in-toto -- Securing the whole software supply chain

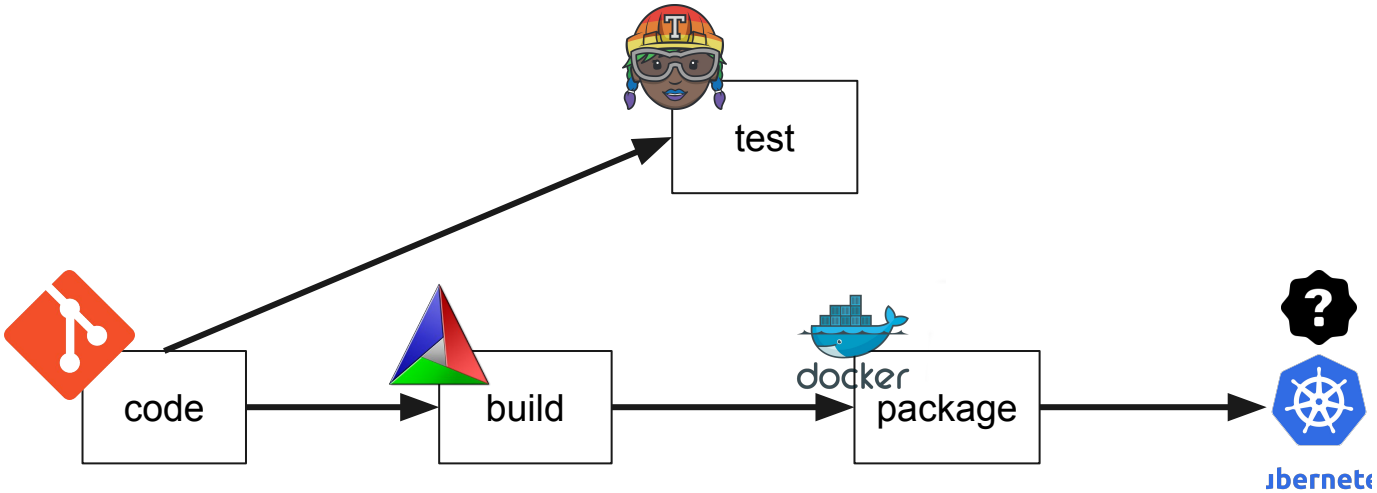
Santiago Torres-Arias, Hammad Afzali,
Lukas Pühringer, Reza Curtmola, Justin Cappos

NYU TANDON SCHOOL OF ENGINEERING

NJIT New Jersey Institute of Technology

How is software made?

A stylized software supply chain




Slashdot
Submit
Build
Entertainment
Technology
IMPORTANT JUNIPER SECURITY ANNOUNCEMENT
CUSTOMER UPDA
Administrative Acces
Decryption (CVE-201
6.3.0r20.
We strongly recomm
with the highest pr
POSTED BY BOB
Juniper is committ
customers aware o
devices running S
During a recent in
could allow a know
decrypt VPN conn
into the matter, an
ScreenOS.
At this time, we ha
strongly recommer
highest priority.
THE H OPEN
Open
In association with
Last 7 days
News Archive
Features
02 December 2010, 19:07
« previous | next »
Back door in ProFTPD FTP server
Unknown attackers penetrated the server hosting the open source ProFTPD FTP server project and concealed a back door in the source code. The back door provides the attackers with complete access to systems on which the modified version of the server has been installed. On installation, the modified version informs the group behind the back door by contacting an IP address in the Saudi Arabia area. Entering the command 'HELP ACIDBITCHEZ' results in the modified server displaying a root shell.
exploit that to root access is currently unknown and is being investigated," according to kernel.org.
Gets
& Freedom -
Slashdot's
re.
536
all-
have started
attack is not
tion, it is
go.
IDA
ged to

FreeBSD Project Discloses Security Breach Via Stolen SSH Key

86

Posted by timothy on Saturday November 17, 2012 @09:22AM from the happy-transparency dept.

An anonymous reader writes

> "Following recent compromises of the Linux kernel.org and Sourceforge, the FreeBSD Project is now reporting that several machines have been broken into. After a brief outage, ftp.FreeBSD.org and other services appear to be back. The project announcement states that some deprecated services (e.g., cvsup) may be removed rather than restored. Users are advised to check for packages downloaded between certain dates and replace them, although not because known trojans have been found, but rather because the project has not yet been able to confirm that they could not exist. Apparently initial access was via a stolen SSH key, but fortunately the project's clusters were partitioned so that the effects were limited. The announcement contains more detailed information — and we are left wondering, would proprietary companies that get broken into so forthcoming? Should they be?"

bsd freebsd it

←

Old Electric-Car Batteries Put into Service For Home Energy Storage

You may like to read:

Outsourced IT Workers Ask Sen Feinstein For Help, Get Form Letter in Return

'DNC Hacker' Unmasked: He Really Works for

→

Anonymous Attacks Israeli Websites In Response To IDF Operation In Gaza

Ian Duffy

Rants of a software engineer

Azure bug bounty Pwning Red Hat Enterprise Linux

Nov 26, 2016 · 5 minute read · 1 Comment

Cloud Security Azure

TL;DR Acquired administrator level access to all of the Microsoft Azure managed Red Hat Update Infrastructure that supplies all the packages for all Red Hat Enterprise Linux instances booted from the Azure marketplace.

