

the software supply chain



CHAINS project
<https://chains.proj.kth.se/>



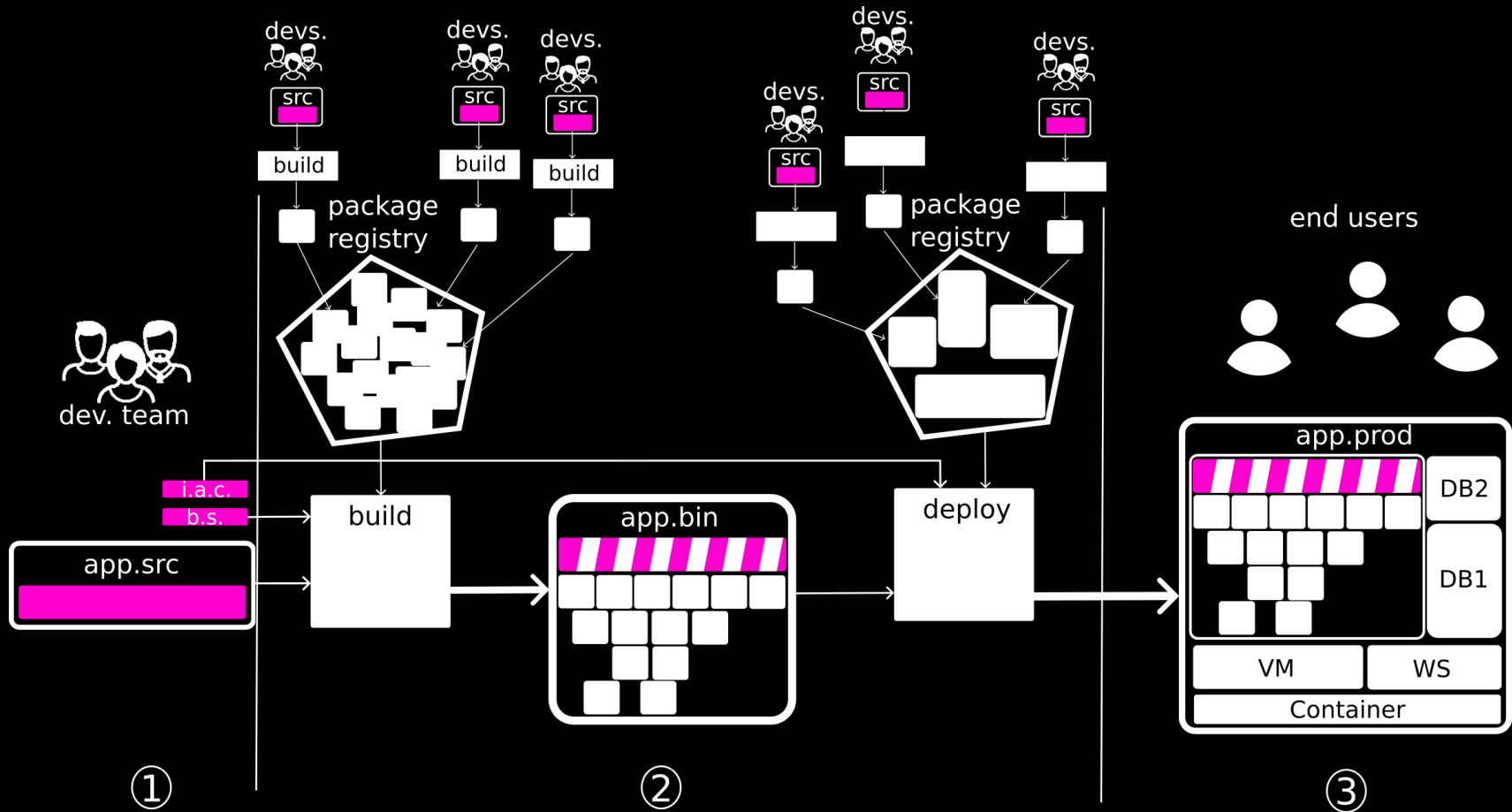
Software Supply Chain



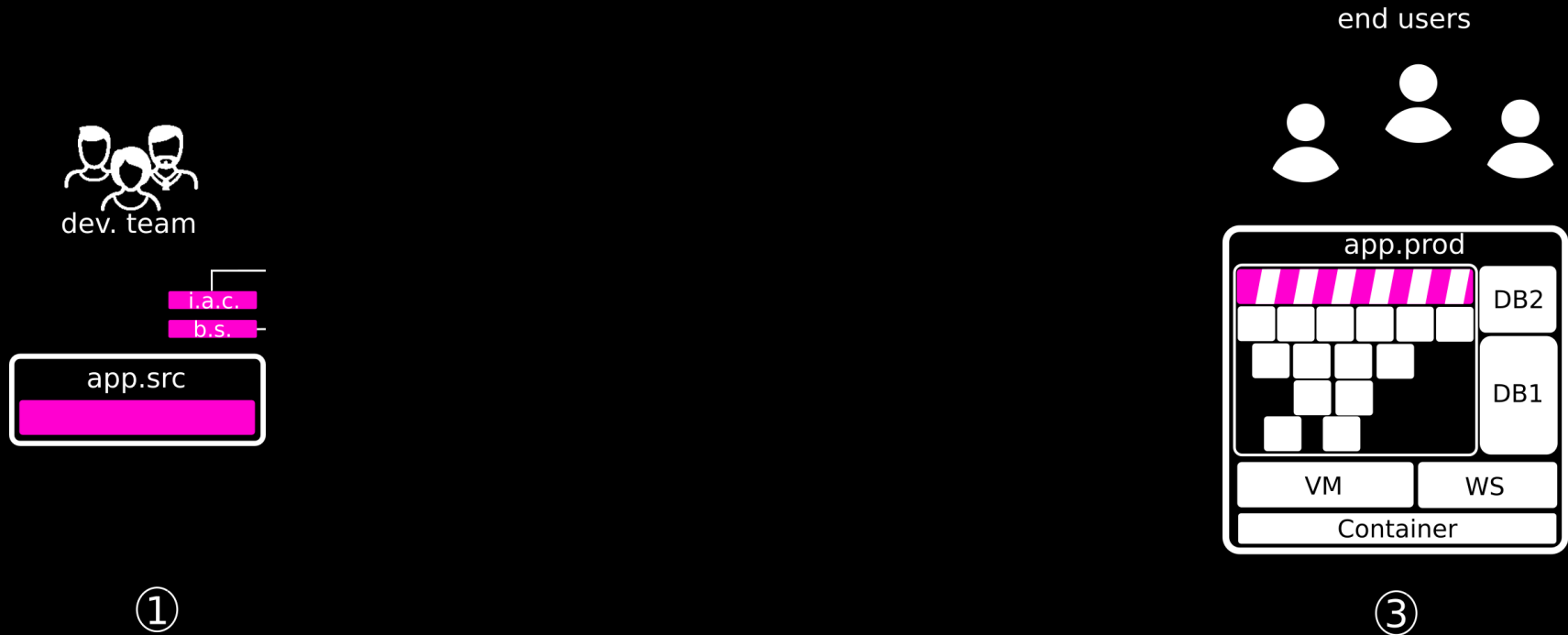
i.a.c.
b.s.



Software Supply Chain



Software Supply Chain



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:bcprov-jdk15on:1.70",
        "org.bouncycastle:bcutil-jdk15on:1.70",
        "org.bouncycastle:bcprov-jdk15on:1.70"
      ]
    }
  ],
  etc...
}
```


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

```
Update dependency com.nimbusds:nimbus-jose-jwt to v9.28
master (#742)
renovate[bot] committed on Jan 5

Showing 1 changed file with 1 addition and 1 deletion.

flink-end-to-end-tests/flink-end-to-end-tests-sql/pom.xml

@@ -218,7 +218,7 @@
218 218      <!-- dependency convergence -->
219 219      <groupId>com.nimbusds</groupId>
220 220      <artifactId>nimbus-jose-jwt</artifactId>
221  -      <version>9.27</version>
221  +      <version>9.28</version>
222 222      </dependency>
```

Reproducible Builds

- + Deterministic behavior of the whole build pipeline
- + Nix and guix
- + Lamb, Zacchioli. “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.)

Diverse Double Compilation

- + Countering trusting trust attacks
- + Wheeler, David A. "Countering trusting trust through diverse double-compiling." 2005

TURING AWARD LECTURE

Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

KEN THOMPSON

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

```
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>
```

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