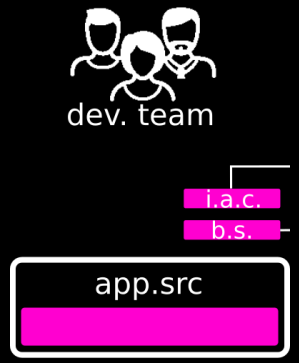
the security implications of the software supply chain

CHAINS project

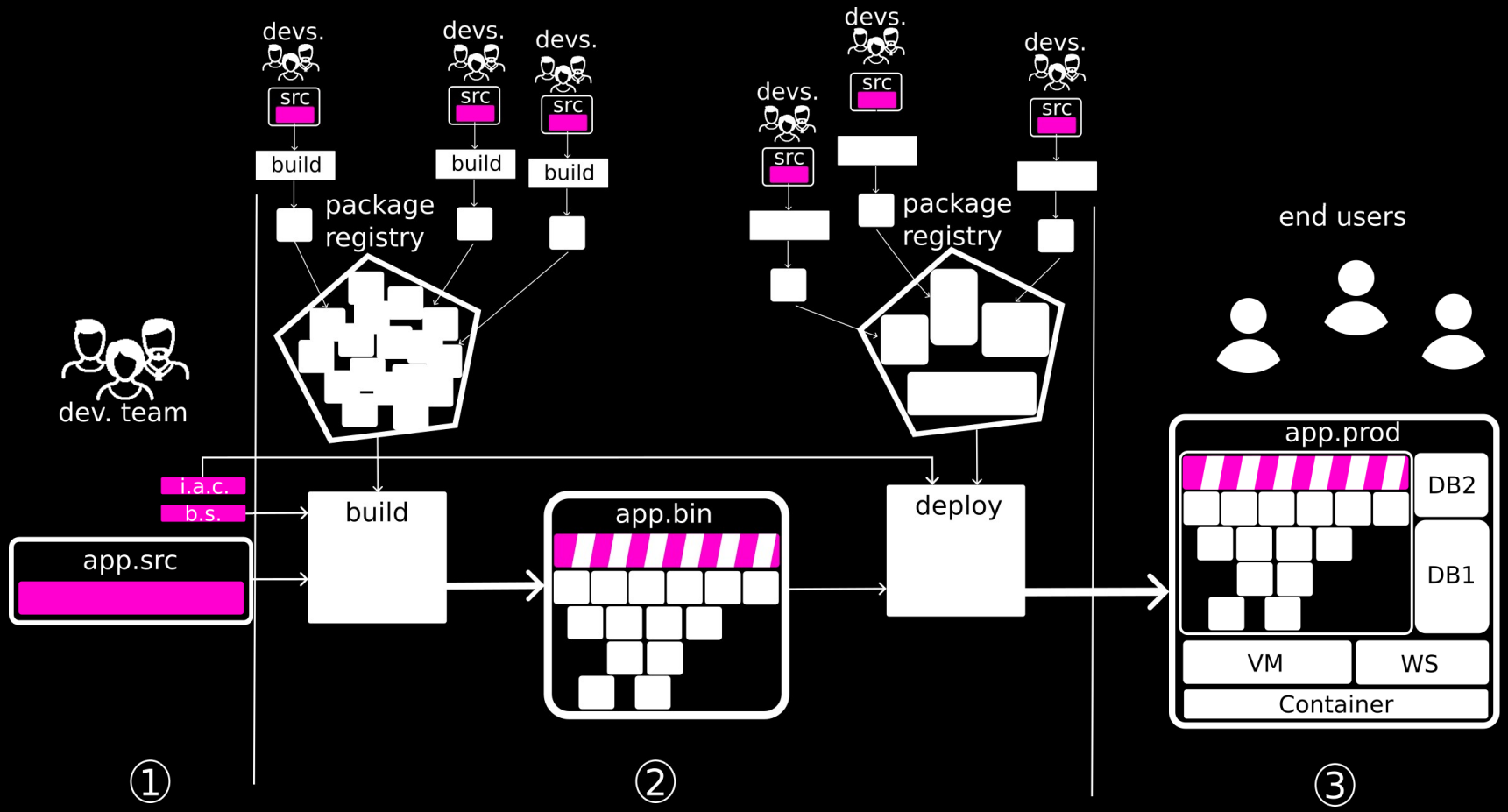
https://chains.proj.kth.se/

Software Supply Chain

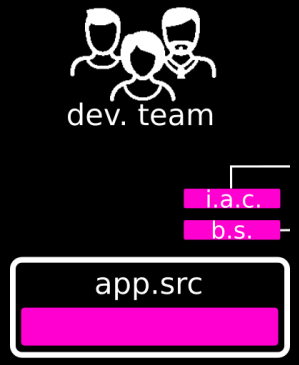


①

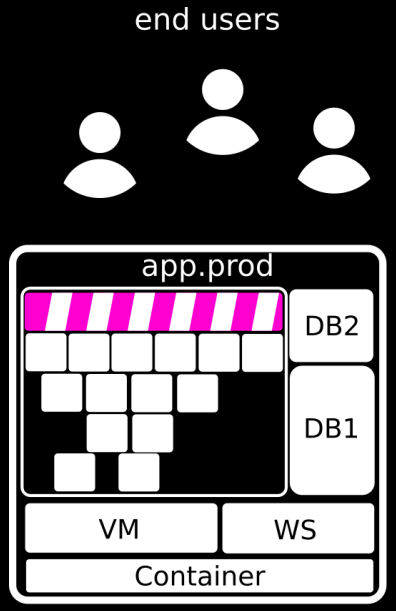
Software Supply Chain



Software Supply Chain



①



③

Attacks on the Supply Chain

- Infect a downstream dependency
  - Inject a vulnerability in an open source package and let it propagate until target
  - Examples: event-stream, log4shell

Attacks on the Supply Chain

- Infect the build system
  - Trusting trust attack: compromise the compiler or other tool involved in the build
  - Example: Solarwinds, codecov

Attacks on the Supply Chain

- Infect the package registry
  - Typosquatting: publish packages which names are very similar to popular packages and let the malicious packages propagate into applications
  - Example: PyPI typosquatting

