controlplane

Untrusted Execution:
Attacking the Cloud Native Supply Chain

Andrew Martin
@sublimino

Francesco Beltramini
@d1gital_f

Roles

- CEO, ControlPlane
- Co-chair, CNCF TAG Security
- CISO, OpenUK

Works

- O'Reilly
  - *Hacking Kubernetes*
  - *Kubernetes Threat Modelling*
- SANS SEC584
  - *Cloud Native Security: Defending Containers and K8s*
- Whitepapers, training

Training

- Hashicorp
- Docker

I'm:

- **Andy**
- **Dev**-like
- **Sec**-ish
- **Ops**-y

🔥🔥 **FREE DOWNLOAD:** https://control-plane.io/hackingkubernetes 🔥🔥

Hacking Kubernetes

With such favourites as:

- Kubernetes YOLO moves
- Threat models ‘n’ attack trees
- Running Kubernetes “securely”
- Exploiting your Kuberneteses
- The future state of containers...

@controlplaneio

🔥🔥 **FREE DOWNLOAD:** https://control-plane.io/hackingkubernetes 🔥🔥

controlplane

About ControlPlane

A Cloud Native security consultancy established in 2017. Our diverse culture empowers and develops individuals with talent and integrity.

We regularly participate in the largest worldwide security conferences.

- Security specialists in cloud, Kubernetes, and containers
- Clients include government, financial services, and regulated industries
- 50 people across the UK, Europe, North America and APAC

Agenda

- (Software) Supply Chain Problem
  Simple Case study: Applications and Supply Chain attack
- Problem Space
- Threat Modelling Supply Chains
- Securing Supply Chains

TL;DR

- Where does our code come from?
- Where does it go?
- What do we ingest?
- How it goes wrong?
- What are we going to do about it?

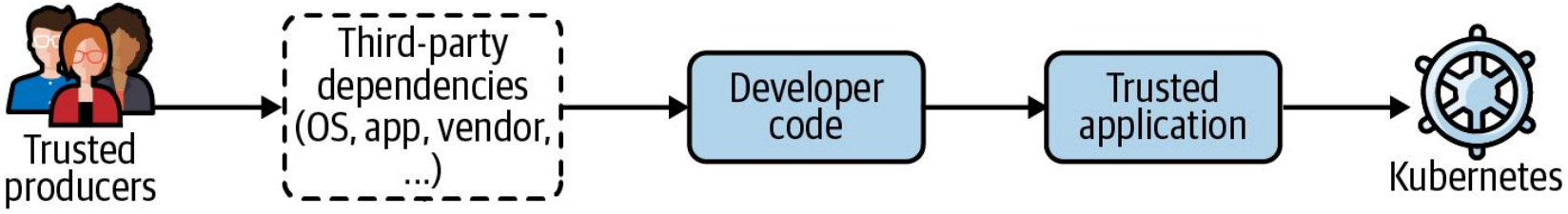
What's the Supply Chain Problem?

What is a supply chain?

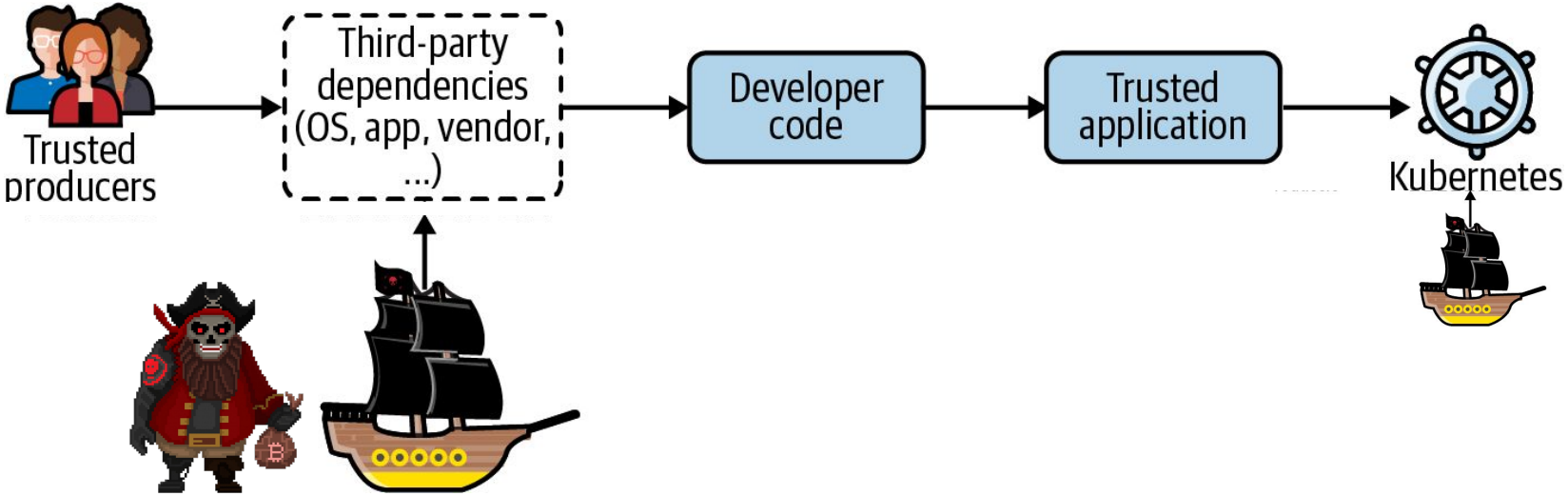
Anything that we depend upon

- e.g., the military need to know where all their hardware and software comes from and who builds them, to protect against state attacks
- e.g., pharmaceutical companies likewise need to know the provenance of their ingredients

