SUPPLY CHAIN SECURITY AT STACKLOK

KTH CHAINS workshop, Apr-26, 2024

Who am I?

- Czech Software developer living in Sweden
- Have been working around software security for most of my career
- ex-Red Hat (Kubernetes security and compliance, identity management)
- Currently at Stacklok

What am I talking about?

- A little different from the initial abstract (sorry Martin)
- This is NOT a product pitch
  - But the initial presentation was turning this way
  - We have nothing to sell you yet..
- Instead, this is:
  - An overview of what we do, why we do it and how it fits into supply chain security
  - Where we are on the supply chain = where we think it's worth spending energy
  - For whom we try to solve the problems
  - ..and then finally the tools we developed to address ^^^

What's Stacklok?

- A supply chain security startup (~30 people, ~20 engineers)
- Founded in May 2023
  - Coding started in summer 2023
  - Several people, including the founders have background in Kubernetes or Sigstore

Trusty: Make safer dependency choices

Minder: Secure your supply chain pipeline

Supply chain attack landscape

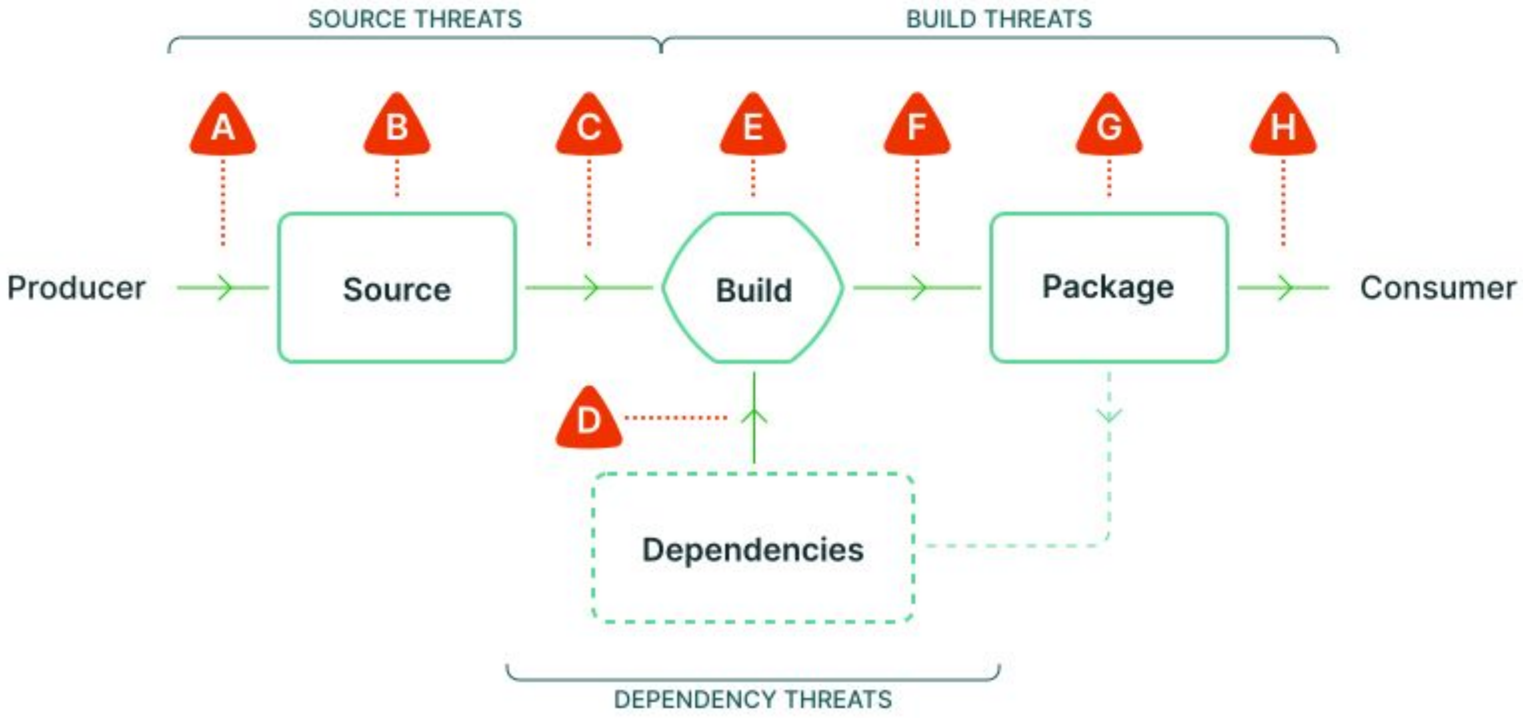
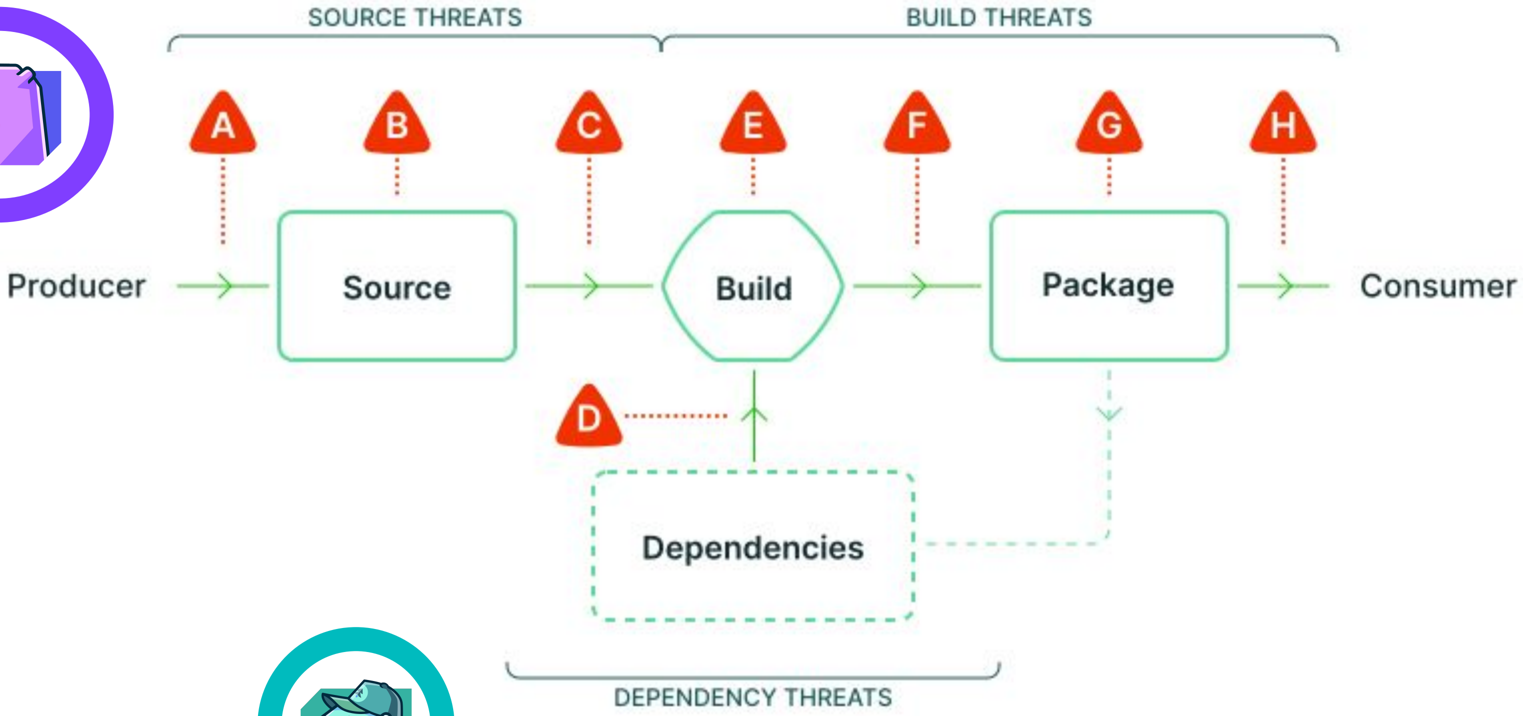


image: slsa.dev; Note: all, but especially D vary over time. D is tech+human

How does Stacklok fit in?



