



# Introduction to Software Supply Chain Security

Laurie Williams



# Digital innovation depends on third-party software



Innovation, competitive advantage



Sonatype finds 747% average increase in open source supply chain attacks over the last three years.

# Oops! Accidental dependency vulnerability

CBS NEWS

NEWS ▾ SHOWS ▾ • LIVE ▾ LOCAL ▾ 🔍 Login

## Nightmare before Christmas: What to know about the Log4j vulnerability

BY NICOLE SGANGA  
UPDATED ON: DECEMBER 17, 2021 / 12:44 PM / CBS NEWS

f t



# Code dependencies as an attack vector

## Code dependencies as a weapon



FORTUNE

RANKINGS ▾ MAGAZINE NEWSLETTERS PODCASTS MORE ▾

SEARCH SIGN IN

Subscribe Now

Most Popular



Russia's ruble has almost totally recovered. Does that mean sanctions aren't working?



PAID CONTENT  
Change your marketer experience and unleash growth for your business  
FROM OPTIMIZELY



Study finds ivermectin, the horse drug Joe Rogan championed as a COVID treatment, does nothing to cure the virus



Binance's founder, who accumulated as much wealth as Mark Zuckerberg in a quarter the time, explains how it feels to become unfathomably rich virtually overnight

INTERNATIONAL • UKRAINE INVASION

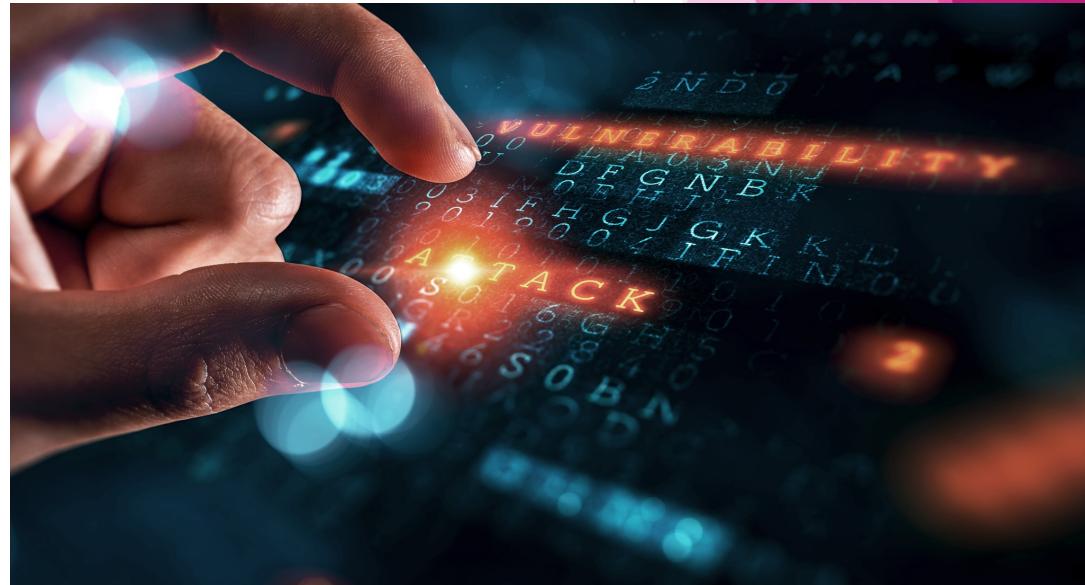
## Russia's largest bank tells its clients to delay downloading software updates after 'protestware' attacks target Russian users

BY NICHOLAS GORDON

March 22, 2022 7:07 AM EDT

node-ipc 

11.1.0 • Public • Published 24 days ago



# Build infrastructure as an attack vector



CNN politics The Biden Presidency Facts First US Elections

• TV CHANNELS CNN+

## Russian hackers behind SolarWinds hack are trying to infiltrate US and European government networks



By Sean Lyngaas, CNN

Updated 3:27 PM ET, Wed October 6, 2021



DIVE BRIEF

## Codecov hack – likened to SolarWinds – targets software supply chain

Published April 23, 2021 • Updated April 30, 2021



# Large Language Models (LLMs) as an attack vector



# Supply Chain as an (inter)national priority

## Executive Order on Improving the Nation's Cybersecurity

MAY 12, 2021 • PRESIDENTIAL ACTIONS



*(Section 4e) Within 90 days of publication of the preliminary guidelines pursuant to subsection (c) of this section, the Secretary of Commerce acting through the Director of NIST, in consultation with the heads of such agencies as the Director of NIST deems appropriate, shall issue guidance identifying practices that enhance the security of the software supply chain. Such guidance may incorporate the guidelines published pursuant to subsections (c) and (i) of this section.*



Software Bill of Materials (SBOM) Generation  
Self-Attestation of Secure Development Practices

# Proposed European Union's (EU's) Cyber Resilience Act



EN English

## Shaping Europe's digital future

[Home](#) | [Policies](#) | [Activities](#) | [News](#) | [Library](#) | [Funding](#) | [Calendar](#) | [Consultations](#)

[Home](#) > [Library](#) > [Cyber Resilience Act](#)

POLICY AND LEGISLATION | Publication 15 September 2022

## Cyber Resilience Act

The proposal for a regulation on cybersecurity requirements for products with digital elements, known as the Cyber Resilience Act, bolsters cybersecurity rules to ensure more secure hardware and software products.