Supply Chain Attacks: Shipping Code

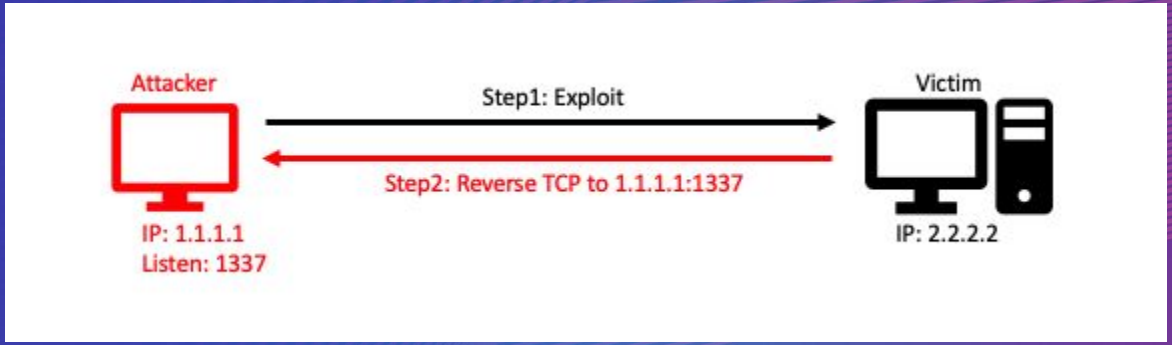


Supply Chain Attacks: Attacker's Goal

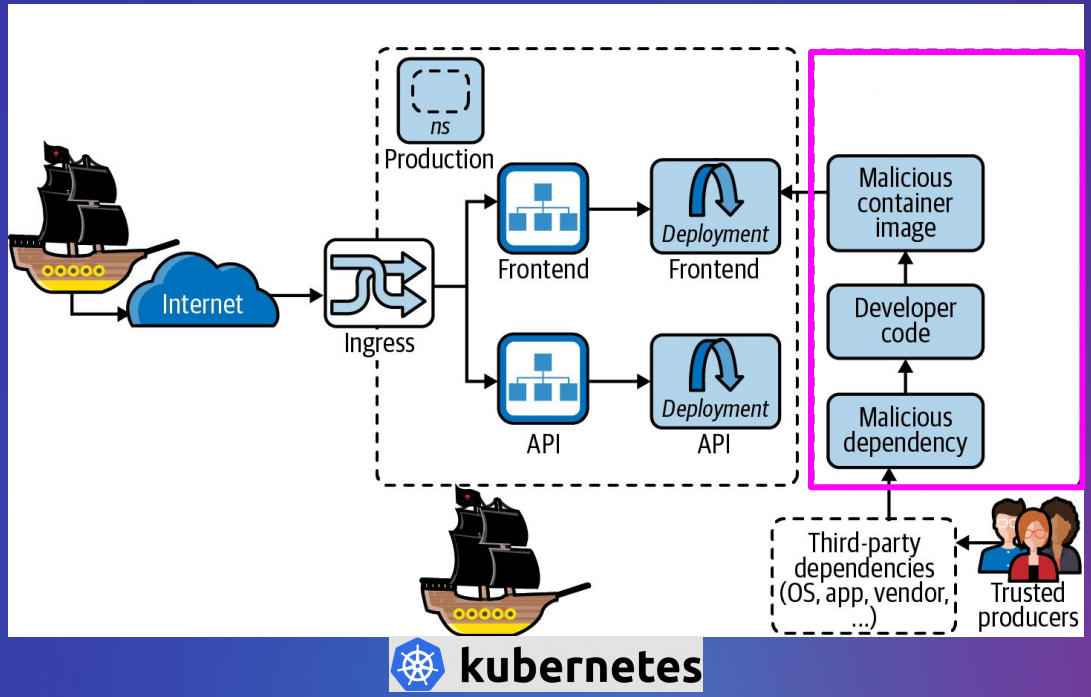


Attacking from Behind the Firewall: Reverse Shells

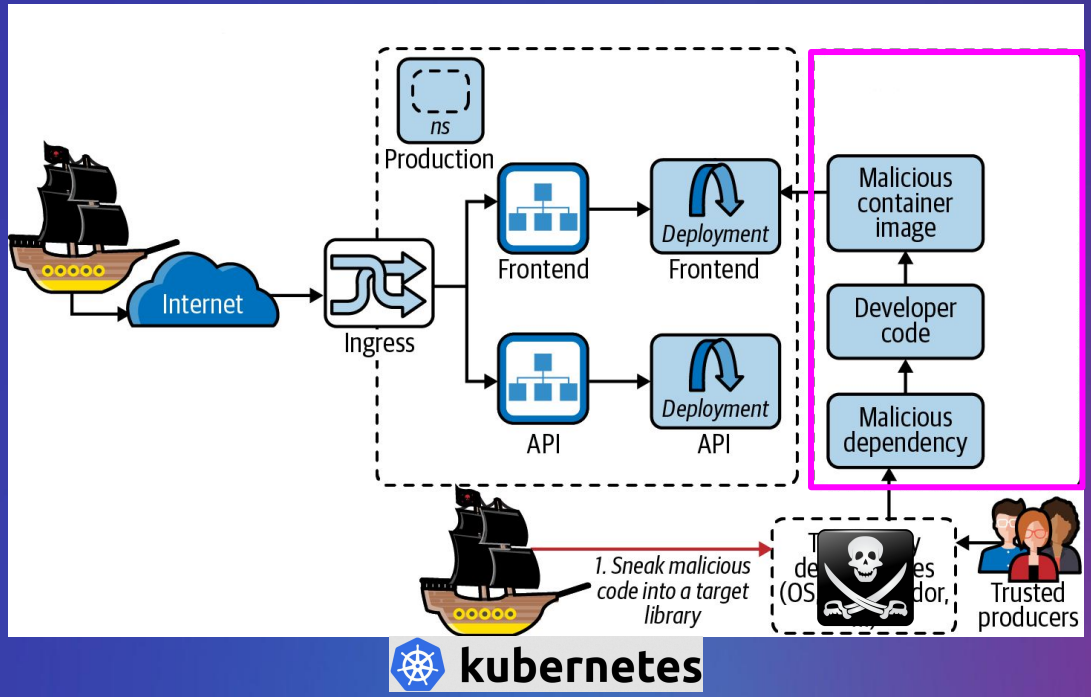
- Attacker forces victim to run malicious code (“implant”, “exploit”, etc)
- Victim inadvertently connects back to Attacker with permissions of user that ran code
- Attacker continues the attack from within the Victim’s host



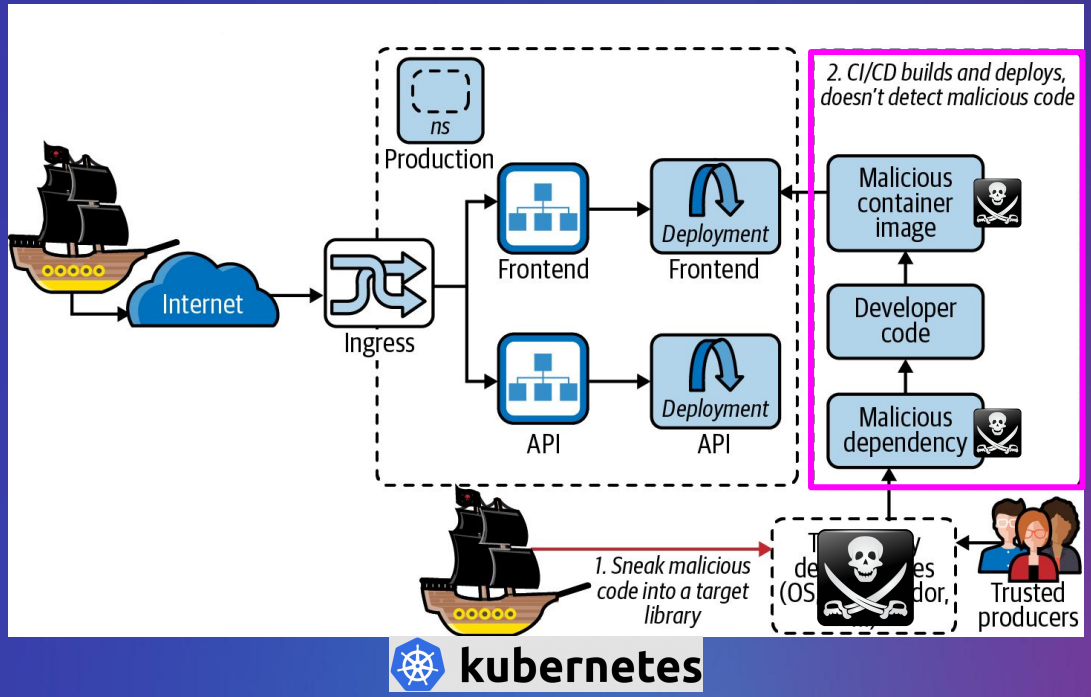
Attacking from Behind the Firewall: Reverse Shells



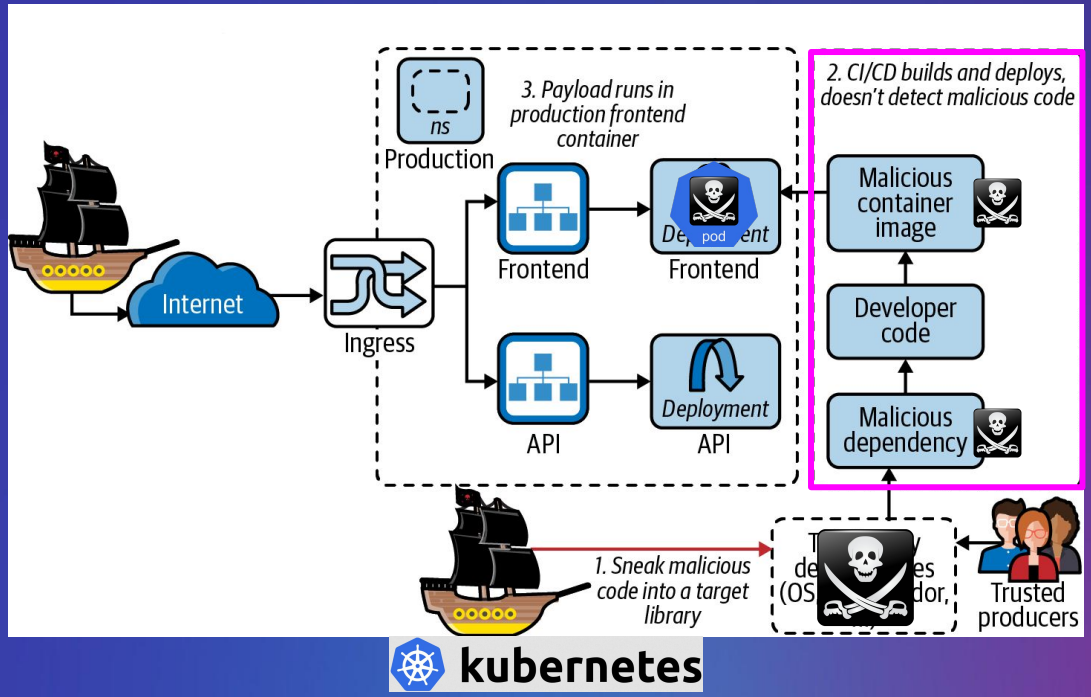
Attacking from Behind the Firewall: Reverse Shells