What do we not do? CVE management

- Hot take: CVEs are overrated

- A lot of malicious code has no CVEs
- Conversely, most (xz excluded) CVEs do not have a malicious intent behind it
- Most CVEs are not exploited, many not exploitable
  - See also: Red Hat’s product security report (tl;dr about 0.4% of CVEs are actively exploited)

Supply chain personas

- The developer
  - Writes code, typically just wants to go about their day
- Infrastructure engineer
  - Runs the code written by developers, securely
- Product security/Security Engineer/
  - Sets the security standard across the organization for both of the above
  - Unfortunately too often perceived like a heavy-handed enforcer and not understanding the other 2

The personas are often at odds with each other

What do they want and need?

- The developer
  - Security is often an annoyance (“why can’t I hardcode this secret temporarily”, “why can’t I just base my images on XYZ”, “why can’t I just use libfoo”), and perceived as hindering productivity
  - Irritating if security comes too late in the cycle
- Infrastructure engineer
  - Must secure the infra, but lacks visibility into the software
  - Their nightmare: xz/log4j/… happened. Find what’s affected and fix it.
- Product security/Security Engineer
  - No bandwidth to keep up with development - sprawl
    - A wild new repo appeared.
    - An old repo has a new dependency.
    - How do I even know that happened? Are they secure?
  - Every subgroup of the above has (and needs!) their own definition of “secure”

So what do you really do?

- Try to find the sweet spot in between these personas
- Something that the security people find useful but developers don’t hate

What does Trusty do?

- Trusty: Helping you make safer dependency choices
    - Scores dependencies and presents with alternatives for low-scoring deps
    - Scores are based on e.g. repo activity, author activity, provenance, trying to compare with other projects to find impersonators
    - Make developers happy by giving them the information soon enough
    - Make developers happy by giving them info where they are, don’t make them redo work
    - Make product security happy because it reduced the number of incidents down the road

What does Minder do?

- Minder: Securing your supply chain
  - Very *flexible and extensible* engine
    - One size fits all ends up fitting none
  - Centralized policies applied across the board (repos)
  - Remediations and alerts directly to repos and PRs
  - Making the developers happy by automating and moving left as much as possible
  - Making product security happy by centralizing and expressing policy as code
  - Making infrastructure engineering happy by giving them insight into the code and means of enforcing prodsec requirements
  - Minder ❤️ Trusty

Checkpoint

Does this make sense so far?

Next: Trusty and dependency safety

Our take on dependency safety

- Can you guess it going back to the personas earlier?
- Example: You are asked to add social logins using OAuth2 to a Python application
- This involves selecting a dependency that you will pip install
- But pip install what?
- The most common case
  - <fill the blank>

Our take on dependency safety

- Can you guess it going back to the personas earlier?
- Example: You are asked to add social logins using OAuth2 to a Python application
- The most common case
  - Google, StackOverflow, tutorials, (chatGPT?)
  - Nothing wrong with that per se, but what about security?
  - In this case chances are they find python-oauth2
- The security conscious developer
  - <fill the blank>

Our take on dependency safety

- Can you guess it going back to the personas earlier?
- Example: You are asked to add social logins using OAuth2 to a Python application
- The most common case
  - Google, StackOverflow, tutorials
- The security-conscious developer:
  - Look up the package on pip, on GH, check how many stars, when was the last commit, who uses this, who contributes
    - Unmaintained dependency might be worse than one with CVEs!
  - Crucially: Who uses and who contributes *that I know and trust (example:. sigstore-go)*
  - You're trying to see if you can trust this project
  - Hopefully they'd find oauthlib instead

Trusty

- I am not an expert on Trusty (but can point you in the right direction)
- An API service with a web UI
- Continuously ingests popular package repositories (pypi, npm, rust crates, Java packages) and assigns several scores to them
- Try to find suspicious packages from package managers
    - Not static analysis on the code, but on the package manager - e..g a new user uploads a bunch of packages with no links to a repo triggers an alarm
- Exposes the data through APIs that:
    - Provide a summary of trust scores
    - Provide alternatives
    - Provide reputation graphs (closed beta)
- Closed source, but free to use SaaS - https://trustypkg.dev