Software Bill of Materials

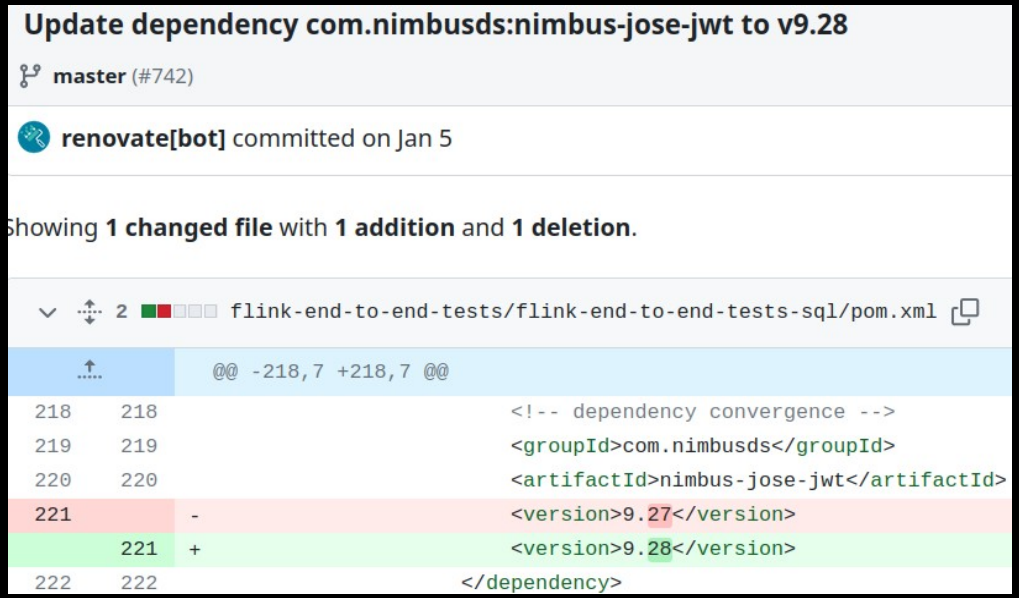
- List all components used to build an application
- White House Executive Order 14028
- SPDX and CycloneDX
- Balliu et al. “Challenges of Producing Software Bill Of Materials for Java”, 2023


```
{ "bomFormat": "CycloneDX",
  "specVersion": "1.4",
  "version": 1,
  "components": [
    {
      "bom-ref": "org.apache.ant:ant:1.10.12",
      "type": "library",
      "group": "org.apache.ant",
      "name": "ant",
      "version": "1.10.12",
      "hashes": [
        {
          "alg": "SHA-256",
          "content": "5c6a438c3ebe7a306eba452b09fa307b0e60314926177920bca896c4a504eaf6"
        }
      ]
    },
    {
      "bom-ref": "commons-fileupload:commons-fileupload:1.4",
      "type": "library",
      "group": "commons-fileupload",
      "name": "commons-fileupload",
      "version": "1.4",
      "hashes": [
        {
          "alg": "SHA-256",
          "content": "a4ec02336f49253ea50405698b79232b8c5cbf02cb60df3a674d77a749a1def7"
        }
      ]
    }
      etc....
  ],
  "dependencies": [
    {
      "ref": "org.springframework.security:spring-security-web:5.8.1",
      "dependsOn": [
        "org.springframework:spring-core:5.3.24",
        "org.springframework:spring-web:5.3.24",
        "org.springframework:spring-context:5.3.24"
      ]
    },
    {
      "ref": "org.jenkins-ci.plugins:bouncycastle-api:2.26",
      "dependsOn": [
        "org.bouncycastle:bcpkix-jdk15on:1.70",
        "org.bouncycastle:bcutil-jdk15on:1.70",
        "org.bouncycastle:bcprov-jdk15on:1.70"
      ]
    }
    etc....
  ]
}
```


https://github.com/chains-project/SBOM-2023/blob/main/results-march-2023/jenkins/build-info-go/build-info-go.json

Automatic Dependency Updates

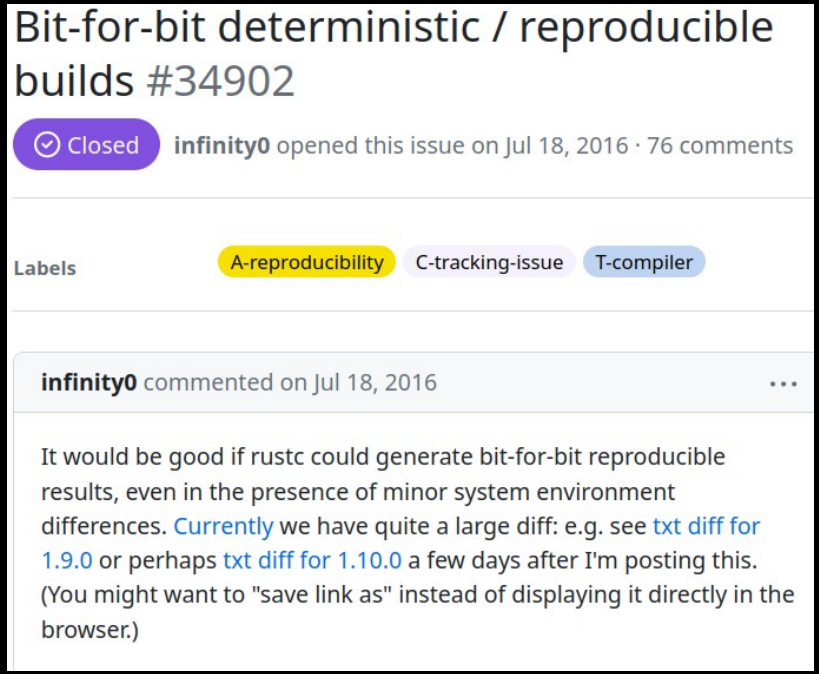
- Automatically scan the dependency tree and suggest changes for updates
- Dependabot, Renovate
- Cogo et al. “Understanding the Customization of Dependency Bots: The Case of Dependabot”. IEEE Software, 2022



https://github.com/chains-project/flink/commit/fe13136ed1406386762082aaef3f5735c3945ab4