I was tasked with creating a machine image of Red Hat Enterprise Linux that was compliant to the Security Technical Implementation guide defined by the Department of Defense.

This machine image was to be used for both Amazon Web Services and Microsoft Azure. Both of which offer marketplace images which had a metered billing pricing model[1][2]. Ideally, I wanted my custom image to be billed

mirror, we are conducting additional validation to confirm and will provide update once this process concludes. The mirror remains out of rotation.

Nerves Rattled By Highly Suspicious Windows Update Delivered Worldwide

Microsoft 217

Posted by samzenpus on Wednesday September 30, 2015 @02:29PM from the proceed-with-caution dept.

An anonymous reader writes:

> If you're using Windows 7 you might want to be careful about which updates you install. Users on Windows forums are worried about a new "important" update that looks a little suspect. Ars reports: "'Clearly there's something that's delivered into the [Windows Update] queue that's trusted,' Kenneth White, a Washington DC-based security researcher, told Ars after contacting some of the Windows users who received the suspicious update. 'For someone to compromise the Windows Update server, that's a pretty serious vector. I don't raise the alarm very often but this has just enough characteristics of something pretty serious that I think it's worth looking at.'"

UPDATE: Microsoft says there's nothing to worry about, the company "incorrectly published a test update."

windows security microsoft

← AdBlock Plus To Introduce Independent Board To Oversee Acceptable Ads Program

You may like to read:

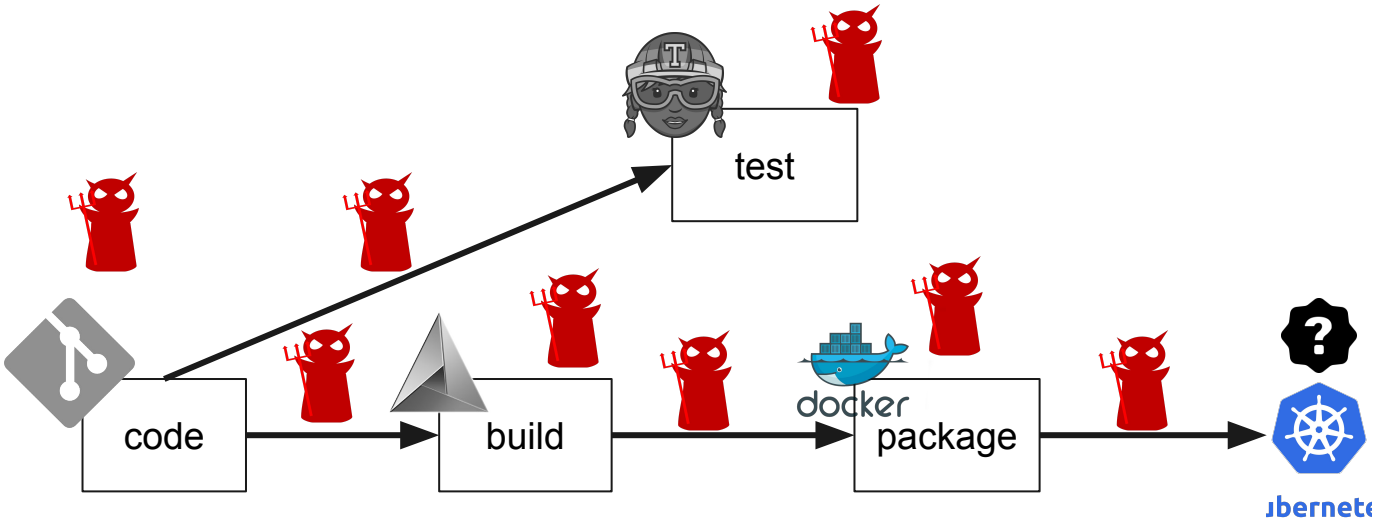
Silicon Valley Investors Call For California To Secede From the US After Trump Win

Twitters Says It Will Ban Trump If He Breaks

→ Will 'Chip and Pin' Credit Card Technology Really Increase Security? (Video)