## EU Cyber Resilience Act



For safer & more secure  
digital products

#DigitalEU #CyberSecEU

But, what “should” we do?  
And, what’s everyone else doing?



# Doing secure software supply chain science: an empirical study

7 companies

“early adopter/  
progressive/leader”

43 interviews  
[more to come]  
~1.5 hours/each



**BSIMM 13**

# Six Secure Software Supply Chain Summits (~60 people)

## **Top Five Challenges in Software Supply Chain Security: Observations From 30 Industry and Government Organizations**

William Enck and Laurie Williams | North Carolina State University

### **Trusting Trust: Humans in the Software Supply Chain Loop**



“The Executive Order is forcing industry to adopt security practice that should have been adopted 20 years ago. We want to actually be more secure, not just comply.”

# Chatham House Rules and other non-disclosures

I could tell you, but then  
I'd have to kill you.



# What "should" we DO about software supply chain security?

**NIST Special Publication 800-218**

## **Secure Software Development Framework (SSDF) Version 1.1:**

*Recommendations for Mitigating  
the Risk of Software Vulnerabilities*



**NIST Special Publication  
NIST SP 800-161r1**

## **Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations**

And also ...



# Software Supply Chain Best Practices



# ENISA Cybersecurity ICT/OT Supply Chain Risk Management Cycle

- Provide secure product and services
- Have the infrastructure protected
- Have secure processes in place
- Have technical measures implemented
- Create transparency in ICT/OT supply chain
- Measure the quality of products and services

- Manage vulnerabilities
- Know your assets
- Understand risks of vulnerabilities
- Monitor vulnerabilities
- Patch vulnerabilities
- Have a defined maintenance policy



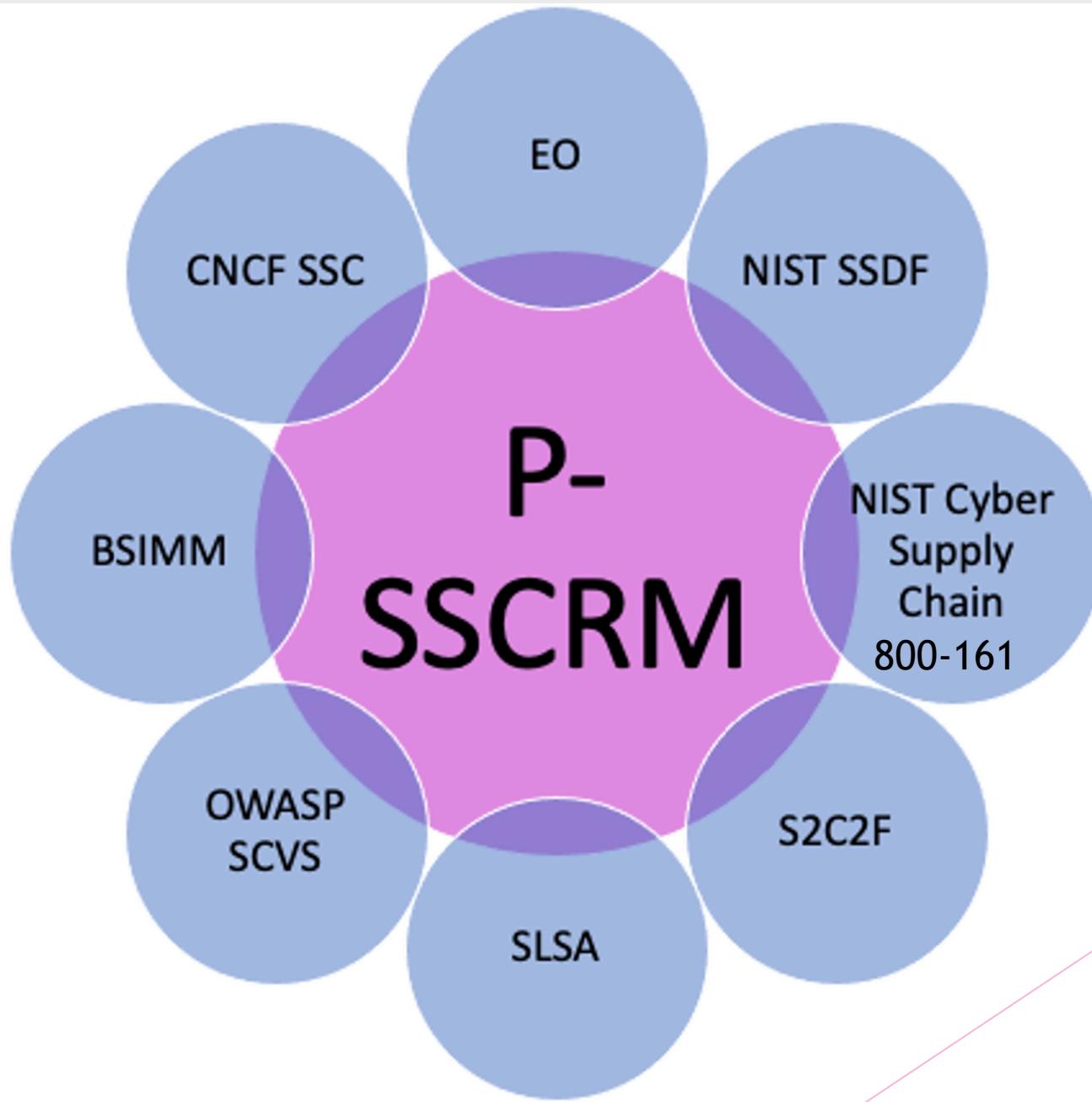
- Understand the supply chain
- Identify suppliers and providers
- Understand the potential risks for the own organisation and for end customer

