

Using Open Source Safely

Verifiable Builds in Tekton

Dan Lorenc
Software Engineer - Google

**DEVOPS
WORLD**
by CloudBees

Using Open Source Safely - Intro

- Risks of using open source software
 - What you can do to protect yourself today
- Verifiable Builds
 - Tekton Demo
- How to get involved

Part 1

The Problem With Open Source Software



**DEVOPS
WORLD**
by CloudBees

Quiz Time!




Would you plug these in?

Quiz Time!



Uranium

92



U

Uranium

238.02891

2

8


18

32

21

9

2



A piece of natural uranium ore

 **U.S.NRC**
United States Nuclear Regulatory Commission
Protecting People and the Environment
As of July 20z18

Quiz Time: *What did we learn?*



```
npm install express
```

Quiz: What's the difference between these?

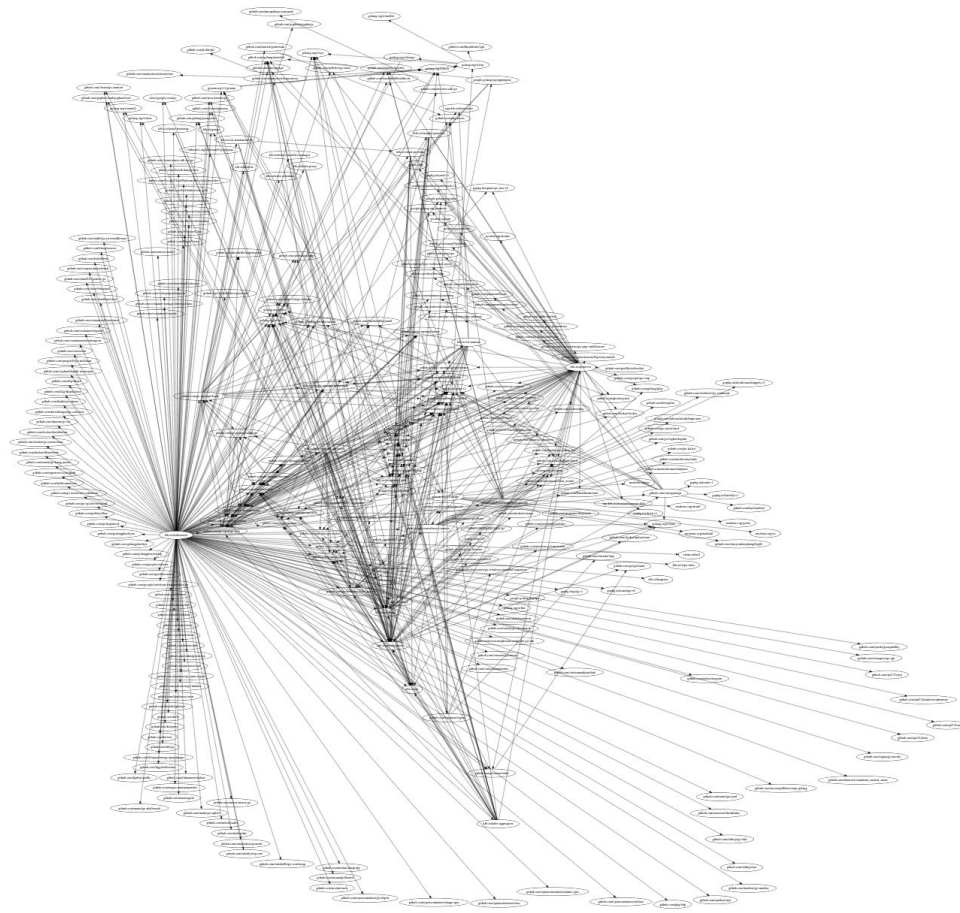
Demo: Supply Chain Attacks!



Go Modules!

- **Code review only shows dependency metadata**
 - code is not vendored by default
- **Module proxy stores code**
 - Attacks can be published and deleted from git history, making them harder to find
- **Number of dependencies is growing rapidly**
 - Hard to take the time to review each line

Kubernetes Dependency Graph



 [kubernetes / kubernetes](#)

 [Code](#)

 [Issues](#) **2,022**

 [Pull requests](#) **893**

Branch: [master](#) ▼

[kubernetes](#) / [vendor](#) / [github.com](#) /

 [Microsoft](#)

 [inconshreveable/mousetrap](#)

 [russross/blackfriday](#)

Protect Yourself!

- **Lock down your own repos.**
 - Enable and require 2FA.
 - Disable force pushes.
 - Require code review.
- **Reduce your dependencies.**
 - Every dependency is an attack point. Evaluate if you really need them all.
- **Audit your dependencies.**
 - Track updates, but review them like first-party code!
- **Observability in your CI/CD pipeline.**
 - Make sure your CI/CD system logs **everything** it does.