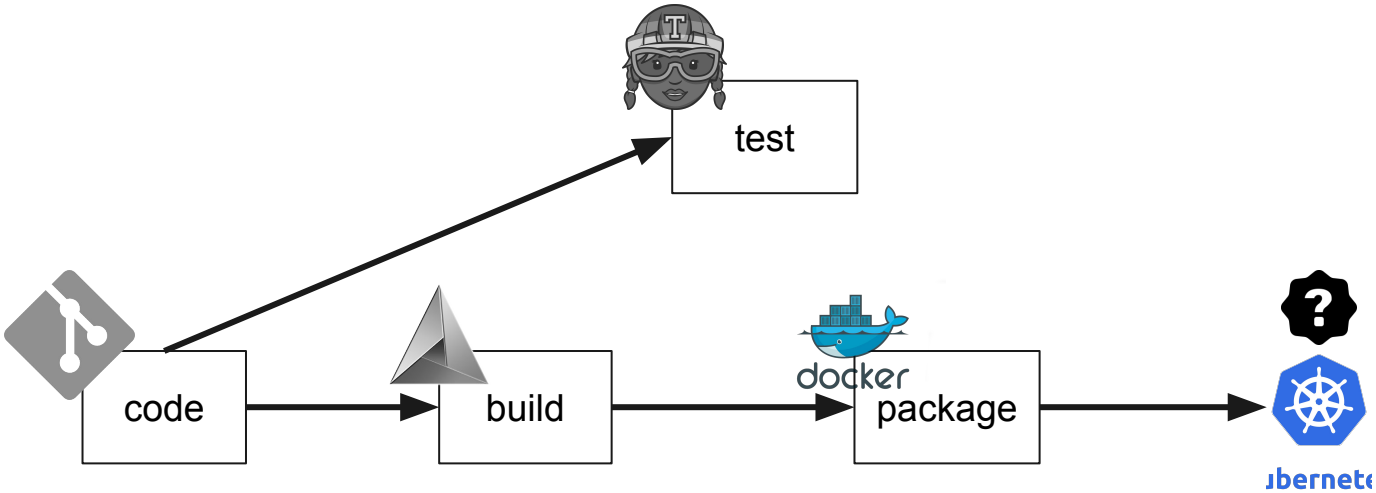
Systemd Rolls Out Its Own Mount Tool

Attackers can hack the software supply chain



How can we fix this?

Many good point solutions



[arch-general] git undetectable tag replacement?

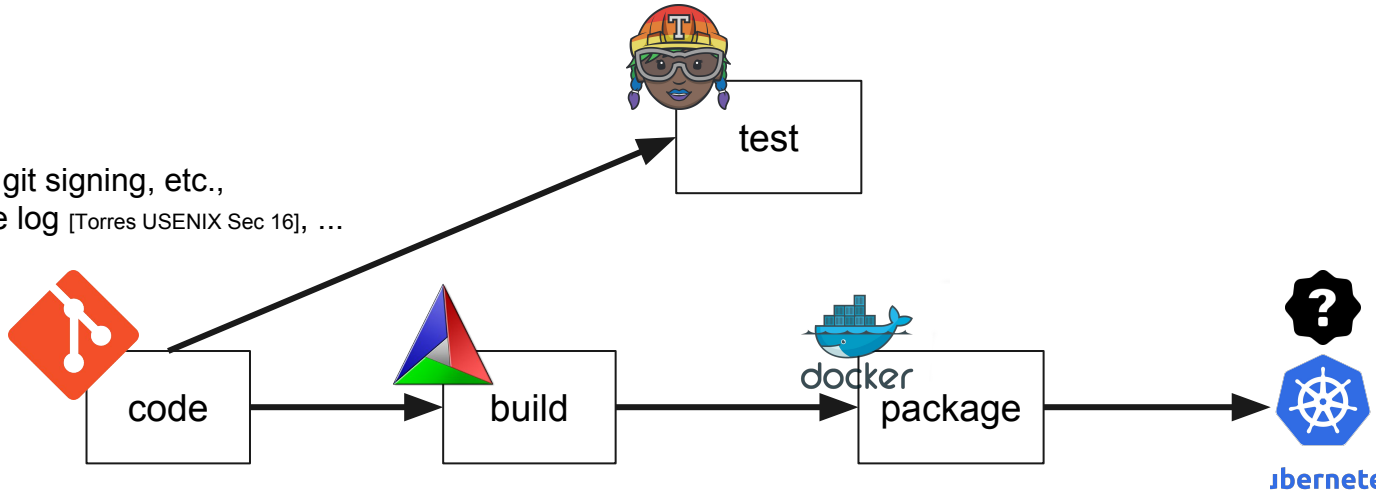
Eli Schwartz eschwartz93 at gmail.com
Mon Jul 3 02:14:00 UTC 2017

- Previous message (by thread): [arch-general] Sébastien Luttringer and Tobias Powalowski
- Next message (by thread): [arch-general] git undetectable tag replacement? (Was: Sébastien Luttringer an
- **Messages sorted by:** [date] [thread] [subject] [author]

```
On 07/02/2017 07:34 PM, Ismael Bouya wrote:
> (Sun, Jul 02, 2017 at 07:22:23PM -0400) Eli Schwartz via arch-general :
>> Okay, this I am genuinely curious about.
>>
>> In what circumstances can I have:
>> - the systemd repository cloned over the git:// protocol
>> - an annotated tag for systemd v233 signed by Lennart Poettering.
>> - an annotated tag for systemd v232 signed by Lennart Poettering.
>> - a man in the middle attack
>> - `git verify-tag --raw v233` reports a GOODSIG with a VALIDSIG
>>   ${fingerprint} that matches with Lennart's known GPG fingerprint as
>>   recorded in validpgpkeys
>>
>> And as a result, when I run the git command `git checkout
>> refs/tags/v233`, I am tricked into getting v232 instead which contains a
>> vulnerability.
>
> Until there, it's exactly the topic of the presentation linked by
> Nicohood

So I was under the impression that git tags encode the tagname in the
actual blob, and I didn't see how that attack (rooted in the basic
nature of a branch as a lightweight, mutable, *pushable* pointer to a
commit) was supposed to work unless of course it was talking about a
lightweight tag (which is not really meant for public/permanent use)...

Having actually tested this out, I find myself quite bewildered.

Because, git *does* encode the tagname in the blob, like I thought.
And... you *can* simply copy .git/refs/tags/tagname to create a fake
tag, and then you see something quite bewildering:
```

Many good point solutions

→ SVN, CVS, git signing, etc.,
reference state log [Torres USENIX Sec 16], ...



→ TPMs, HSMs,verifiable compilers,
reproducible builds, ...