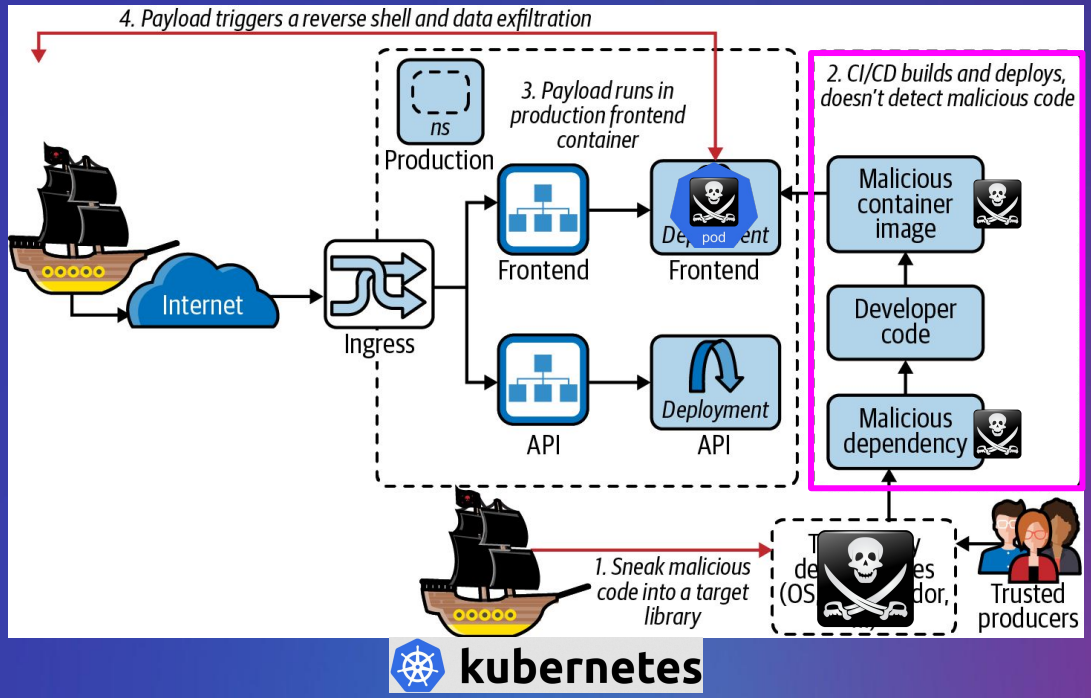
Attacking from Behind the Firewall: Reverse Shells



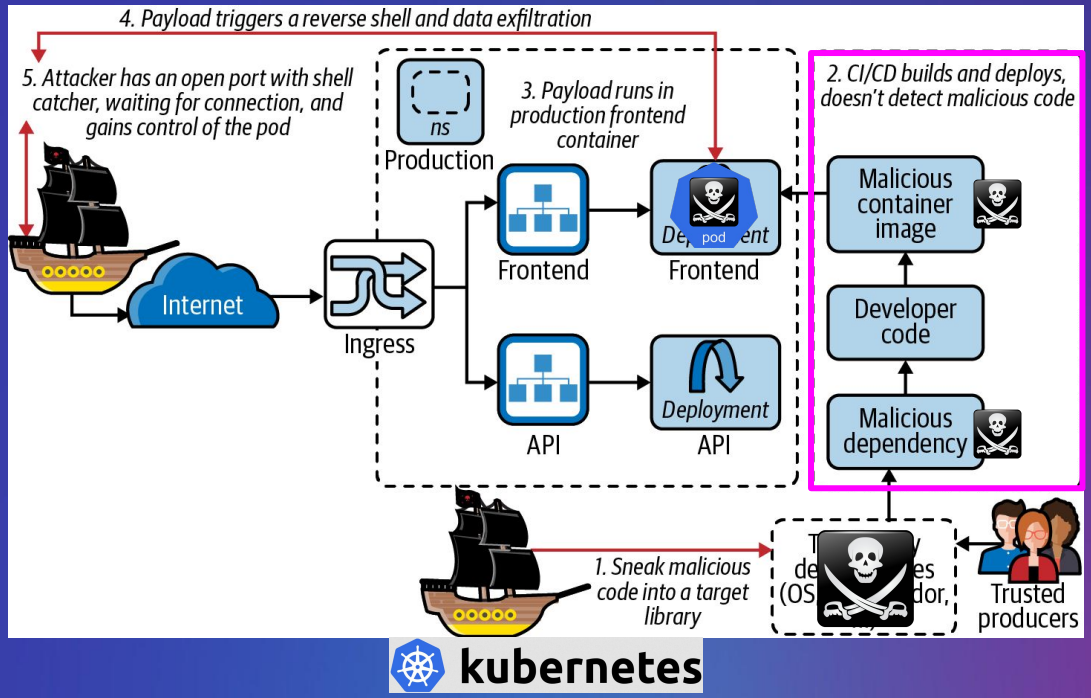
Attacking from Behind the Firewall: Reverse Shells



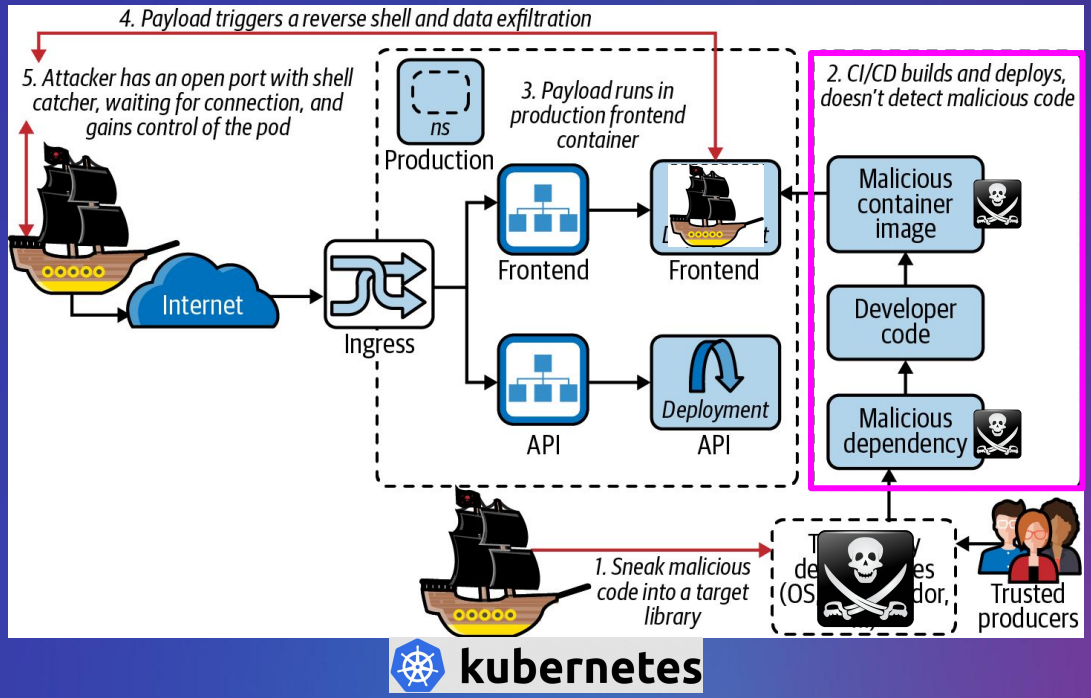
Attacking from Behind the Firewall: Reverse Shells



Attacking from Behind the Firewall: Reverse Shells



Attacking from Behind the Firewall: Reverse Shells



How to Attack a Supply Chain

- **Infect** a **trusted supplier**
  - Infect a dependency the source code pulls in
  - Infect a trusted package that runs as root
- **Infect** the **source code**
  - Developer's machine & git
  - Source repository
- **Infect** the **Build infrastructure**
- **Infect** the **Runtime environment** the workload is running in
  - Legitimate application code, compromised OS

Supply Chain Threats

Compromises can and will lead to:

- Remote code execution
- **Advanced persistent threats**
- Theft of data and IP
- Commercial espionage / Reputation damage
- Compromise of secrets
- Malware or implants
- **Cryptojacking and extortion**
- Denial of Service
- **Further supply chain attacks and compromised software**