- Manage ICT/OT supply chain
- Have policies and agreements in place
- Have cybersecurity requirements defined
- Monitor supplier and service provider performance
- Manage changes

## Vision: Proactive Software Supply Chain Risk Management (P-SSCRM) framework

P-SSCRM is a holistic framework that industry uses to **proactively mitigate** software supply chain risk through guided adoption of tasks; and that supports **assessment, scoring, and comparison** against industry peers, standards, and guidelines.

# P-SSCRM: The union of the frameworks



# Layout of P-SSCRM (v0.3)

J	L	M	N	S
Task Name	Objective	Definition	Question(s)	References
G.1.1 Organizational security requirements and policies	Organizational security requirements, such as those imposed by standards and regulations, are included in the SDLC.	Identify, document, communicate, and maintain security requirements and policies for the organization's software development infrastructure and secure SDLC. Maintain the requirements and policies over time. Incorporate constraints imposed by standards and regulations and customer-driven security requirements.	Do you have a defined secure SDLC that the engineers are aware of? Do you define security requirements and policies for the organization, its development infrastructure, contributions, and processes? How are these requirements and contributions maintained over time? Are constraints imposed by regulatory and compliance drivers included in these requirements, policies, and the SDLC?	<b>EO:</b> 4e(ix) <b>SSDF:</b> PO.1.1 <b>BSIMM:</b> CP1.1, CP1.2, CP1.3, SR1.1, SR2.2, SR3.3 <b>800-161:</b> SA-15 <b>CNCF SSC:</b> C: Establish and adhere to contribution policies <b>Self-attestation:</b> 2
G.1.2 Software licenses	Software licenses that conflict with the organization's objectives are identified.	Software licenses may or may not allow certain types of usage, contain distribution requirements or limitations, or require specific action if the software is modified. Risk is increased if the licenses of components are in conflict with an organization's objectives. Software licenses should be documented and tracked to enable tracing the users and use of licenses to access control information and processes according to software usage restrictions. License metadata should be recorded during build and made available in the SBOM.	Do you scan software to check if the license is in compliance with an organization's use policies? Is the process automated? Do you document and track users and uses of software licenses relative to access control policies and software usage restrictions?	<b>800-161:</b> CM-10 <b>OWASP SCVS:</b> 5.12 <b>S2C2F:</b> SCA-2 <b>CNCF SSC:</b> AU: Scan software for license implications
	Produce evidence of the use	Configure tools to generate artifacts to create an audit trail of the use of secure software development practices in a manner that conforms with record retention requirements and preserves the integrity of the findings and the confidentiality of the information. Assign responsibility for creating artifacts that tools cannot generate. Attestation should be immutable and published in the source repository releases, in the package registry, or elsewhere with their existence in a	Is the toolchain configured such that artifacts that attest to using secure development practices and other auditable are recorded consistent with retention requirements? Is responsibility assigned for creating needed artifacts that tools cannot generate? Do you use a framework, like in-toto, to produce authenticated meta-data about artifacts such as for attestation? Do you need to provide self-attestation for your product? Is the attestation immutable and published in the source repository releases, in the package registry, or elsewhere with their	<b>EO:</b> 4e(i)(F), 4e(ii), 4e(v) <b>SSDF:</b> PO.3.3 <b>BSIMM:</b> SM1.4, SR1.3 <b>800-161:</b> SA-15, AU-2, AU-3, AU-12 <b>SLSA:</b> Distributing attestation

# Mapping of “all the things” to “all the things”

G.1.1 Org security requirements	<p><b>EO:</b> 4e(ix) <b>SSDF:</b> PO.1.1 <b>BSIMM:</b> CP1.1, CP1.2, CP1.3, SR1.1, SR2.2, SR3.3 <b>800-161:</b> SA-15 <b>CNCF SSC:</b> C: Establish and adhere to contribution policies <b>Self-attestation:</b> 2</p>
G.1.2 Software licenses	<p><b>800-161:</b> CM-10 <b>OWASP SCVS:</b> 5.12 <b>S2C2F:</b> SCA-2 <b>CNCF SSC:</b> AU: Scan software for license implications</p>
G.1.3 Attestation	<p><b>EO:</b> 4e(i)(F), 4e(ii), 4e(v) <b>SSDF:</b> PO.3.3 <b>BSIMM:</b> SM1.4, SR1.3 <b>800-161:</b> SA-15, AU-2, AU-3, AU-12 <b>SLSA:</b> Distributing attestation <b>Self-attestation:</b> 1f</p>
G.1.4 Deliver provenance	<p><b>EO:</b> 4e(vi), 4e(vii), 4e(x) <b>SSDF:</b> PS.3.2 <b>SLSA:</b> Build L1: Prevenance exists <b>800-161:</b> SR-4 <b>OWASP SCVS:</b> 6 <b>CNCF SSC:</b> V: Ensure clients can perform verification of artifacts and associated metadata <b>Self-attestation:</b> 3</p>