Reproducible Builds

- Deterministic behavior of the whole build pipeline
- Nix and guix
- Lamb, Zacchiroli. “Reproducible Builds: Increasing the Integrity of Software Supply Chains”. IEEE Software, 2022



Bit-for-bit deterministic / reproducible builds #34902

Closed **infinity0** opened this issue on Jul 18, 2016 · 76 comments

Labels A-reproducibility C-tracking-issue T-compiler

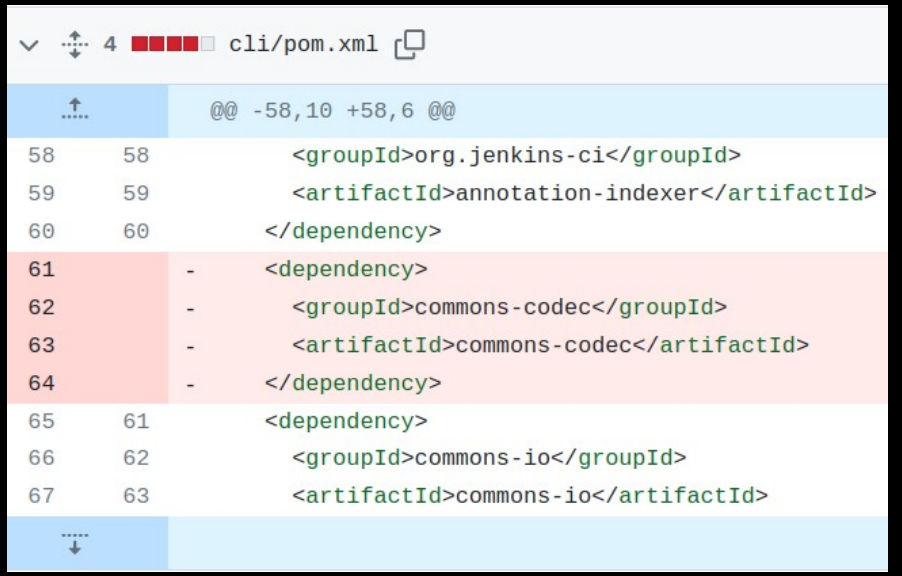
infinity0 commented on Jul 18, 2016

It would be good if rustc could generate bit-for-bit reproducible results, even in the presence of minor system environment differences. Currently we have quite a large diff: e.g. see txt diff for 1.9.0 or perhaps txt diff for 1.10.0 a few days after I'm posting this. (You might want to "save link as" instead of displaying it directly in the browser.)

https://github.com/rust-lang/rust/issues/34902

Dependency Debloating

- Remove unnecessary dependencies, reduce maintenance and security risks
- Depclean, for Java
- Soto-Valero,et al. “A longitudinal analysis of bloated Java dependencies”. ESEC/SIGSOFT FSE 2021



4 cli/pom.xml

```
@@ -58,10 +58,6 @@
58  58      <groupId>org.jenkins-ci</groupId>
59  59      <artifactId>annotation-indexer</artifactId>
60  60    </dependency>
61      -   <dependency>
62      -     <groupId>commons-codec</groupId>
63      -     <artifactId>commons-codec</artifactId>
64      -   </dependency>
65  61    <dependency>
66  62      <groupId>commons-io</groupId>
67  63      <artifactId>commons-io</artifactId>
```


https://github.com/jenkinsci/jenkins/pull/4378/files

Conclusion

- The software supply chain spans the vast amount of dependencies and tools that are necessary to build modern applications
- The software supply chain is a target for malicious actors
- Industry and academia develop solutions to address the risks of the software supply chain