Catalog of Supply Chain Compromises

- Source Code
- Dev Tooling
- Publishing Infrastructure
- Trust and Signing
- Negligence

https://github.com/cncf/tag-security/tree/main/supply-chain-security/compromises

@controlplaneio

Name	Year	Type of compromise	Link
PHP self-hosted git server	2021	Dev Tooling	1
Homebrew	2021	Dev Tooling	1, 2
Codecov	2021	Source Code	1
VSCode GitHub	2021	Dev Tooling	1
SUNBURST/SUNSPOT/Solarigate	2020	Publishing Infrastructure	1, 2, 3
The Great Suspender	2020	Malicious Maintainer	1,2
Abusing misconfigured SonarQube applications	2020	Dev Tooling	1, 2
Octopus Scanner	2020	Dev Tooling	1,2
NPM reverse shells and data mining	2020	Dev Tooling	1
Webmin backdoor	2019	Dev Tooling	1, 2
purescript-npm	2019	Source Code	1 and 2
electron-native-notify	2019	Source Code	1, 2
PyPI typosquatting	2019	Negligence	1
ROS build farm compromise	2019	Trust and Signing Publishing Infrastructure	1, 2
ShadowHammer	2019	Attack Chaining	1, 2
PEAR Breach	2019	Publishing Infrastructure	1, 2
Canonical's GitHub org compromised	2019	Dev Tooling Source Code Publishing infrastructure	1
The event-stream vulnerability	2018	Malicious Maintainer	1,2
Dofoil	2018	Publishing Infrastructure	1
Operation Red	2018	Publishing Infrastructure	1
RCE in go get -u	2018	Dev Tooling	1, 2
acroread compromised in AUR	2018	Malicious Maintainer	1, 2

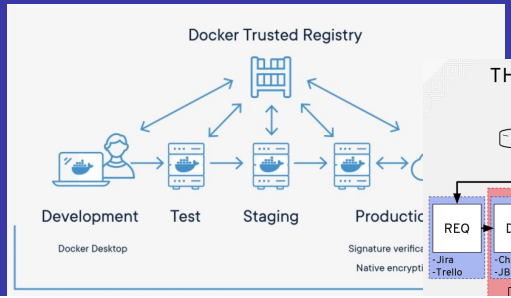
Agenda

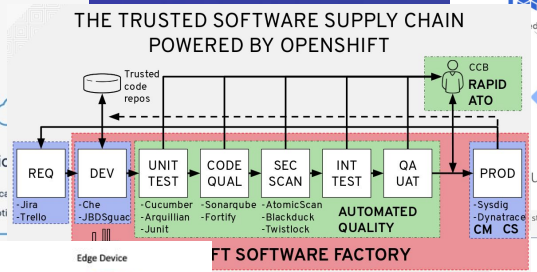
- ~~(Software) Supply Chain Problem~~
  ~~Simple Case study: Applications and Supply Chain attack~~
- Problem Space
- Threat Modelling Supply Chains
- Securing Supply Chains

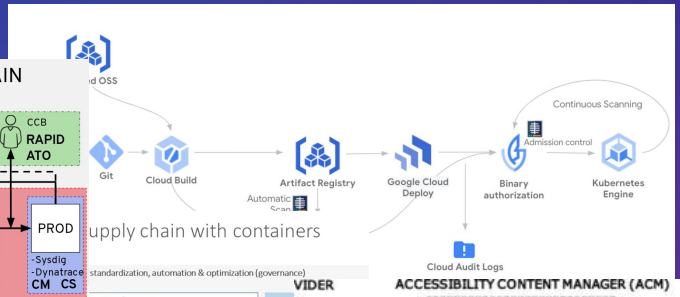
Securing Supply Chains is worse than most people think

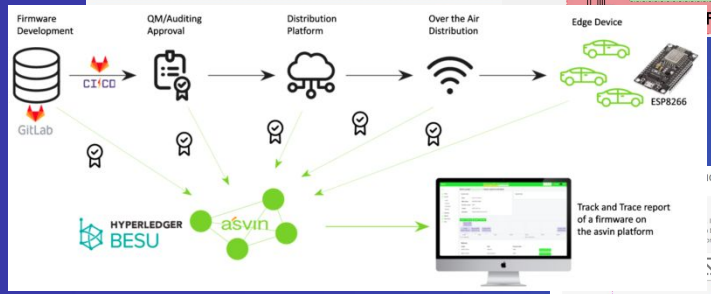
Why?

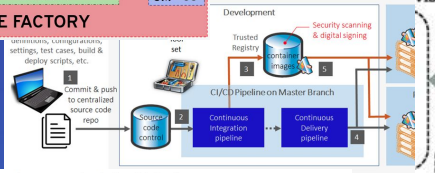
Problem Space

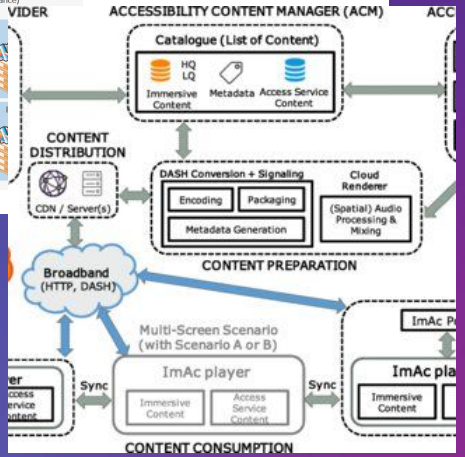


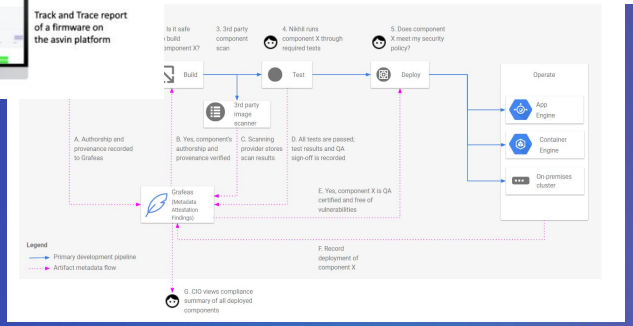












@controlplaneio

controlplane

What do we do about it

Options:

- We don’t use them (too late)
- We use them and don’t care (really?)
- We secure everything (you know better!)

What do we do about it

Options:

- We don’t use them (too late)
- We use them and don’t care (really?)
- We secure everything (you know better!)
- **We Threat Model them!**
- **We develop and adopt standards, best practices, reference architectures and keep improving them**

“Organizations rarely control their entire software supply chain and lack authority to compel every organization in their supply chain to take prompt mitigation steps”

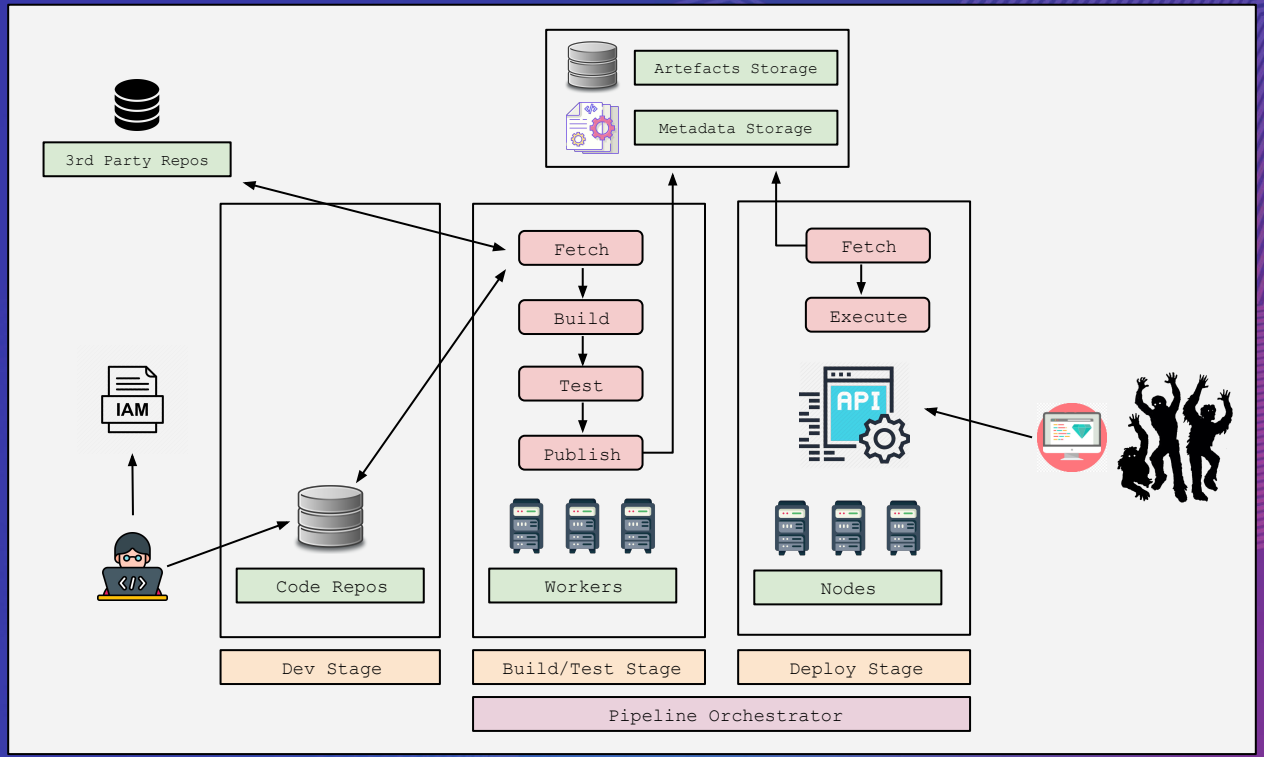
CISA - Defending Against Software Supply Chain Attacks