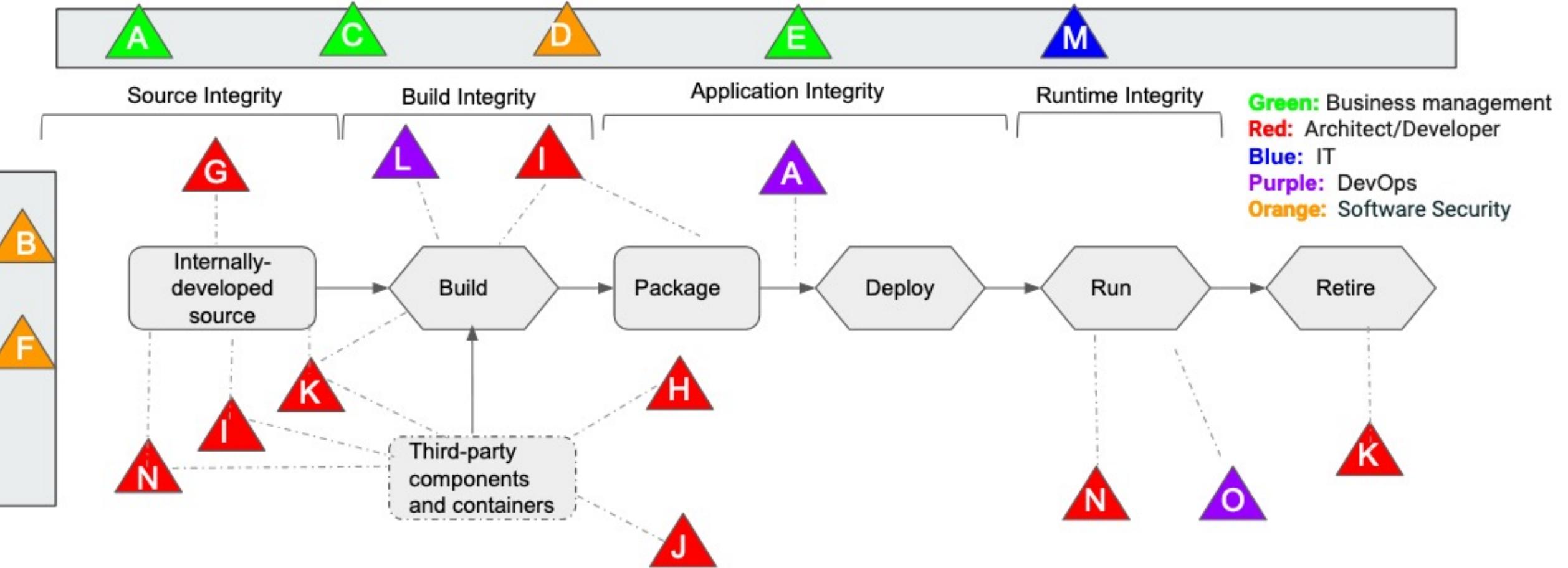
Bi-directional equivalence

Feedback welcome!

# P-SSCRM Framework (4 Groups, 15 Practices, 72 Tasks)

<b>Governance (23 tasks)</b>	<b>Product (19 tasks)</b>	<b>Environment (22 tasks)</b>	<b>Deployment (8 tasks)</b>
<p>Tasks that focus on the organization and measurement of a secure software supply chain and of policies for decision making, accountability to third-party obligations, and remaining compliant with legal and regulatory requirements.</p>	<p>Tasks to lead to the deployment of a secure product with minimal vulnerabilities with associated required attestations and artifacts.</p>	<p>Tasks to protect the confidentiality and integrity of source code, software components, and the build infrastructure from tampering and unauthorized access.</p>	<p>Tasks for identifying, analyzing, and addressing vulnerabilities in products.</p>
<ul style="list-style-type: none"><li>• Perform compliance (5)</li><li>• Develop security policies (6)</li><li>• Manage suppliers (5)</li><li>• Train (3)</li><li>• Assess and manage risk (4)</li></ul>	<ul style="list-style-type: none"><li>• Develop security requirements (2)</li><li>• Build security in; software security (5)</li><li>• Make good component choices (5)</li><li>• Discover vulnerabilities (4)</li><li>• Manage vulnerable components (2)</li></ul>	<ul style="list-style-type: none"><li>• Safeguard artifact integrity (6)</li><li>• Safeguard build integrity (7)</li><li>• Secure environment (9)</li></ul>	<ul style="list-style-type: none"><li>• Respond to vulnerabilities (6)</li><li>• Monitor intrusions/violations (2)</li></ul>

# P-SSCRM Framework - Lifecycle View



## Governance

- **A** Perform compliance
- **B** Develop security policies
- **C** Manage suppliers
- **D** Training
- **E** Assess and manage risk

## Product

- **F** Develop security requirements
- **G** Build security in
- **H** Manage component and container choices
- **I** Discover vulnerabilities
- **J** Manage vulnerable components and containers

## Environment

- **K** Safeguard artifact integrity
- **L** Safeguard build integrity
- **M** Secure software development environment

## Deployment

- **N** Respond to/disclose vulnerabilities
- **O** Monitor intrusions/violations