CHRYSLER


OTAinfo
SECURED OVER THE AIR INFORMATION TRANSFER


FIAT


Advanced
Telematic
SYSTEMS


here

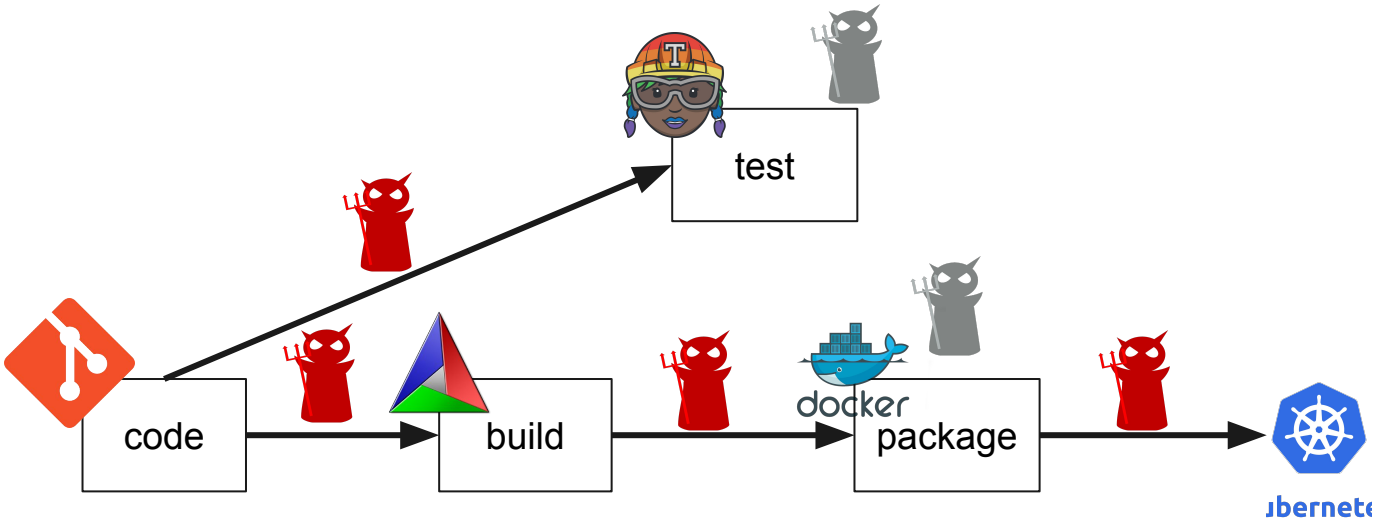

Airbiquity®


AUTOMOTIVE
GRADE LINUX


GENIVI®

Fixed?

Gaps between steps? Compliance?



We want to secure the complete software supply chain!

→ Verifiably define the steps of the software supply chain

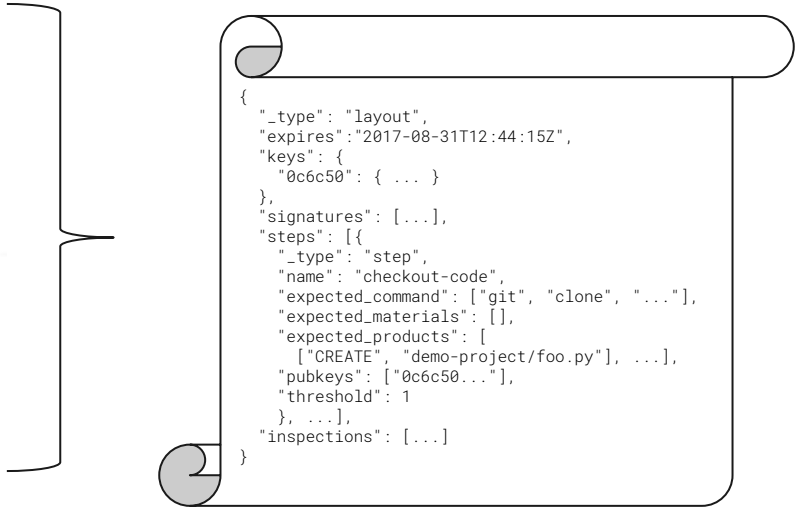
→ Verifiably define the authorized actors