Trusty demo

- Entry point: https://trustypkg.dev

- Examples of bad dependencies
- Unmaintained dependency - recency of commits, are deps updated, are PRs and issues stale
- Masquerading - a package copies metadata to its own repo (README.md, docs, …)
    - This package seems to been taken down but was pretending to be “Marked” (w/o js)
- Typosquatting - Levenstein distance of names + diff between activity and popularity
- Repojacking, starjacking - sigstore and historical provenance
    - Sigstore > historical provenance, but HP often good enough

Exciting stuff, but can't demo, sorry

- Package and developer reputation - Proof-of-Diligence aka reputation graphs
  - Closed beta, can't show to general public yet
  - Difficult to get right, scoring humans is a touchy topic
    - Can you guess the first thing *anyone* does?
  - Read more here before this feature is public
  - Constructs a trust graph across projects and their dependencies and their contributors
    - The graph is seeded with Trusty scores for packages and initial scores for contributors based on their historical contributions
    - Contributors affect projects and vice versa, projects affect project

Checkpoint

- Does this make sense so far?

Next: Minder and securing the supply chain

Minder: Securing your supply chain

- Minder aims to be the control plane for your supply chain (buzzword bingo!)
    - Some lessons learned from Kubernetes - extensibility, level vs. edge triggering, extensibility
- Open Source
- Stacklok runs a SaaS free to use (with caveats)
- Mostly targeting the prodSec and infra engineer personas
    - And open source maintainers!

Minder: Securing your supply chain

- Concepts:
    - Minder connects to providers (e.g. GitHub)
    - Providers give Minder access to entities (e.g. repositories, PRs, container images)
    - Minder applies profiles to entities
    - Profiles live in projects (think groups or namespaces)
    - Profiles consist of rules
    - The policy evaluations can be read by an administrator..
    - ...or automatically remediated
    - ...or propagated to <somewhere> with alerts
    - And potentially consumed by agents through API

Minder demo

Live demo because what could go wrong...

https://cloud.stacklok.com

Minder: policy pipeline

- Minder policy pipeline in four steps
    - Read details about an entity (rest, git)
    - Check if an entity conforms to a policy profile (rego, jq)
        - if yes, the policy succeeds
    - If not, try to automatically remediate the failure (rest, pull request)
        - If remediation succeeds, the policy succeeds as well
    - If not, raise an alert (GHSA)

Minder: policy pipeline

- Minder’s policy engine does not care what the policy does actually do
- Building blocks available but you can stack the block yourself
- Batteries included though
    - Stacklok’s profiles and rules are available online
    - We dogfood these ourselves.

Minder: Policy building blocks

- Ingest - git, REST, container, dependency
- Evaluate - JQ, rego, OSV.dev, Trusty
- Remediate - REST, pull request
- Alert - GHSA (planned: Slack, PagerDuty, …)
- Examples:
    - Ingest a repository using git, make a check on the contents using rego, remediate using pull request
    - Ingest information about a repository using REST, remediate using REST
    - Ingest information about a container using container, check on the container using rego
    - Ingest pull request using dependency, evaluate using Trusty, remediate using pull request
    - …..

Batteries included

- A lot of rules available for different areas:
    - Ensure that github actions are pinned to SHAs
    - Ensure that workflows only use approved actions with defined permissions
    - Ensure that every repo has a license
    - Ensure that artifacts have sigstore provenance
    - Ensure that no PR introduces a new CVE or low Trusty dependency
    - Ensure that there are no binaries in the repo
    - ....
- Rules are then grouped into profiles
- Profiles can be grouped into projects
- Stacklok's profiles and rules are available online
    - We dogfood these ourselves.

Minder roadmap

- This slide is speculative!
- More rules and profiles to cover different scenarios
- Dependency analysis with Trusty help (graph across transitives)
- More providers
  - OCI registry, build environments
- Evaluating minder policies for workloads
  - Admission, mapping?

Thank you

- Questions?
- Links!
  - My LinkedIn
  - Stacklok homepage
  - Talk to us on Discord
  - Trusty homepage and docs
  - Minder Cloud and docs
  - Minder GitHub repo