# Task coverage with all the frameworks #[#unique]

Framework	Governance	Product	Environment	Deployment	Total
P-SSCRM	23	19	22	8	72
EO / SSDF	11	14	4	5	34/34
Self-attestation	8	12	4	5	23/34 SSDF
BSIMM	17 [1]	14	2	4	37/125
SLSA	2	1	3	0	6/6
NIST 800-161	20 [5]	10	9	5 [1]	44/183
OWASP SCVS	1	5	5	0	11/11
S2C2F	3	7 [1]	3	2 [1]	15/15
CNCF SSC	4	6	13 [8]	1 [1]	24/24

# Empiricism

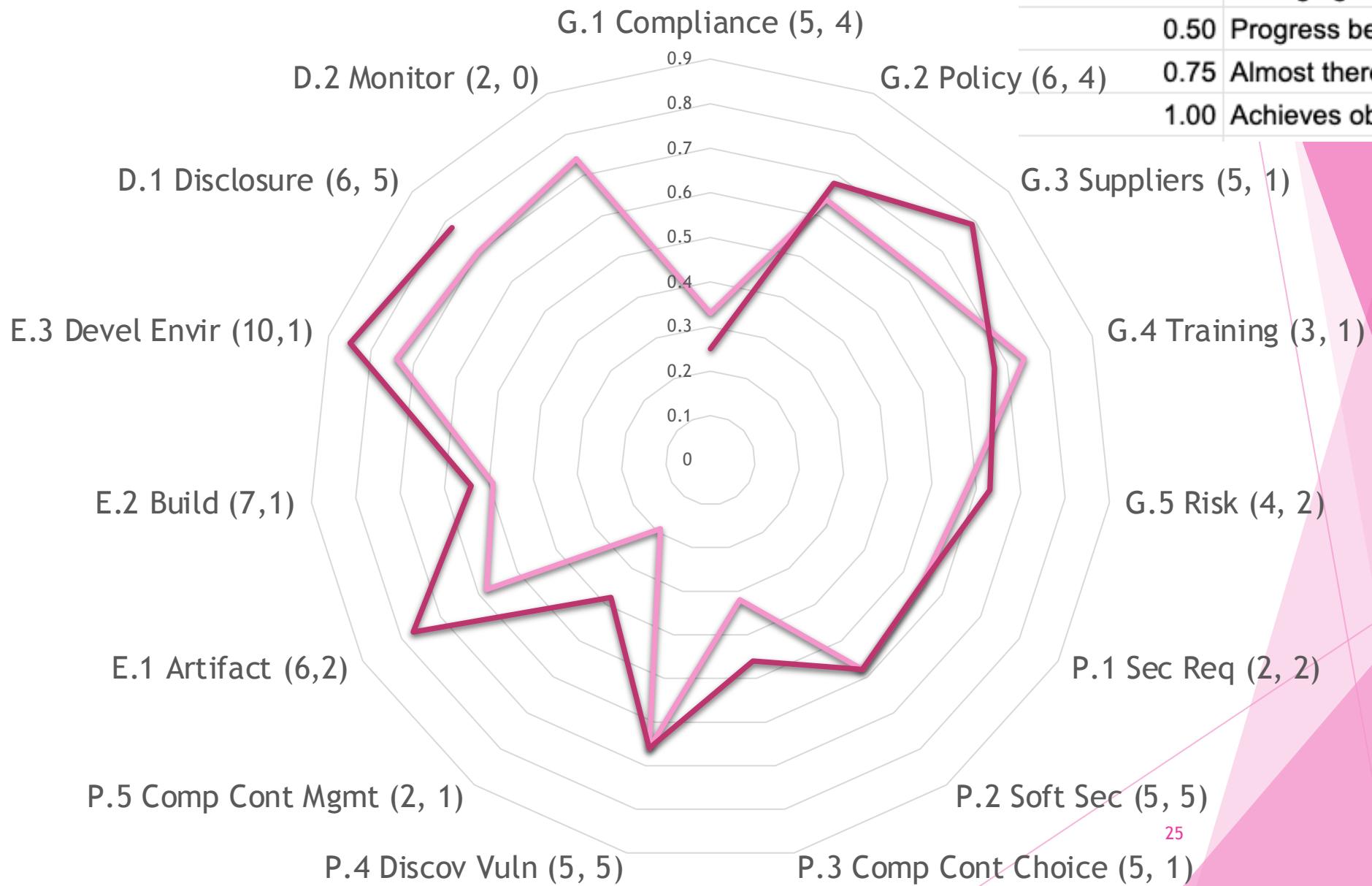


# Where everybody's at

—All —SSDF

## Rating

0.00	Does not achieve objective
0.25	Emerging
0.50	Progress being made
0.75	Almost there
1.00	Achieves objective/exemplary



# Governance: perform compliance

Identifier	Task name	Average	Practice average	SSDF average
G.1.1*	Org security requirements	0.64		
G.1.2	Software licences	0.65		
G.1.3*	Produce attestation	0.13		
G.1.4*	Deliver provenance	0.07		
G.1.5*	Deliver SBOM	0.14	0.33	0.25

- ▶ In general, organizational security requirements = defined software development lifecycle (SDLC)
- ▶ Checking licenses (via Software Compositional Analysis (SCA tools) is pretty mature, pre-dates this supply chain security mess
- ▶ Just starting to think about attestation and delivering provenance and SBOM
  - ▶ Most are experimenting with or are already producing SBOM
  - ▶ Sharing, delivering ... not so much

