

Supply Chain Integrity

Andrew Hardie

Infrastructure Services Lead, R3

Chairman, BCS DevSecOps Expert Group

Longstanding London DevOps Regular

Presented at the meeting on 9th December 2021

Threat Landscape

- Like weeds, the threats are everywhere...



- And, new ones keep popping up all the time...
- So, what to think, what to do, how to sleep at night, and so on...?

Ground Rules

- Trust nothing
- Trust nobody
- Check everything
- Check everyone
- Assume nothing

Ground Rules

- See, easy!
- I can go home now... 😊

Scope (at least, for here & now...)

- Everything from code commit to running production deployments
- All the code and tools used to achieve that – your DevOps pipeline
- All the components you import from external sources for that
- No further here - direct attacks on running instances: out of scope
- But, it's still a lot...!

So, where to start...?

- Back to the meadow...



- Which plants do you not recognize?
- Which plants do you not want?

Threat Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Lateral Movement	Exfiltration	Impact
Supply Chain Compromise on CI/CD	Modify CI/CD Configuration	Compromise CI/CD Server	Get credential for Deployment(CD) on CI stage	Add Approver using Admin permission	Dumping Env Variables in CI/CD	Exploitation of Remote Services	Exfiltrate data in Production environment	Denial of Services
Valid Account of Git Repository (Personal Token, SSH key, Login password, Browser Cookie)	Inject code to IaC configuration	Implant CI/CD runner images	Privileged Escalation and compromise other CI/CD pipeline	Bypass Review	Access to Cloud Metadata	(Monorepo) Get credential of different folder's context	Clone Git Repositories	
Valid Account of CI/CD Service (Personal Token, Login password, Browser Cookie)	Inject code to source code	Modify CI/CD Configuration		Access to Secret Manager from CI/CD kicked by different repository	Read credentials file	Privileged Escalation and compromise other CI/CD pipeline		
Valid Admin account of Server hosting Git Repository	Supply Chain Compromise on CI/CD	Inject code to IaC configuration		Modify Caches of CI/CD	Get credential from CI/CD Admin Console			
	Inject bad dependency	Inject code to source code		Implant CI/CD runner images				
	SSH to CI/CD pipelines	Inject bad dependency						
	Modify the configuration of Production environment							
	Deploy modified applications or server images to production environment							

Source: <https://github.com/rung/threat-matrix-cicd>

Begin at the beginning...

- Who has access to the source code repo?
- Remember, in the “Everything as Code” world, trust shifts from who has root access to running systems to who has access to the code that will run as root, i.e. your infrastructure code!
- Are you controlling user **write** access to your Infrastructure as Code repos tightly enough? (many processes may need read-only access)
- Can those users prove to the system who they are? E.g. hardware token?
- Do you have signed tags/commits/merges?
- And, protected branches to enforce rules?
- And trusted timestamping?

So, you have a code commit/merge, now what?

- Third-party components problem, & local trust problem (SolarWinds)
- Modern code relies heavily on third-party libraries, plugins, etc.
- Much “application code” now can be little more than some business logic applied on top of libraries providing the real functionality
- The more “rich functionality” component provides, the bigger the risk
- Build process incorporates those components into the artefact build
- Hard/impossible to determine their safety *a priori* (i.e. theoretically)
- Have to rely on reported vulnerability discoveries: CVE libraries, etc.

So, you have security in the build components

- Now, what about the build tools and the processes they run...?
- Can you be sure they haven't been compromised?
- Have you used signatures to verify the binaries?
- Did you use source code protection principles for your pipeline code?
- If you are making a binary or library or runtime code, e.g. Java JAR, then that's about as far as you can go to assure artefact integrity
- If you're packaging your code into a container, you have more fun!
- CVE check of all imported container layers; layer substitution, etc.

So, you have artefacts you are confident of...

- Remember the artefact repo mantra: “only tested artefacts that are ready to be run at any time”; now you can add “security assured”
- But, that is assured at the time they were built – may not still be...
- Keep checking them, or keep re-building them with checks included
- Now, you have to deploy them...
- Same concerns as for the artefact build process re code & toolchain
- But, now you have environments to worry about, and policy
- Which processes are allowed to deploy which artefacts into which environments? (non-humans assumed as this is automated, right?)

So, your code is running in <some-random-environment>

- Now what? Promote towards production...
- How will environment promotion be policed and initiated?
- Is a manual step required to auth production deploy? Same concerns as a code commit or a Pull Request approval if using GitOps principles
- Same concerns as for the artefact build process re code & toolchain
- Plus, securing access to the target environments (e.g. restrict to local processes and actors that can be verified tightly for deployments)
- So, you've got it to production as carefully as you could...
- Happy now?

Not hardly!