Part 2

Verifiable Builds



Verifiable Builds - Definitions

A Verifiable Build is a **build** that creates a **provenance** that can be **verified**

Verifiable Builds - Provenance

Provenance: the set of steps used to produce the artifact

```
minikube_iso: # old target kept for making tests happy
echo $(ISO_VERSION) > deploy/iso/minikube-iso/board/coreos/minikube/rootfs-overlay/etc/VERSION
if [ ! -d $(BUILD_DIR)/buildroot ]; then \
    mkdir -p $(BUILD_DIR); \
    git clone --depth=1 --branch=$(BUILDR00T_BRANCH) https://github.com/buildroot/buildroot $(BUILD_DIR)/buildroot; \
fi;
$(MAKE) BR2_EXTERNAL=../../deploy/iso/minikube-iso minikube_defconfig -C $(BUILD_DIR)/buildroot
mkdir -p $(BUILD_DIR)/buildroot/output/build
echo "module buildroot.org/go" > $(BUILD_DIR)/buildroot/output/build/go.mod
$(MAKE) -C $(BUILD_DIR)/buildroot
mv $(BUILD_DIR)/buildroot/output/images/rootfs.iso9660 $(BUILD_DIR)/minikube.iso
```

Verification

- Several different techniques:
 - Cryptographic build signing
 - Reproducible builds
- Reproducible builds require bit-for-bit reproducibility and determinism
- Cryptographically-signed envelope containing:
 - Inputs (source code, tools, etc.)
 - Outputs (binary artifacts produced)
 - Steps (what ran with what parameters)

Demo: Unverified Builds



**DEVOPS
WORLD**
by CloudBees

Demo: Verifiable Builds

**DEVOPS
WORLD**
by CloudBees

What's Next?

- In-Toto and other provenance formats!
- Different signature strategies (KMS integrations, minisign, etc.)
- Sharing of metadata!
 - Transparency logs, graph databases, Notary v2, etc.

Part 3

Call to Action!



**DEVOPS
WORLD**
by CloudBees

Secure the Software Ecosystem Together!

- Verified builds make it possible to verify how artifacts have been produced
- We can formalize this system, to scale it to the rising number of open source dependencies in use
- Use and demand verifiable builds for your software!
 - Pick dependencies that build responsibly and verifiably
- As more OSS uses this, we can build a secure dependency graph together
 - Common formats, metadata locations, provenance verification
- Hop on board in the CDF, in Tekton Chains or in the new OSSF!

Thanks!

- dlorenc@google.com
- twitter.com/lorenc_dan
- github.com/tektoncd/chains
-


```
133 + Volumes: []corev1.Volume{{
134 +     Name: "${inputs.params.F00}",
135 +     VolumeSource: corev1.VolumeSource{
136 +         ConfigMap: &corev1.ConfigMapVolumeSource{
137 +             LocalObjectReference: corev1.LocalObjectReference{
138 +                 Name: "${inputs.params.F00}",
139 +             },
140 +         },
141 +     }, {
```


 **ImJasonH** 3 days ago Member

nit: indentation might be off here, I'd expect a `}`, `{` at the same indentation as `Volumes:` above.


 Reply...


Resolve conversation

```
142 +     Name: "some-secret",
143 +     VolumeSource: corev1.VolumeSource{
144 +         Secret: &corev1.SecretVolumeSource{
145 +             SecretName: "${inputs.params.F00}",
146 +         },
147 +     }, {
148 +     Name: "some-pvc",
149 +     VolumeSource: corev1.VolumeSource{
150 +         PersistentVolumeClaim: &corev1.PersistentVolumeClaimVolumeSource{
151 +             ClaimName: "${inputs.params.F00}",
152 +         },
153 +     },
154 + }, {
```

 **ImJasonH** 3 days ago Member

same nit here: equally-indented `}` s just look wrong.

 Reply...

plumbing to get latest fixes 


fix issues when the diff is too big.



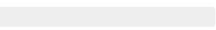
y: Vincent Demeester <vdemeest@redhat.com>

)

ter authored and **tekton-robot** committed 4 days ago 1 parent [49ec715](#) commit [efbcf9d01](#)


anged files with 49 additions and 16 deletions.


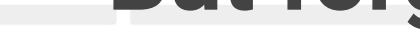

Gopkg.lock 



[Load diff](#)


Some generated files are not rendered by default. [Learn more.](#)


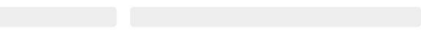
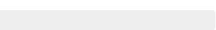
vendor/github.com/tektoncd/plumbing/scripts/library.sh 



[Load diff](#)

Some generated files are not rendered by default. [Learn more.](#)

vendor/github.com/tektoncd/plumbing/scripts/presubmit-tests.sh 



[Load diff](#)

Some generated files are not rendered by default. [Learn more.](#)

But forgot for open source...