# Governance: develop security policies

Identifier	Task name	Average	Practice average	SSDF average
G.2.1*	Upper management support	<b>0.79</b>		
G.2.2*	Secure SDLC & security checks	<b>0.63</b>		
G.2.3*	Roles & responsibilities	<b>0.80</b>		
G.2.4*	Code review policy	<b>0.50</b>		
G.2.5	Asset inventory	<b>0.29</b>		
G.2.6	Protection of info at rest	<b>0.81</b>	0.64	<b>0.68</b>

- ▶ Code review policy doesn't always involve security checking
- ▶ Current-day asset inventory is confusing and dynamic - some don't really understand what to do
  - ▶ Containers
  - ▶ Ephemeral environments
  - ▶ Cloud resources

# Governance: manage suppliers

Identifier	Task name	Average	Practice average	SSDF average
G.3.1*	Security-related contract terms	<b>0.79</b>		
G.3.2	Separation of duties	<b>0.33</b>		
G.3.3	Information disclosure	<b>0.56</b>		
G.3.4	Session audits	<b>0.63</b>		
G.3.5	Notification agreement	<b>0.83</b>	0.63	<b>0.79</b>

- ▶ Vendor managers seem to be pretty good at imposing “all the things” on the vendors
- ▶ Less mature at more than one person reviewing contractors and contracts
  - ▶ Exemplary - collaboration between contract manager and software security

# Governance: training

Identifier	Task name	Average	Practice average	SSDF average
G.4.1	Role-based training	0.89		
G.4.2	Contingency training	0.67		
G.4.3*	Gather attack trends	0.67	0.74	0.67

- ▶ Prevent, detect, respond
- ▶ Room for improving:
  - ▶ Procedures in the event of a security emergency
    - ▶ Some do table-top exercises and simulations
  - ▶ Studying cyberthreat intelligence, attending conferences, etc., and getting trends out to the organization



# Governance: assess and manage risk

Identifier	Task name	Average	Practice average	SSDF average
G.5.1	Criticality analysis	<b>0.46</b>		
G.5.2*	Track security risks & decisions	<b>0.64</b>		
G.5.3	Security metrics	<b>0.61</b>		
G.5.4*	Data-informed product decisions	<b>0.63</b>	0.58	<b>0.63</b>

- ▶ Everybody knows they need to do these, the actual processes are less structured, repeatable, objective
- ▶ Security metrics = hard problem
  - ▶ Are we getting more secure? Less secure? Are the tasks working?



# Product: develop security requirements

Identifier	Task name	Average	Practice average	SSDF average
P.1.1*	Product security requirements	<b>0.54</b>		
P.1.2*	Software release integrity	<b>0.55</b>	0.55	<b>0.55</b>

- ▶ Specifying product-specific security requirements less mature than organizational (SDLC) requirements
  - ▶ Architectural
  - ▶ Memory-safe languages
  - ▶ Sandboxing/isolation
  - ▶ Modularity
  - ▶ Security features
- ▶ Providing “customers” assurance your software is legit
  - ▶ Such as signing code

# Product: build security in

Identifier	Task name	Average	Practice average	SSDF average
P.2.1*	Security design review	<b>0.61</b>		
P.2.2*	Secure coding	<b>0.61</b>		
P.2.3*	Secure by default implementation	<b>0.39</b>		
P.2.4*	Standard security features	<b>0.70</b>		
P.2.5*	In-house components	<b>0.65</b>	0.59	<b>0.59</b>

- ▶ “Getting there” with lots of proactive software security practices
- ▶ Secure by default versus usability is the general dilemma
- ▶ Sometimes in-house components are forgotten and not scanned, can get stale, and not maintained



# Product: manage component and container choices

Identifier	Task name	Average	Practice average	SSDF average
P.3.1	Component and container choice	0.43		
P.3.2	Trusted repositories	0.40		
P.3.3	Require signed commits	0.13		
P.3.4*	Vetted third-party repositories	0.46		
P.3.5	Prevent component vetting bypass	0.17	0.32	0.46