- Remember that CVE reports are at a point in time; things can change!
- So, you can re-scan your artefact repo and if a new CVE shows up, then what do you do? Where is that artefact or component running? Which artefact repos, environments and functionalities are affected?
- Similar problems:
 - If a third-party component author or their repo is compromised, where are those components running? Hiding somewhere up your “route to live”?
 - If suspicion falls on one of your developers, where is their code running?
 - And, if you have deployed any of the possibly tainted code to a client environment, now what...?!

Observability could be your friend

- A tool is needed that can:
 - Maintain an audit trail of
 - Code authors
 - Third party components
 - Third party suppliers
 - Intermediate repos (e.g. CDNs)
 - right the way through the build and deploy pipelines to live production
 - and all the stops along the way...
- Anyone know one...?
- Carefully crafted log entries could be the answer, with rich tagging

And your next headache..?

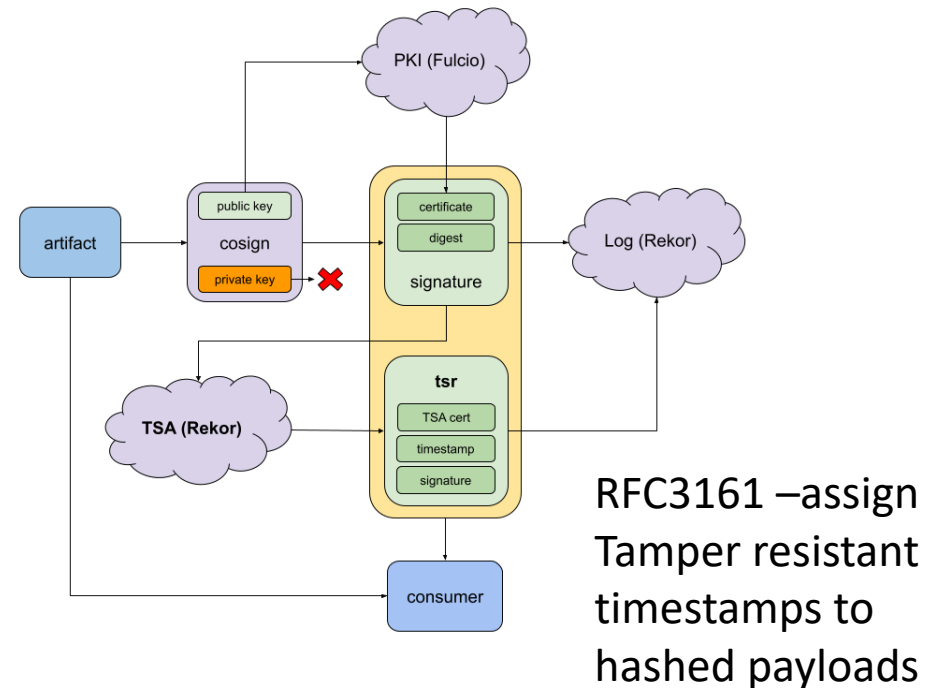
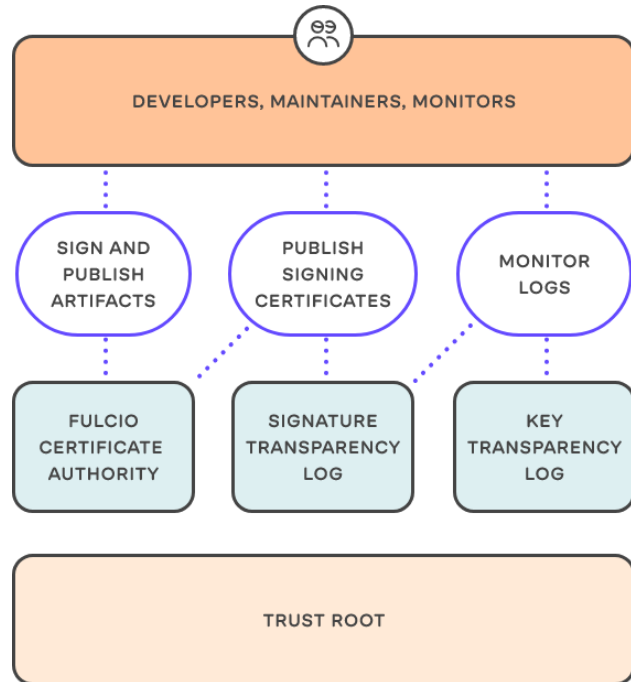
- Your clients!
- You sell them stuff...
 - They want to be confident you've done your best to ensure and verify that
 - You want to be sure you have and that they are confident...
- But, hard to do this alone, because of all the third-party components
- What to do about the upstream long tail all the way back to origins?
- What's going on out there to address all this...?
- Let's have a look...