→ Guarantee that everything happens according to definition, and nothing else

in-toto -- Layout

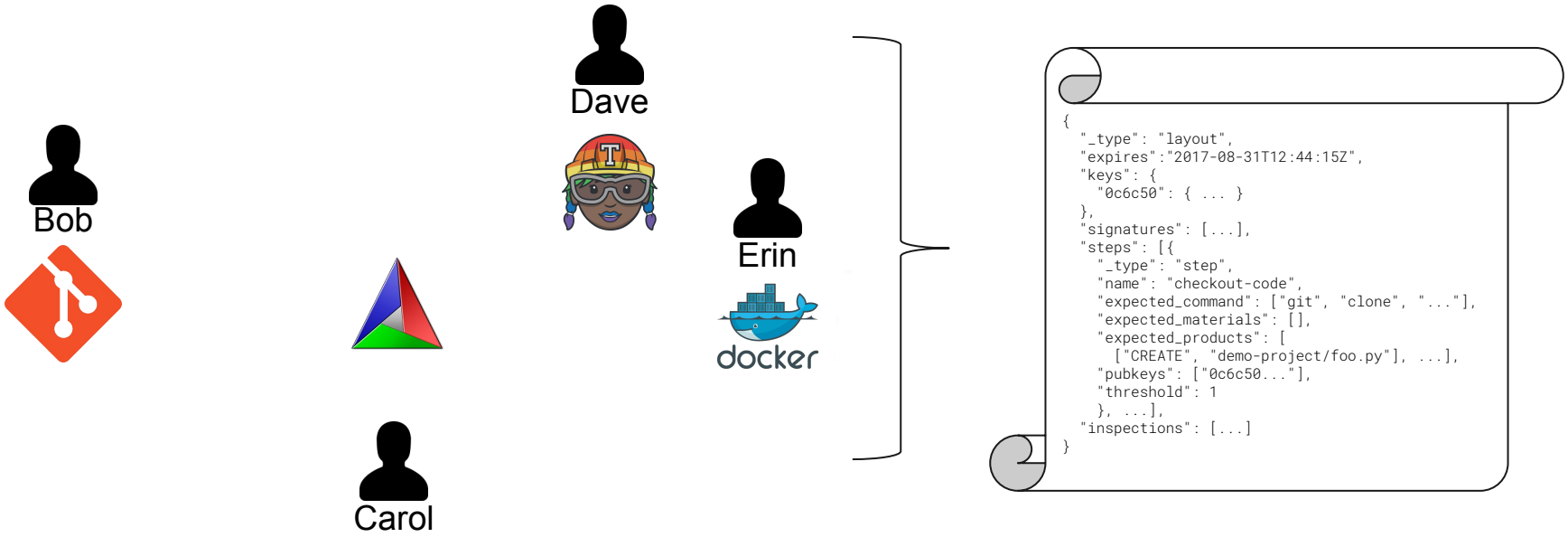
```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

in-toto -- Layout -- Steps

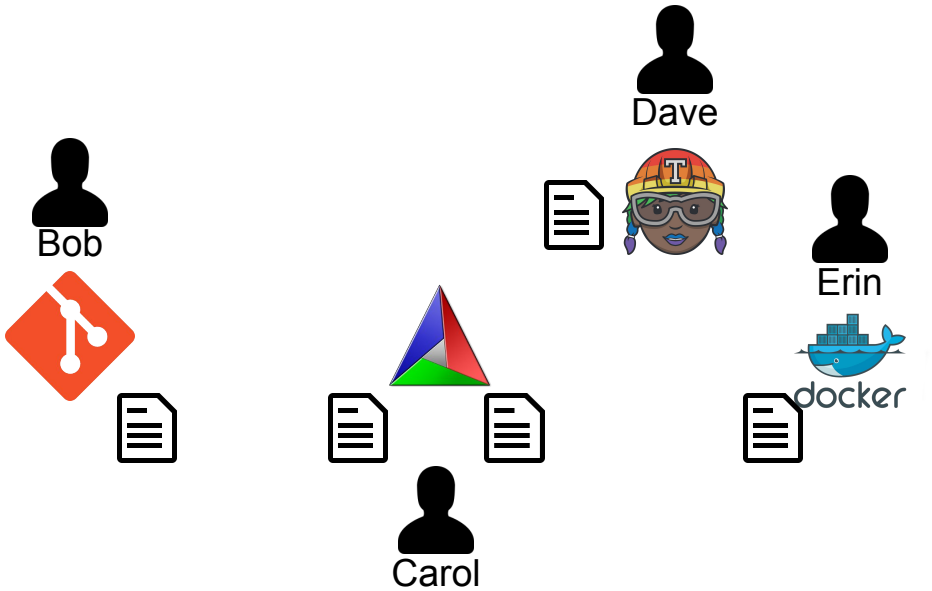