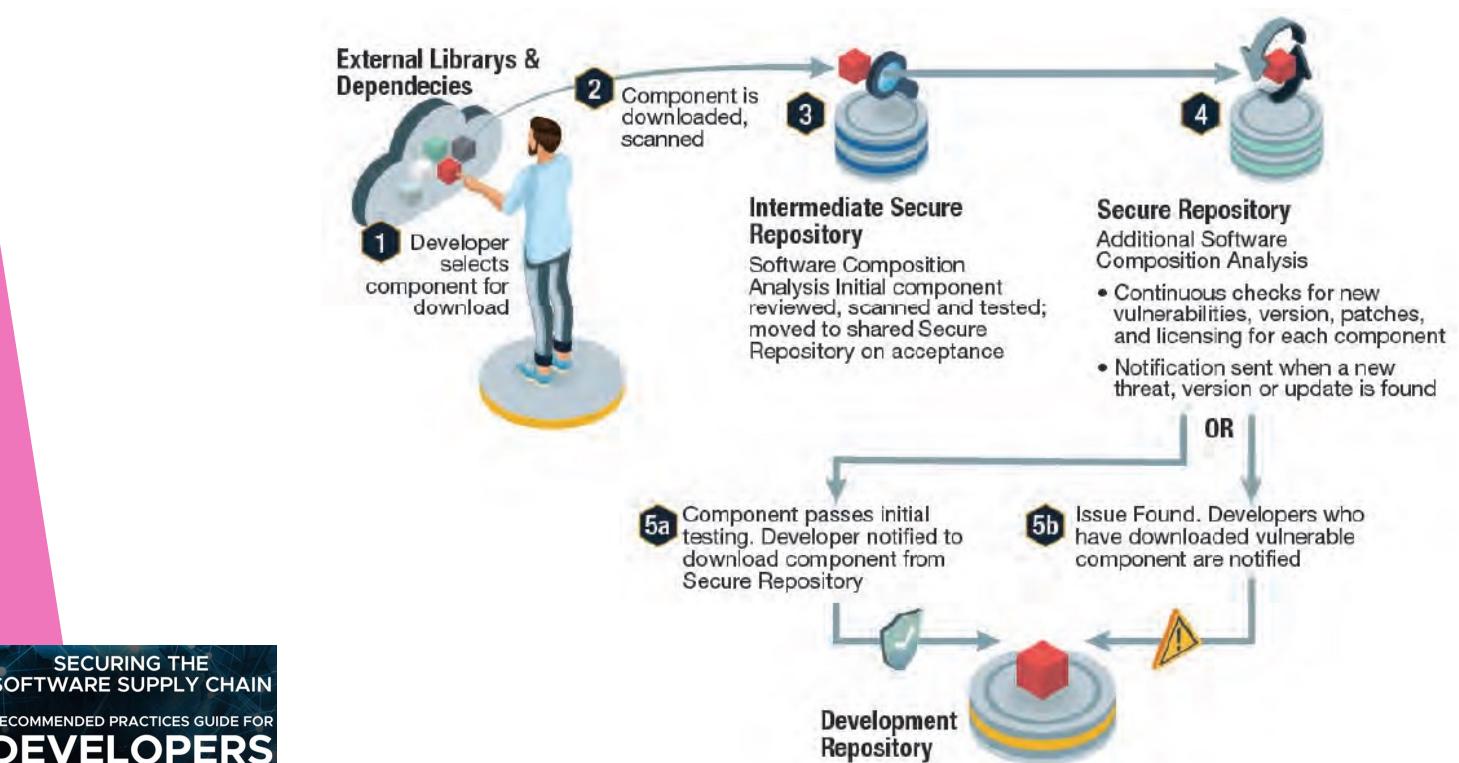


Figure 3: Secure Repository Process Flow



(second to last)

## Components > Containers

# Product: discover vulnerabilities

Identifier	Task name	Average	Practice average	SSDF average
P.4.1*	Code review	0.40		
P.4.2*	Automated security scanning tools	0.96		
P.4.3*	Automated vulnerability detection	0.63		
P.4.4*	Executable security testing	0.71		
P.4.5*	Regular third-party compliance	0.57	0.66	0.66



- ▶ Prevent, detect, respond
- ▶ Code review .... “for every PR”, but unsure of how enforced in reality
- ▶ Have lots of tools, but run regularly and vulnerabilities fixed, less so
- ▶ External pen test, bug bounty - yes, internal red team, testing less
- ▶ Review of third-party compliance to contract lacking
- ▶ Review of “are open source components abandoned” type of checks lacking
  - ▶ Relying on SCA tools to find vulnerabilities

# Product: manage vulnerable components

Identifier	Task name	Average	Practice average	SSDF average
P.5.1	SBOM consumption	0.00		
P.5.2*	Dependency update	0.38	0.19	0.38



- ▶ Not really doing anything with SBOM (or real plans to)
- ▶ Dependency update = drinking from the firehose
  - ▶ Not a systematic process for handling this overwhelm



# Environment: safeguard artifact integrity

Identifier	Task name	Average	Practice average	SSDF average
E.1.1*	Safely store release artifacts	<b>0.70</b>		
E.1.2*	Version control	<b>0.83</b>		
E.1.3	Multi-factor authentication (MFA)	<b>0.40</b>		
E.1.4	Developer SSH key	<b>0.50</b>		
E.1.5	Branch protection	<b>0.46</b>		
E.1.6	Decommission assets	<b>0.25</b>	0.52	0.77

- ▶ Advanced authentication maturing
- ▶ Lack of security checks enforced in branching process
  - ▶ Especially with mono repo
- ▶ Security risks of end-of-life systems, program, assets
  - ▶ Just added to P-SSCRM, so low sample size
  - ▶ Not a task in any of the originating frameworks