Agenda

- ~~(Software) Supply Chain Problem~~
  ~~Simple Case study: Applications and Supply Chain attack~~
- ~~Problem Space~~
- Threat Modelling Supply Chains
- Securing Supply Chains

Threat Model Supply Chains

Supply Chain Example



What is threat modelling?

A **systematic** approach that:

- Brings **everyone** at the same table
- More informal (no requirement for established risk framework)
- Happens early in the design and build processes
- Focuses on data / data-flows
- Aids in finding and addressing security risks
- Derives tactical, actionable data:
  - Attack trees
  - Security controls and countermeasures

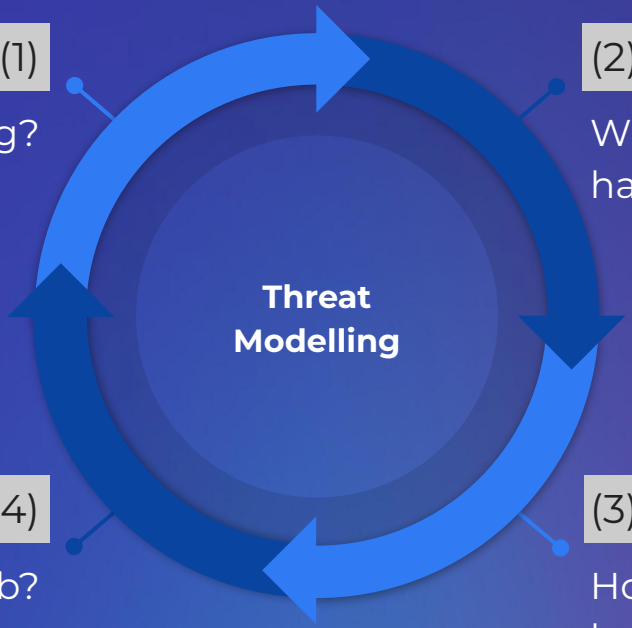
What is the process?

One (1)

What are we building?

(2) Two

What's the worst that could happen?



Four (4)

Did we do a good job?

(3) Three

How can we reduce the risk of bad things happening?

STEP 1: What are we building?

Scope

- Supply Chain components and trust boundaries
- Data flows

Business Impact

- Key risks to the business arising from compromise of C.I.A.

Use Cases

- Understand what the Supply Chain is used for

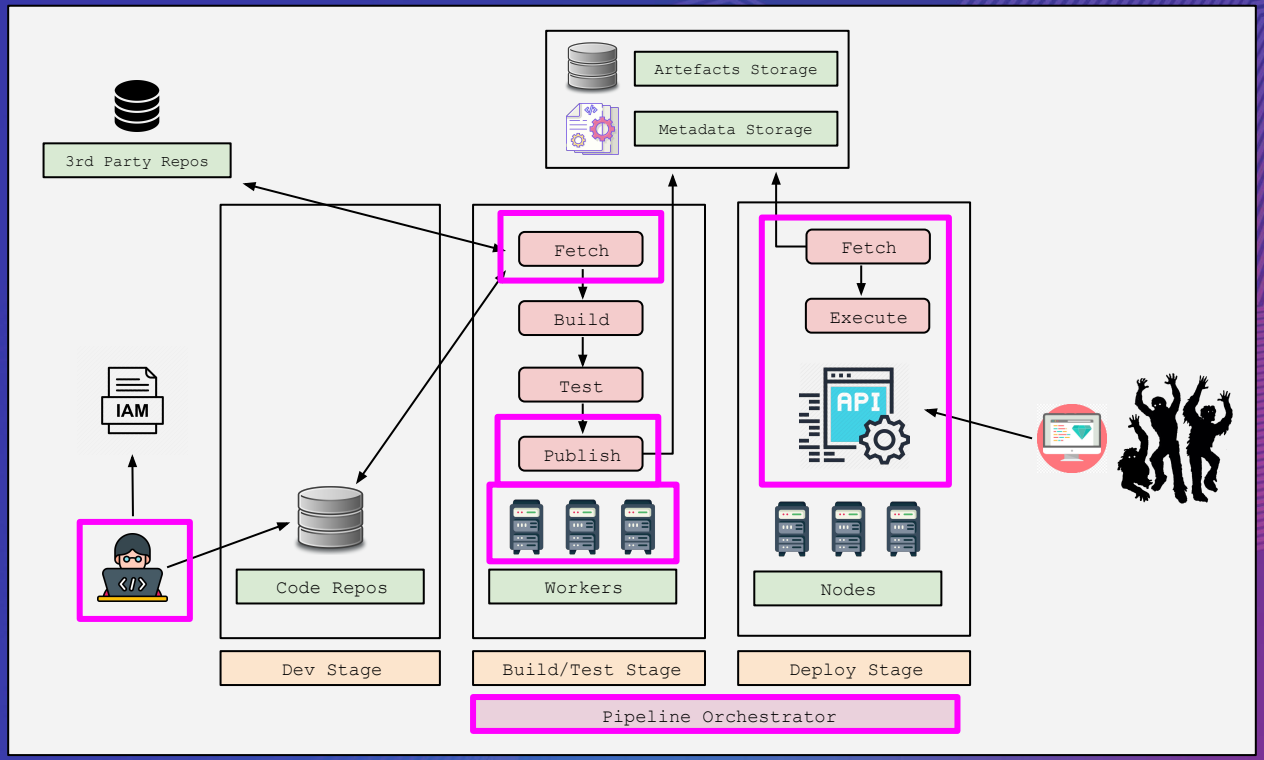
Operating Model

- How the Supply Chain is operated and consumed or maintained by teams.

STEP 1: Tips

- Scope
  - Define what's **in scope** (inc processes, ...) and what's **NOT in scope**
  - **Decouple**! Any common component may warrant **individual threat models**
- Documentation
  - Lack of deep technical quality documentation
  - Technical reference architectures, patterns and anti-patterns
- Communication to stakeholders
  - **Understand your audience**
  - No witch-hunting - people are part of Supply Chain too
  - **Collaboration is everything** - make them comfortable

Decouple, scope it down



STEP 2: What's the worst that could happen?

1 **Scenarios**

2 **Threats**

Top threat statements map back to the business risk.`

3 **Attack Paths**

Individual steps to exploit a potential threat.

4 **Attack Trees**

Generate visual representation of threat models.

Threat Actors

Depending on the use cases, threat actors categorise the potential adversaries that a system is configured to defend against.

Actor	Motivation	Capability	Sample attacks
Vandal: Script Kiddie, Trespasser	Curiosity, Personal Fame	Uses publicly available tools and applications (Nmap, Metasploit, CVE PoCs)	Small scale DOS / Launches prepackaged exploits / cryptomining
Motivated individual: Political activist, Thief, Terrorist	Personal Gain, Political or Ideological	May combine publicly available exploits in a targeted fashion. Modify open source supply chains	Phishing / DDOS / Exploit known vulnerabilities
Insider: employee, external contractor, temporary worker	Disgruntled, Profit	Detailed knowledge of the system, understands how to exploit/conceal	Exfiltrate data (to sell on) / Misconfiguration / "code bombs"
Organised crime: syndicates, state-affiliated groups	Ransom, Mass extraction of PII/credentials/PCI data, financial gain	Devotes considerable resources, writes exploits, can bribe/coerce, can launch targeted attacks	Social Engineering / Phishing / Ransomware / Coordinated attacks
Cloud Service Insider: employee, external contractor, temporary worker	Personal Gain, Curiosity	Depends on segregation of duties and technical controls within cloud provider	Access to or manipulation of datastores
Foreign Intelligence Services (FIS): nation states	Intelligence gathering, Disrupt Critical National Infrastructure	Disrupt or modify supply chains. Infiltrate organisations. Develop multiple zero-days. Highly targeted.	Stuxnet, SUNBURST

HOWTO - Identify potential threats

Examples:
- threats to data
- threats to service availability
- threats to system integrity

Participate!