```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

in-toto -- Layout -- Functionaries

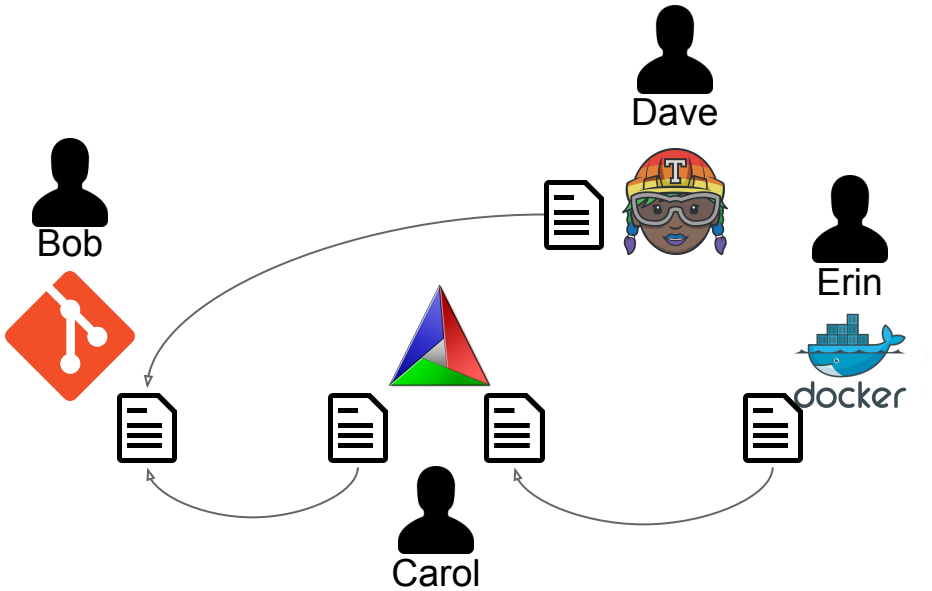


in-toto -- Layout -- Materials/Products



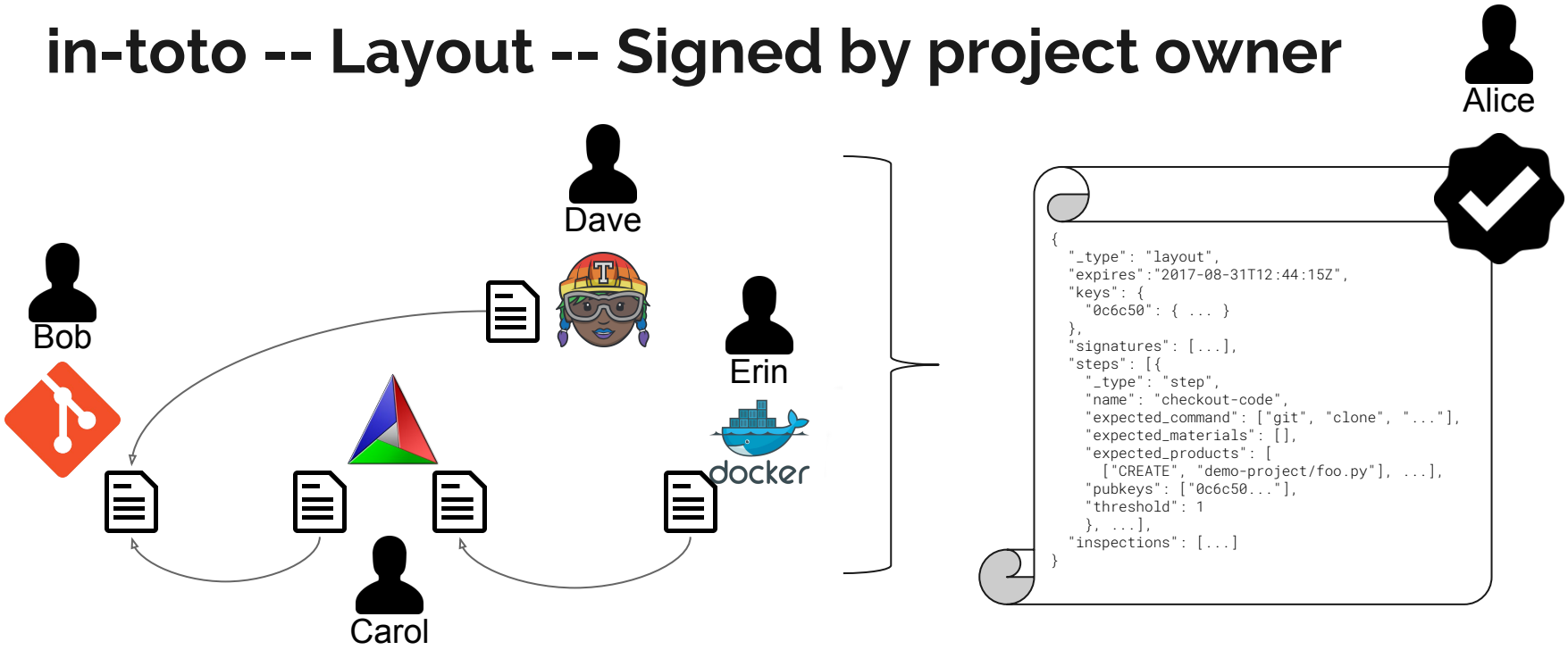

```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

in-toto -- Layout -- Artifact rules



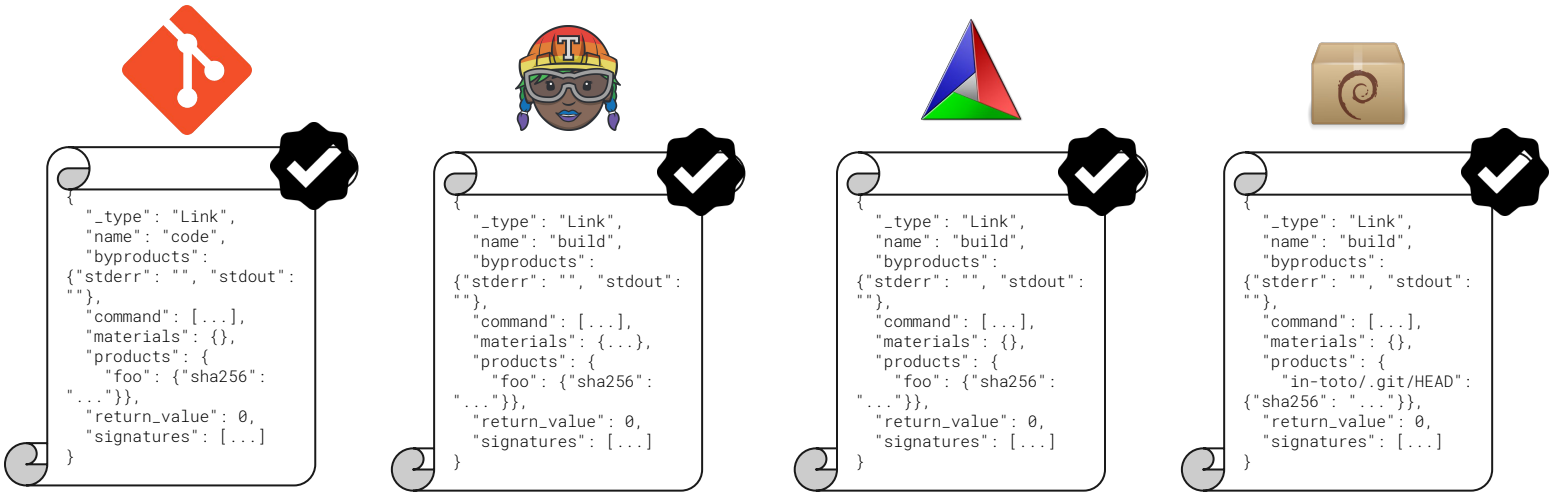