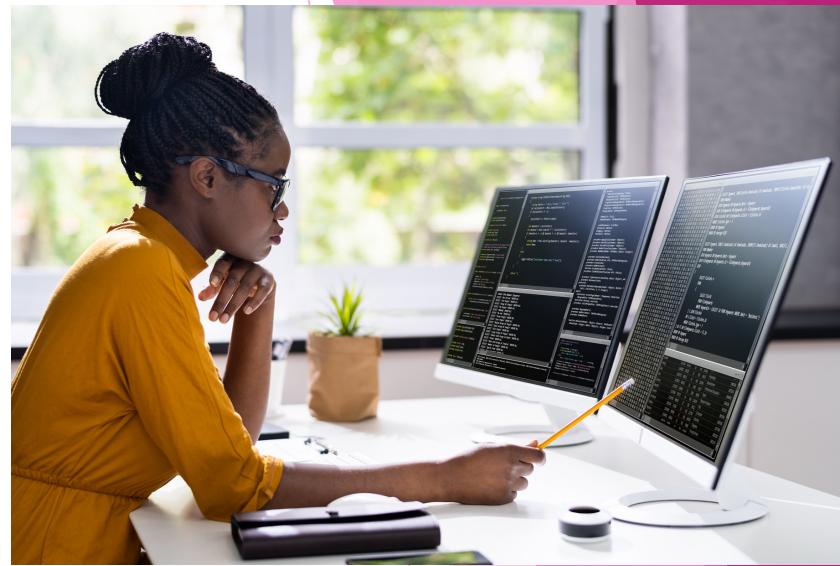
# Environment: safeguard build integrity

Identifier	Task name	Average	Practice average	SSDF average
E.2.1	Release policy verification	<b>0.42</b>		
E.2.2	Verify dependencies & environment	<b>0.30</b>		
E.2.3	Defensive compilation & build	<b>0.29</b>		
E.2.4*	CI/CD automation & protection	<b>0.54</b>		
E.2.5	Secure orchestration platform	<b>0.88</b>		
E.2.6	Reproducible builds	<b>0.00</b>		
E.2.7	Build output	<b>1.00</b>	0.49	<b>0.54</b>

- ▶ Automated release policy verification could be better
  - ▶ “as code”, templated, standardized
- ▶ Verifying dependencies & environment on build could be better
- ▶ Don’t utilize compiler, interpreter, such as to fail rather than give a security warning
- ▶ Ephemeral builds pretty good
- ▶ Hermetic, parameter-less builds emerging

# Environment: secure software development environment

Identifier	Task name	Average	Practice average	SSDF average
E.3.1	Authentication	1.00		
E.3.2*	Environmental separation	0.85		
E.3.3	Role-based access control	0.75		
E.3.4	Information flow enforcement	0.70		
E.3.5	Baseline configuration	0.42		
E.3.6	Monitor changes to config settings	1.00		
E.3.7	Boundary protection	0.83		
E.3.8	Key rotation	0.80		
E.3.9	Ephemeral credentials	0.35		
E.3.10	Establish a root of trust	0.67	0.74	0.85



- ▶ Prevent, detect, respond
- ▶ RBAC: maybe too widespread “everyone can read everything” and not enough least privilege
- ▶ Could use more baseline configuration, use of ephemeral credentials

# Deployment: respond to/disclose vulnerabilities

Identifier	Task name	Average	Practice average	SSDF average
D.1.1*	Vulnerability analysis	0.85		
D.1.2*	Risk-based vulnerability remediation	0.88		
D.1.3*	Vulnerability disclosure	0.83		
D.1.4*	Vulnerability eradication	0.55		
D.1.5	Emergency fix	0.31		
D.1.6*	Root cause analysis	0.80	0.70	0.78



- ▶ Prevent, detect, respond
- ▶ Emergency fix (from S2C2F) = what to do if the component supplier won't fix?
- ▶ Could be more proactive eradication
  - ▶ One company said "... if we don't, the bug bounty people will just keep finding more of the same."



# Deployment: monitor intrusions/violations

Identifier	Task name	Average	Practice average	SSDF average
D.2.1	System monitoring	1.00		
D.2.2	Build process monitoring	0.50	0.75	N/A

- ▶ Solarwind, Codecov ... need to get better at monitoring for build process intrusions

# Top 10 Tasks (1G, 1P, 5E, 3D)

Identifier	Task name	Average
E.2.7	Build output	1.00
E.3.1	Authentication	1.00
E.3.6	Monitor changes to config settings	1.00
D.2.1	System monitoring	1.00
P.4.2*	Automated security scanning tools	0.96
G.4.1	Role-based training	0.89
E.2.5	Secure orchestration platform	0.88
D.1.2*	Risk-based vulnerability remediation	0.88
E.3.2*	Environmental separation	0.85
D.1.1*	Vulnerability analysis	0.85

# Bottom 10 Tasks (4G, 3P, 3E, 0D)

Identifier	Task name	Average
P.5.1	SBOM consumption	0.00
E.2.6	Reproducible builds	0.00
G.1.4*	Deliver provenance	0.07
G.1.3*	Produce attestation	0.13
P.3.3	Require signed commits	0.13
G.1.5*	Deliver SBOM	0.14
P.3.5	Prevent component vetting bypass	0.17
G.2.5	Asset inventory	0.29
E.2.3	Defensive compilation & build	0.29
E.2.2	Verify dependencies & environment	0.30

# Summary

- ▶ Software supply chain attacks are on the rise
- ▶ Attack vectors:
  - ▶ Accidentally-injected vulnerabilities
  - ▶ Maliciously-injected vulnerabilities
  - ▶ Attacks through the build infrastructure
  - ▶ [Probably emerging] LLM-generated code
- ▶ International regulation is imposing software security practices on development teams
- ▶ Liability for insecure code is emerging
- ▶ Practices to protect from vulnerable components are not being adopted as fast as probably needed.
- ▶ We [software engineering researchers] need to make secure development less disruptive to a development workflow