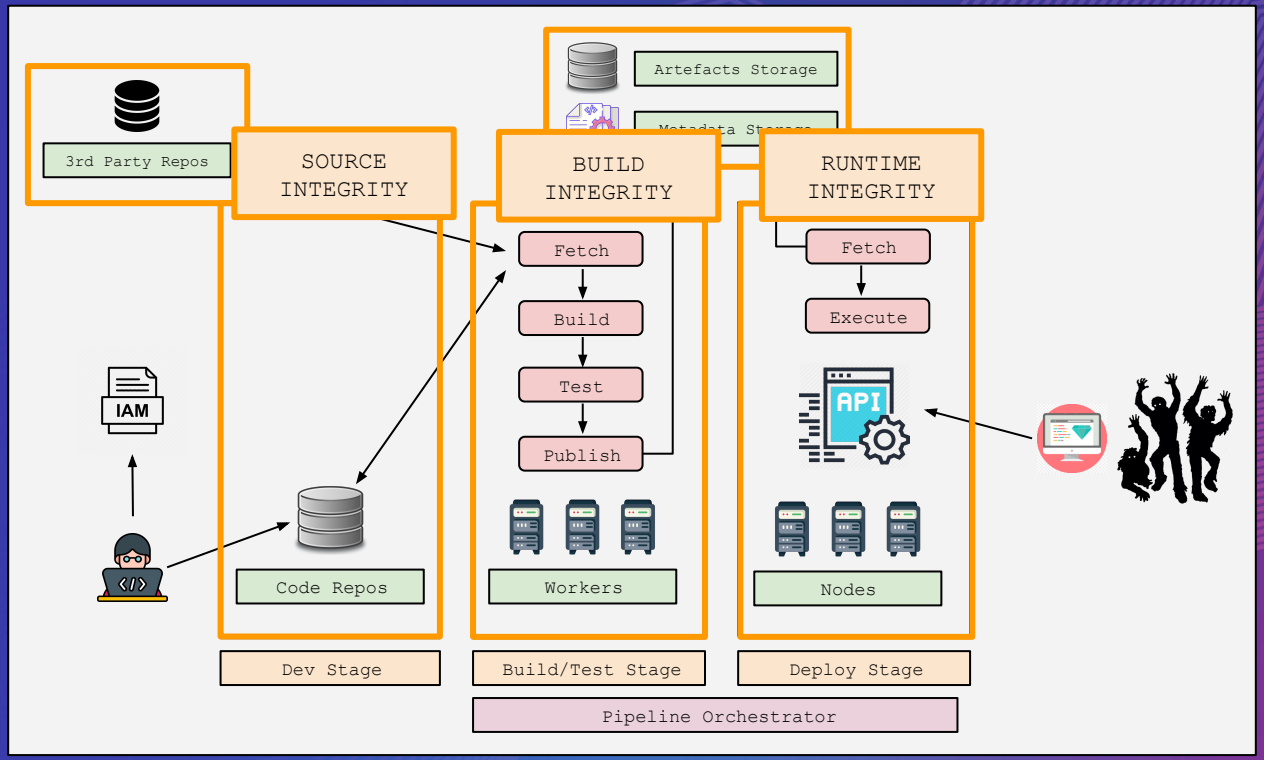
OSINT

*-ISAC

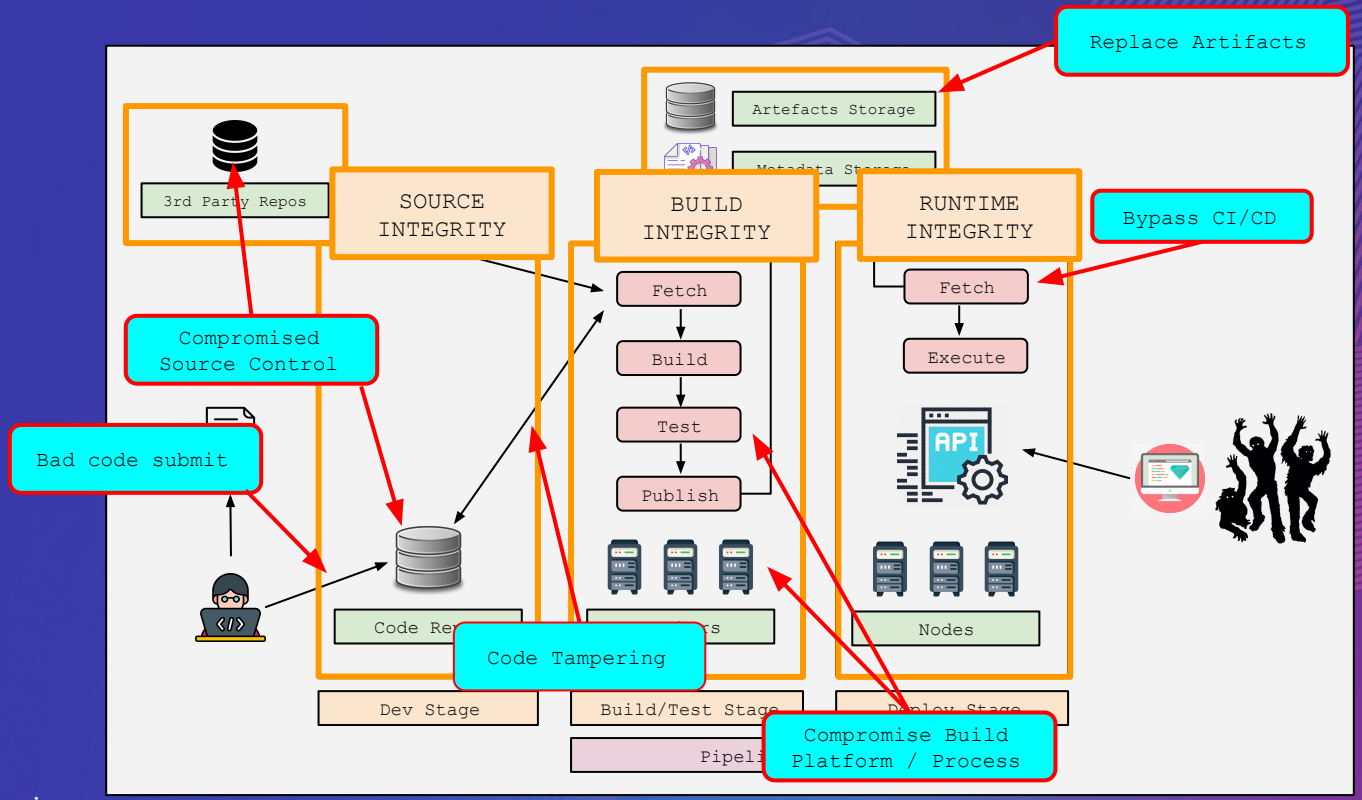
* SIGs

- Leverage team experience, intuitions (not limited to security), **prioritize**
- Exploit threat intelligence sources
  - MITRE ATT&CK
  - CNCF financial services attack vectors
- Profile threats based on well–known frameworks, e.g. STRIDE, PASTA, ...