```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

in-toto -- Layout -- Signed by project owner



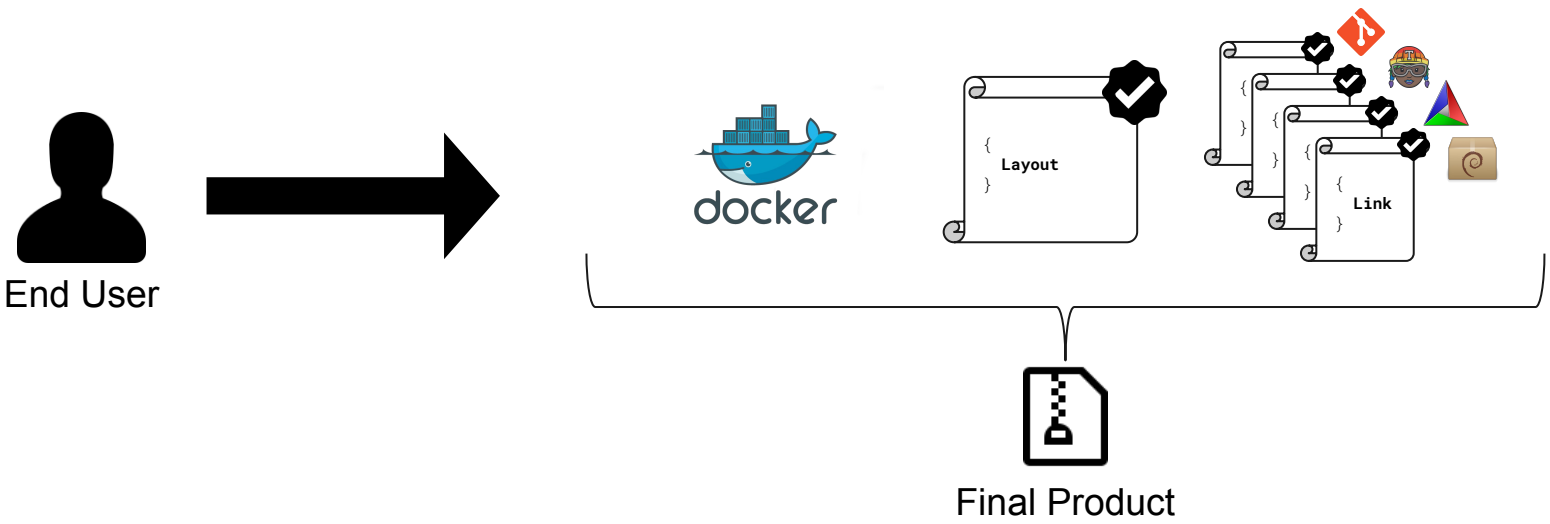
in-toto -- Links -- Signed evidence for each step

```
$ in-toto-run -- ./do-the-supply-chain-step
```



in-toto -- Verification

```
$ in-toto-verify --layout <layout> --key <pub key>
```



Real-world impact

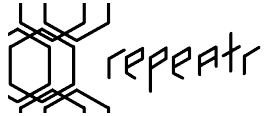
Related work

GOVREADY

SPDX

repeatr

Integration efforts



Other tools to protect your cloud native ecosystem

Vulnerability scanners

→ Scan containers for known Common Vulnerability Enumeration (CVE) signatures:

⇒ black duck, twistlock, veracode (now CA), aquascanner, Xray

Static analyzers

→ Scan containers, binaries, and sources to identify memory leaks, potential vulnerabilities etc.

- ⇒ containers: kubesec
- ⇒ binaries: valgrind, coccinelle
- ⇒ source: mostly language-specific (e.g., bandit for python)

Pipeline Managers

→ Orchestrate development of artifacts before admitting into the cluster

- ⇒ JenkinsCI
- ⇒ CircleCI
- ⇒ HerokuFlow

Artifact Metadata stores

→ Store metadata about artifacts as they travel through the pipeline

⇒ Repeatr: artifact memoization framework

⇒ Grafeas: a cloud native supply chain metadata store

⇒ Sparcs: a blockchain metadata store

Artifact validation

→ Verify artifacts before they are deployed into your cluster

⇒ k8s admission controllers

- Binary authorization

→ Notary, TUF

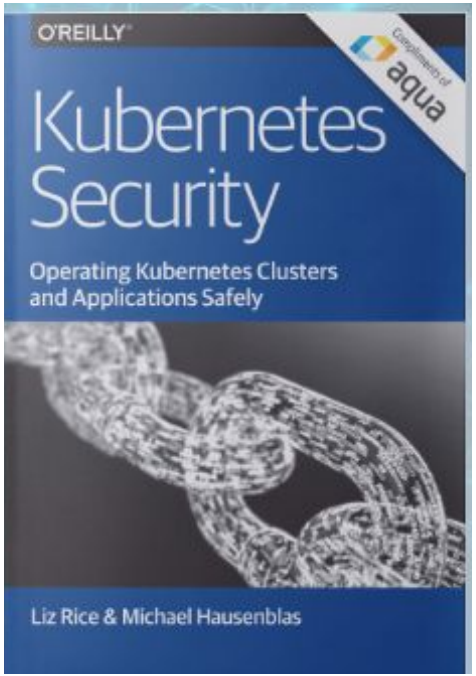
Aside: free book on cloud native security!

