00

SBOMs improve a lot of things but do not solve every problem you have

01

Don't assume the next zero-day will have a one-shot fix

02

SBOMs enable continuous, automated security/compliance checks

03

SBOMs reduce time spent identifying and remediating issues

References &c

Notes, Additional Reading, Bibliography

Log4j Bibliography &c

Dealing with log4shell (detection, mitigation, workarounds):

<https://cloudsecurityalliance.org/blog/2021/12/14/dealing-with-log4shell-aka-cve-2021-44228-aka-the-log4j-version-2/>

Keeping up with log4shell (post mortem)

<https://cloudsecurityalliance.org/blog/2021/12/16/keeping-up-with-log4shell-aka-cve-2021-44228-aka-the-log4j-version-2/>

Mysterious tweet hinting at the exploit:

<https://twitter.com/sirifu4k1/status/1468951859381485573>

Another mysterious tweet:

<https://twitter.com/CattusGlavo/status/1469010118163374089>

“THE” pull request:

<https://github.com/apache/logging-log4j2/pull/608>

Cloudflare digs for evidence of pre-disclosure exploits in the wild:

<https://twitter.com/eastdakota/status/1469800951351427073>

SBOM Reading List

Draft of upcoming site content for SBOM.me: <https://github.com/joshbressers/sbom-examples/blob/readme-update/site/index.md>

Making Better SBOMs: <https://kccncna2022.sched.com/event/182GT/> – <https://www.youtube.com/watch?v=earq775L4fc>

Announcing GUAC: <https://security.googleblog.com/2022/10/announcing-guac-great-pairing-with-slsa.html>

Reflections on Trusting Trust: https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf

Generate sboms with syft and jenkins: https://www.youtube.com/watch?v=nMLveJ_TxAs

Solar Winds post mortem: <https://www.lawfareblog.com/solarwinds-and-holiday-bear-campaign-case-study-classroom>

Profound Podcast – Episode 10 (John Willis and Josh Corman):
<https://www.buzzsprout.com/1758599/8761108-profound-dr-deming-episode-10-josh-corman-captain-america>

Creating a trusted container supply chain: <https://thenewstack.io/creating-a-trusted-container-supply-chain/>

Footnotes

Other notes:

Slide 6: <https://www.mend.io/resources/blog/popular-cryptocurrency-exchange-dydx-has-had-its-npm-account-hacked/>

Slide 6: <https://www.mend.io/resources/blog/cybercriminals-targeted-users-of-packages-with-a-total-of-1-5-billion-weekly-downloads-on-npm/>

Slide 6: <https://www.darkreading.com/threat-intelligence/w4sp-stealer-aims-to-sting-python-developers-in-supply-chain-attack>

Slide 6: <https://www.darkreading.com/application-security/supply-chain-attack-pushes-out-malware-to-more-than-250-media-websites>

Slide 11: <https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE.pdf>

Slide 12: <https://hypixel.net/threads/psa-there-is-a-fatal-remote-code-execution-exploit-in-minecraft-and-its-by-typing-in-chat.4703238/>

Slide 12: https://twitter.com/_r_netsec/status/1469120458083962882

Slide 12: <https://twitter.com/eastdakota/status/1469800951351427073>

Slides 16-17: Maslow's Hierarchy of Supply Chain Needs: <https://www.youtube.com/watch?v=rcP8QHFMwCw>

Images used for SBOM generation timing benchmarks:

- registry.access.redhat.com/ubi8:latest
- https://gitlab.com/pvn_test_images/devops-supply-chain
- <https://github.com/pvnovarese/devops-supply-chain-demo>

Integration of cosign with syft: <https://github.com/anchore/syft/issues/510>

Add support for Hints in syft: <https://github.com/anchore/syft/issues/31>

Best Practices for Securing the Software Supply Chain

00

Centralized, secure CI/CD process for all software

01

Build images from trusted sources

02

Automate security testing and policy enforcement

03

Deploy only trusted images into production

Q&A

Download Syft

<https://github.com/anchore/syft>

Download Grype

<https://github.com/anchore/grype>

Let us know if you like it by giving us a star on GitHub

Get an invite to our open source community Slack at
<https://anchore.com/slack/>

These slides and lab examples archived here:

<https://github.com/pvnovarese/2022-devopsdays>

Appendix A

SBOM Formats

Existing SBOM formats

	SPDX "Software Package Data eXchange"	CycloneDX	SWID "Software ID"
<i>Organization</i>	SPDX Workgroup (~20 orgs) under the Linux Foundation	A "meritocratic, consensus-based community project" with a Industry Working Group	ISO/IEC Joint Technical Committee Trusted Computing Group
<i>Initial Draft</i>	2010	2017	2009
<i>Formats</i>	RDF, XLS, SPDX, YAML, JSON	XML, JSON	XML, CBOR (CoSWID only)
<i>Spec</i>	spdx.github.io/spdx-spec BS ISO/IEC 5962 – 2020 Draft	github.com/CycloneDX/specification	iso.org/standard/65666.html ISO/IEC 19770-2:2015

Existing SBOM formats: Use Cases

	SPDX "Software Package Data eXchange"	CycloneDX	SWID "Software ID"
<i>Original use cases</i>	License management	For use with OWASP Dependency-Track	Inventory and change tracking
<i>Unique Features</i>	Extensive support for expressing license details	Extensible format and integrates SPDX license IDs, pURL, and other external identifiers	Deeply integrated into the build and publishing process for a software component
<i>Use cases of latest format versions</i>	<ul style="list-style-type: none">• Tracking attributes of multiple software components (e.g. vendor, license, version, etc.)• Generically describe packages, containers, os distributions, archives, etc• Integrity verification of software components and sub-components		

A “good” SBOM describes...