Active areas

- April 2021 was National Supply Chain Integrity month!
 - (just four months after the notorious SolarWinds compromise – and the FBI warned about SCI attacks in Feb 2020!)
 - CISA (US Cybersecurity and Infrastructure Security Agency)
 - DHS (US Department of Homeland Security)
 - CDSE (US DoD Centre for the Development of Security Excellence)
 - NCSC (US National Counterintelligence and Security Center – confusingly, not our National Cyber Security Centre...)
- At (our) NCSC – Principles of Supply Chain Security (note: principles!)
 - 12 principles, grouped into Understand, Control, Check, Continuously Improve
- But what about Government & industry initiatives...?

Open Source and Community initiatives

- Sigstore (<https://www.sigstore.dev/>)
“A new standard for signing, verifying and protecting software”
Supported by Google, RedHat, Linux Foundation and Purdue University



Open Source and Community initiatives

- in-toto – “A framework to secure the integrity of software supply chains”

<https://in-toto.io/>

Open metadata standard that you can implement in your software's supply chain toolchain Project is collaborating with Git, Debian, ArchLinux, Docker, SUSE, ControlPlane & Datadog Supported by US National Science Foundation, DARPA and USAF Research Laboratory

- The Update Framework – “A framework for securing software update systems”

<https://theupdateframework.io/>

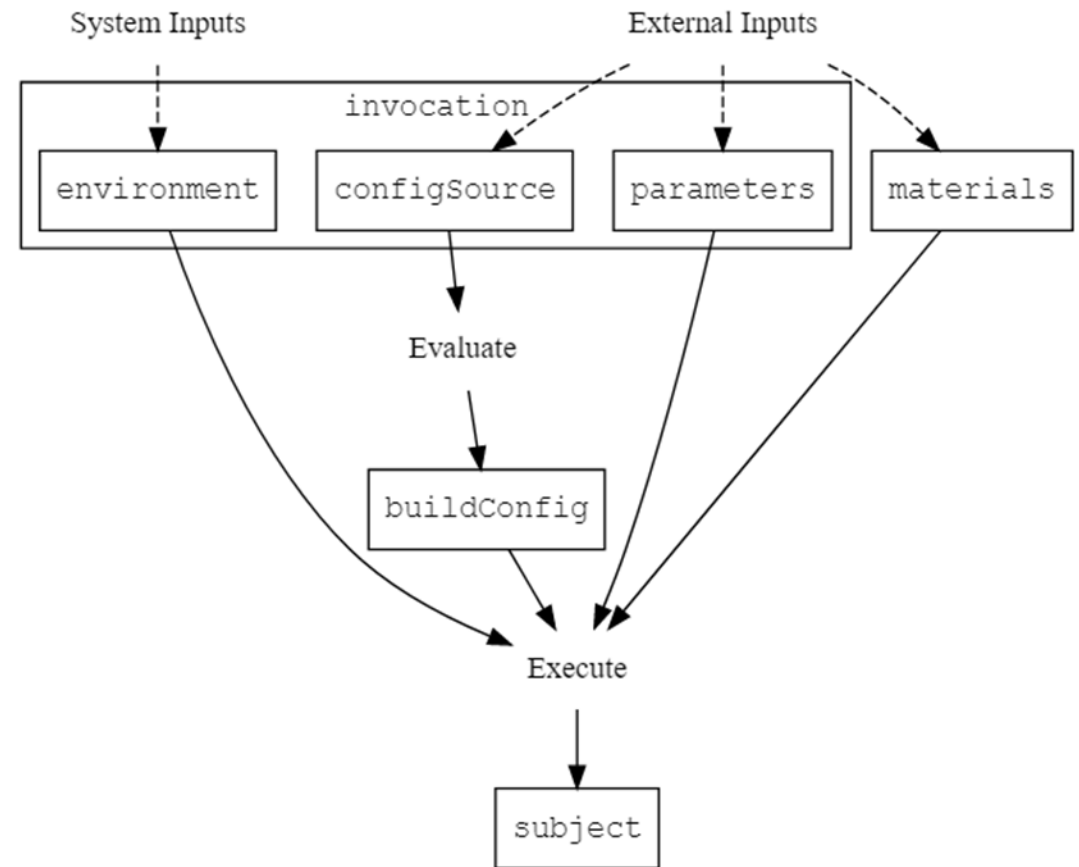
The Update Framework (TUF) helps developers maintain the security of software update systems, providing protection even against attackers that compromise the repository or signing keys. TUF provides a flexible framework and specification that developers can adopt into any software update system.

Managed by Linux Foundation and the CNCF


- in-toto and TUF can work together to deliver updates and their corresponding in-toto metadata

Doing the Google Salsa (<https://slsa.dev>)

- SLISA - Supply-chain Levels for Software Artifacts (Google, June 2021)
- Key item is establishing Provenance – an attestation that some entity (**builder**) produced one of more artifacts (the **subject** of an in-toto attestation **statement**) by executing some **invocation** using some other artifacts as input (**materials**).