→ Just released

→ Free!

→ Covers in-toto, tuf, and other tools out there

→ https://info.aquasec.com/kubernetes-security



Conclusion

→ Securing the software supply chain is important

→ in-toto is the first tool to secure the whole supply chain

⇒ Many ‘simple-but-subtly-wrong’ approaches

→ Try out in-toto!

⇒ https://in-toto.io

Thank You! Questions?

https://in-toto.io/
santiago@nyu.edu
github: @santiagotorres
twitter: @torresariass

DEMO

→ Alice is the project owner, and she will create a supply chain

→ Functionaries:

Bob will clone the repository, and mark the new release

Carl packages the project

→ The end user will verify that the steps described by Alice are performed correctly by Bob and Carl

Links / Language / Matching

in-toto -- Layout

→ Describes the steps to be performed by functionaries and the inspections to be performed by the end user

→ Contains other useful information (e.g., an expiration date)

→ Signed by the project owner, for authenticity.

→ Lists the public keys that each functionary should use to sign link metadata

in-toto -- Steps

→ Steps are contained in the layout and describe an operation in the supply chain

→ Performed by functionaries, who will provide signed link metadata as proof that the step was carried out

→ Contains a list of rules that will limit the actions that the functionary can perform, and that link steps between each other

in-toto -- Artifact rules -- Limiting trust

→ Trust within in a step is limited using rules that apply to artifacts (materials, products)

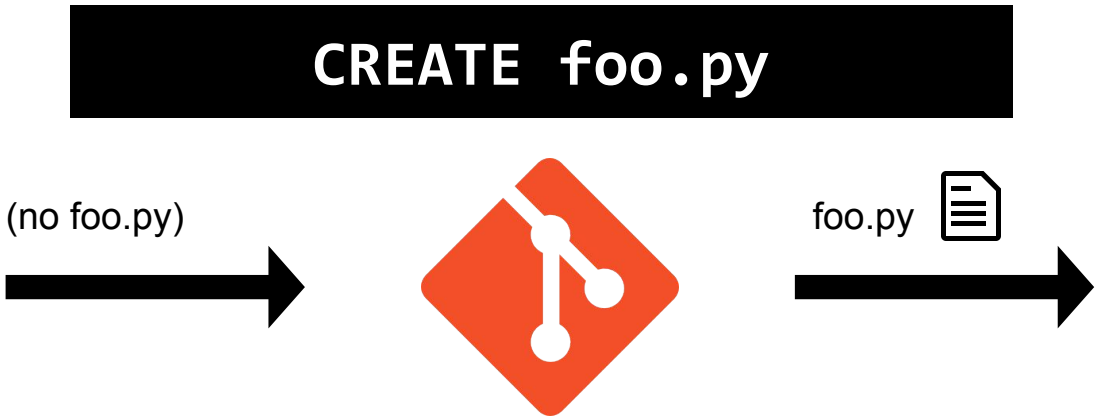
```
CREATE foo.py
```

(this functionary **can create** a file called **foo.py)**

→ Principle of least privilege --- limit the amount of privilege a functionary needs in order to perform his or her duties.

in-toto -- Artifact rules -- Limiting trust

→ Trust within in a step is limited using rules that apply to artifacts (materials, products)



in-toto -- Artifact rules -- Linking steps

→ Steps are also linked together using rules

```
MATCH foo.py WITH PRODUCTS FROM tag-release
```

(this step's **foo.py** must **match** the one in tag-release)

→ The MATCH rule provides a way to define the source of every artifact within the supply chain.

in-toto -- Artifact rules -- Linking steps

→ Steps are also linked together using rules

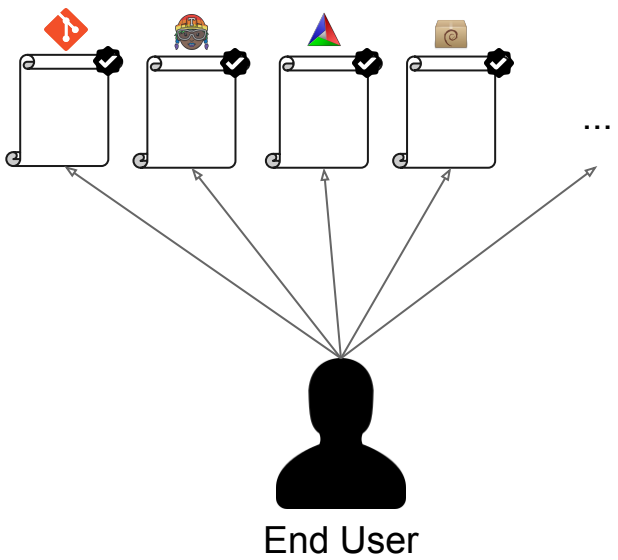
```
MATCH foo.py WITH PRODUCTS FROM tag-release
```

in-toto -- Inspections

→ Used to verify metadata from within a step

→ Performed by the client

→ Uses link + additional (app specific) metadata and the layout



in-toto -- Inspections (example)

→ Verify git reference state log

⇒ Ensure all commits are signed

⇒ Verify only people allowed merged to master