What is being catalogued

For example a running system, a machine image, a container image, etc.

Each item uniquely

Such as each component name, version, UUID, and relationships to other components.

What did the cataloguing

The tool that generated the document with its configuration.

A “great” SBOM also includes...

In scope and out of scope

For example “only these paths were searched” or “only JARS and RPMs are being search for”.

Exceptional conditions

Such as warnings or errors that occur during processing or missing environmental factors

Additional metadata

Such as Java pom properties, key-values, additional RPM DB tag entries, and licenses.

Introducing Syft

- Syft is an **open source tool that generates SBOMs** from container images and filesystems
- Syft supports **many package ecosystems**:
 - APK, DEB, RPM, Ruby Bundle, Python Wheel/Egg/requirements.txt, JavaScript NPM/Yarn, Java JAR/EAR/WAR, Jenkins plugin JPI/HPI, Go modules, Rust Crate
- Syft also supports **multiple output formats**
 - Syft-Native
 - CycloneDX
 - [SPDX](#)

Appendix B

Additional Bonus Slides



Paul Novarese

Software Supply Chain Security at Anchore

9mo · Edited



The **#log4j** debacle is going to have ramifications far beyond the vulnerability itself. There has been a lot of inertia in how issues are evaluated and classified, how information about those issues is disseminated, and how organizations respond to them, and **#log4shell** has exposed a lot of these problems. This will be a catalyst for a lot of changes that are way overdue.



April King



@CubicleApril



The fact that there are almost 10,000 CVEs with the same CVSS score as the Log4j vulnerability suggests to me that maybe the scale should be logarithmic.

6:26 PM · Dec 11, 2021 · Twitter for iPhone

71 Retweets

6 Quote Tweets

736 Likes



Paul Novarese

Software Support & Main Security at Anchore

9mo · Edited

The [#log4j](#) debacle is going to have ramifications far beyond the vulnerability itself. There has been a long inertia in how issues are evaluated and classified, how information about those issues is disseminated, and how organizations respond to them, and [#CVEshell](#) has exposed a lot of these problems. This will be a catalyst for a lot of changes the way

we

King

bicycle

The fact that there are 1000 CVEs with the same [CVSS](#) score as the Log4j vulnerability suggests to me that maybe the scale should be logarithmic

6:26 PM · Dec 10, 2021 · Twitter for iPhone

71 Retweets

6 Quote

736 Likes

WRONG



Jake Williams
@MalwareJake



Hey infosec: remember that your job is risk reduction, not risk elimination. There's a BIG difference.

9:31 PM · Aug 29, 2021 · Twitter for Android

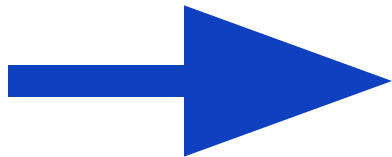
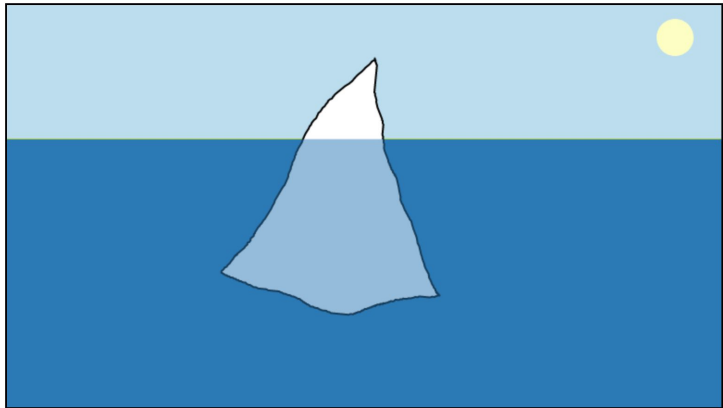
258 Retweets **26** Quote Tweets **1,677** Likes

Quick Aside

Iceberger

Draw an iceberg and see how it will float.

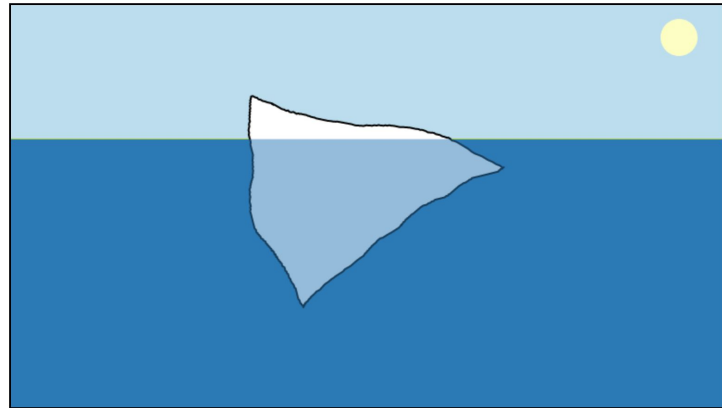
(Inspired by a [tweet by @GlacialMeg](#))



Iceberger

Draw an iceberg and see how it will float.

(Inspired by a [tweet by @GlacialMeg](#))





Dan Kaminsky 

@dakami



Amateurs think about vulnerabilities, professionals think about vectors.

4:04 PM · Aug 12, 2017 · [Twitter Web Client](#)

One Last Bad Analogy