Threats to Supply Chain Security Properties



Threats to Supply Chain Security Properties



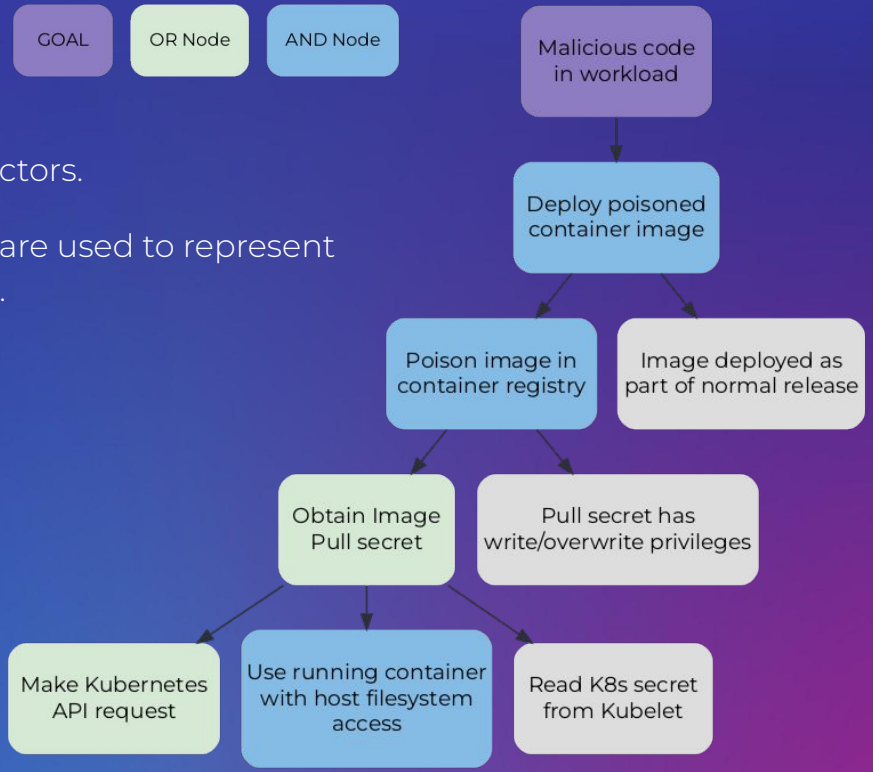
Attack Trees

Graphical representation of the potential infiltration vectors.

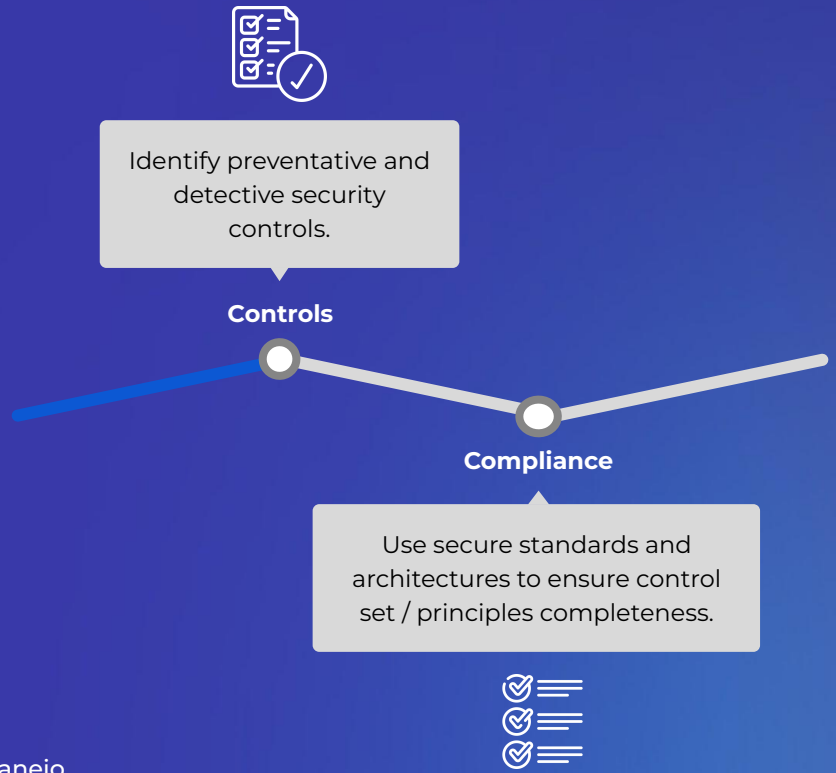
Different shapes or color schemes (as in this example) are used to represent logical gates, attacker goal, and individual attack steps.

Can annotate nodes to provide extra context

- likelihood and impact
- security controls to mitigate individual steps



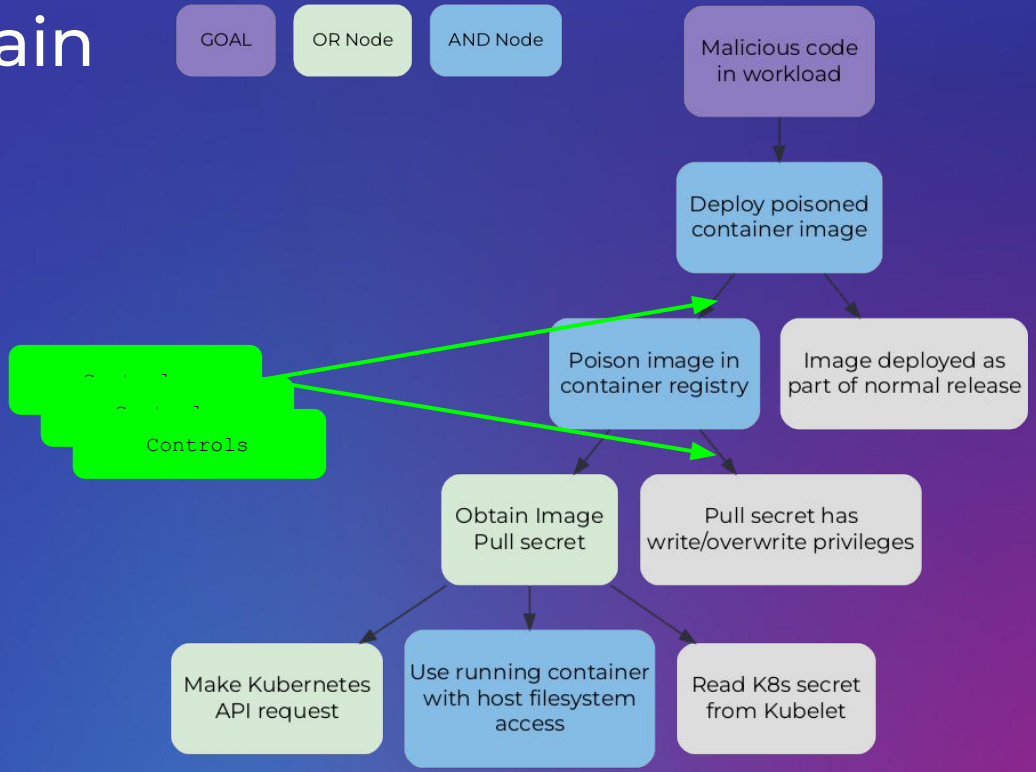
STEP 3: How can we reduce the risk of bad things happening?



It will rarely be possible to implement all controls, due to:

- Budget implications
- Prioritisation of work with respect to other business requirements
- Balancing operator, developer, and user requirements and “Total Security”

Break the Attack Chain



STEP 4: Did we do a good job?

Risk Analysis

Understand the inherent and residual risk of the threats.

Test Suites

Generate infrastructure test suites to demonstrate control effectiveness.

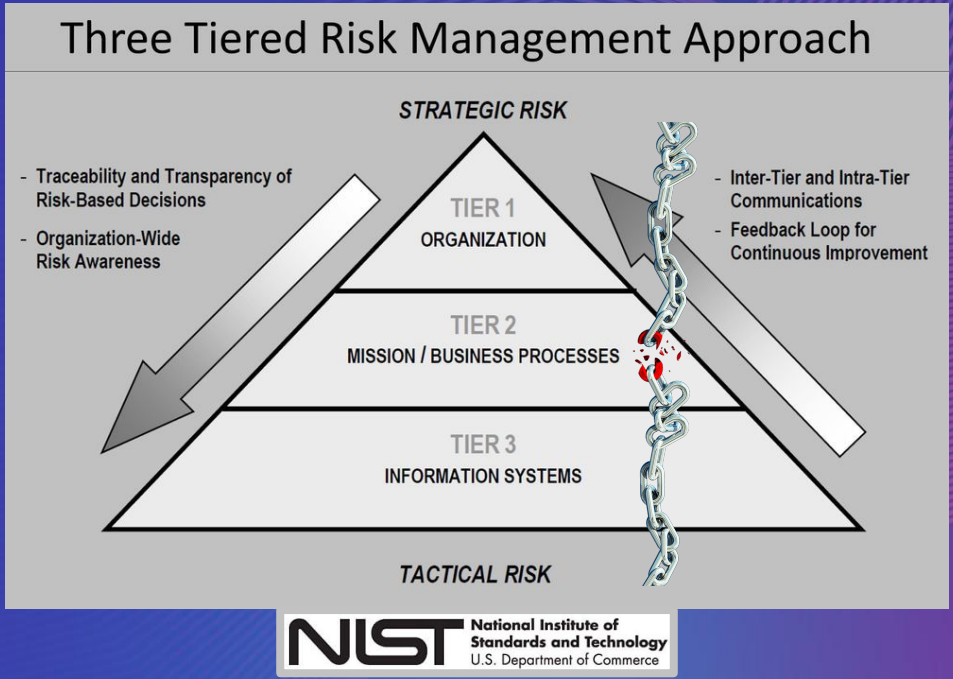
Pentest

Dynamically assess the security posture of the environment.