Start here...

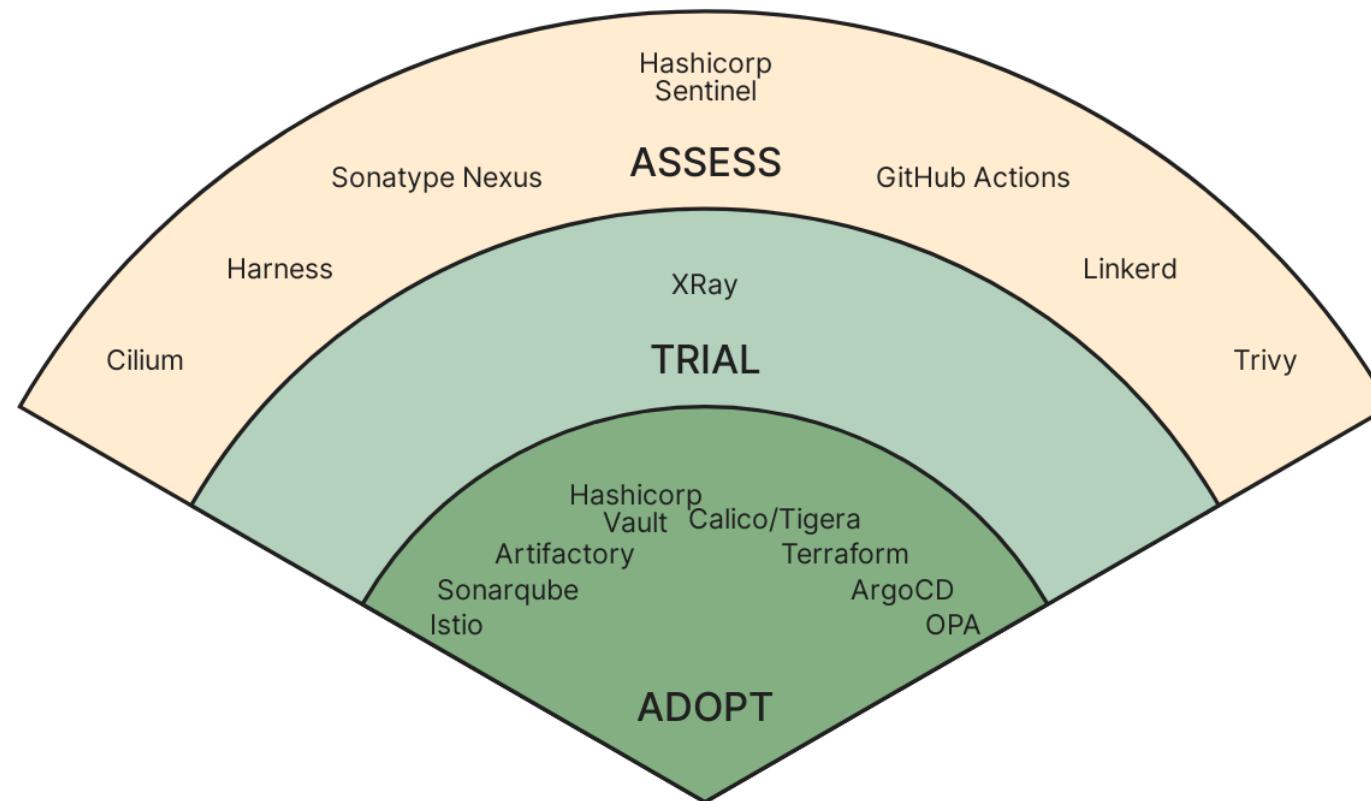
- CNCF – Software Supply Chain Best Practices
https://project.linuxfoundation.org/hubfs/CNCF_SSCP_v1.pdf
45 pages of excellent advice you must read! 
- NCSC (UK) Supply Chain Security Guidance
<https://www.ncsc.gov.uk/collection/supply-chain-security>
“Proposing a series of 12 principles...”
- CISA (US) Supply Chain Risk Management Essentials
https://www.cisa.gov/sites/default/files/publications/ict_scrm_essentials_508.pdf
Two page “Leader’s Guide”, summarizing six essential steps:
Identify, Manage, Assess, Know, Verify, Evaluate

CNCF Technology Radar

Report of a point in time, not predictions!

CNCF Technology Radar

DevSecOps, September 2021



Are we there yet, now?

- Not quite, but it's getting close... 😊
- Don't forget:
 - There is no endpoint, it's an ever-ongoing battle with threats & actors
 - You are having to project security & assurance outside your own boundaries
 - So many things to consider, ***on top of "normal" IT security stuff...***
 - This stuff is complicated!

My head hurts...

- I feel like lying down in this warm meadow with the sun on my face...



- But, I might manage a few (simple) questions... 😊