Revise Model

Identify new threats in response to intelligence.

Threat Model Everything



Agenda

- ~~(Software) Supply Chain Problem~~
  ~~Simple Case study: Applications and Supply Chain attack~~
- ~~Problem Space~~
- ~~Threat Modelling Supply Chains~~
- Securing Supply Chains

Securing Supply Chains

Frameworks and Controls

Glossary

Provenance

“*A claim that some entity (builder) produced one or more software artifacts (statement’s subject) by executing some recipe, using some other artifacts as input (materials)*”

Source commit / Build url / Test results / Security analysis / Deployments / Approvals

Attestation

“Independently *verify the statement (set of claims) not only the artifact. Provide the data to automate governance and compliance tasks.*”

Verification

“The Consumer can verify the artifact and its signed provenance”

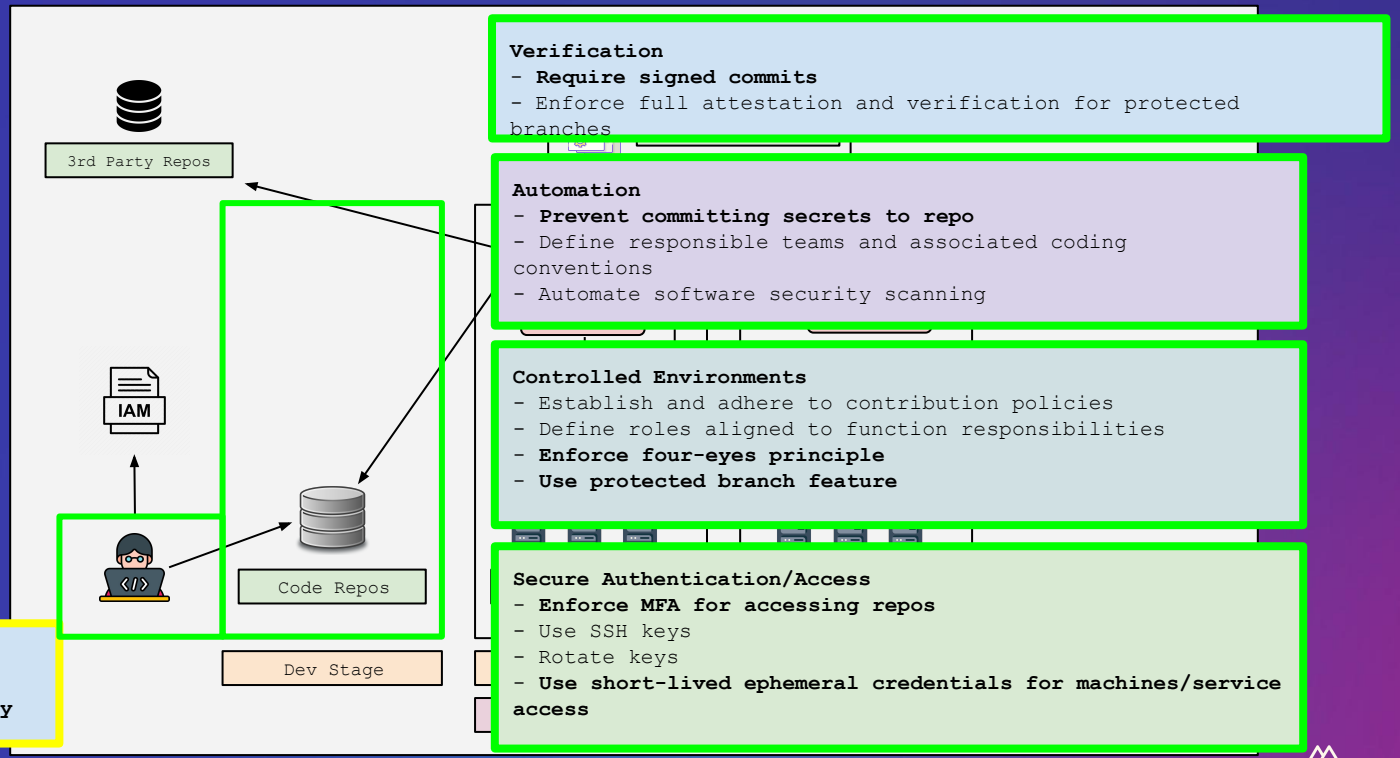
Cloud Native Supply Chain Security Whitepaper

- Security Principles
  - **Verification**
    - E.g. Each stage is **cryptographically verified**
    - **E.g. Metadata attestation**
  - **Automation**
    - **E.g. Deterministic** CI/CD **pipelines**, aiding attestation and verification
    - E.g. Infrastructure and security controls defined as **IaC**
  - **Controlled Environments (Authorization)**
    - E.g. **All entities clearly defined and limited in scope**
    - E.g. **Least privilege**: just enough to do the job
  - **Secure Authentication / Access**
    - E.g. All entities must mutually authenticate (securely)

Cloud Native Supply Chain Security Whitepaper

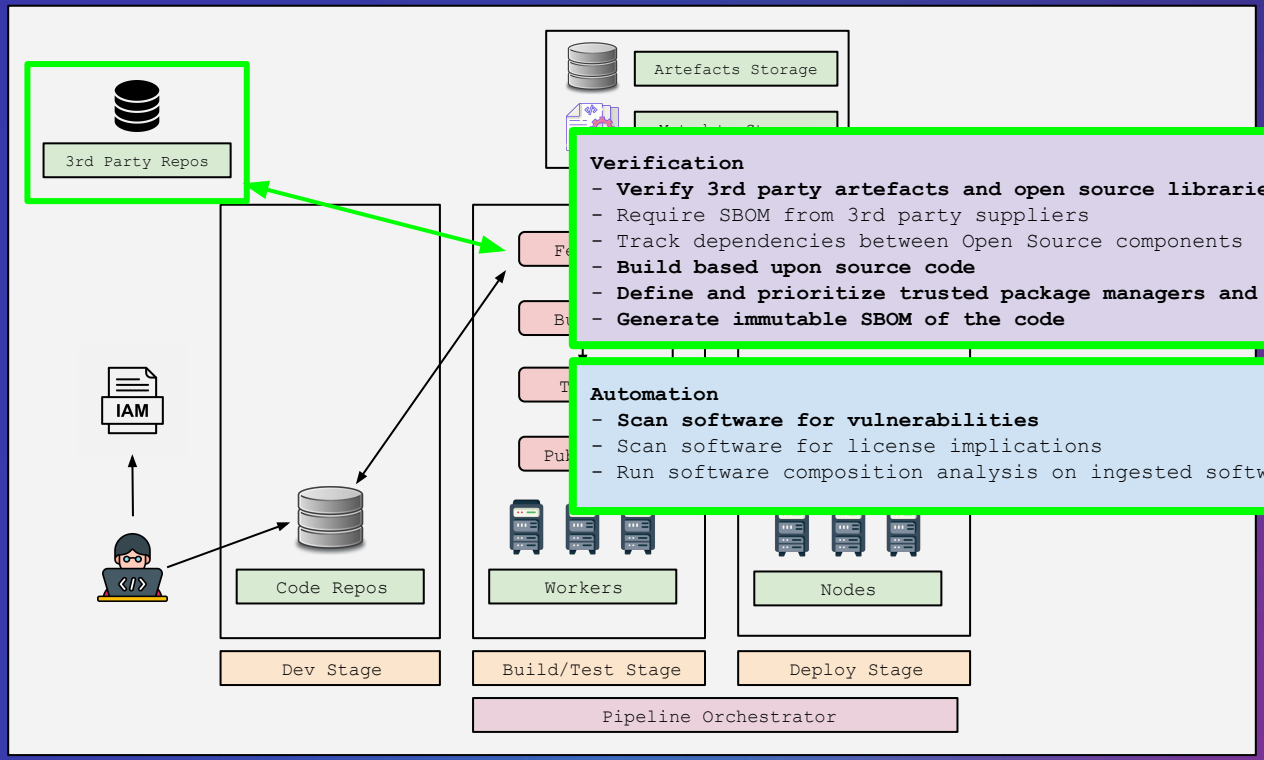
- Securing in 5 stages, built on Security Principles foundations
  - **Securing the Source Code**
  - **Securing Materials**
  - **Securing Build Pipelines**
  - **Securing Artefacts**
  - **Securing Deployments**

Securing the Source Code

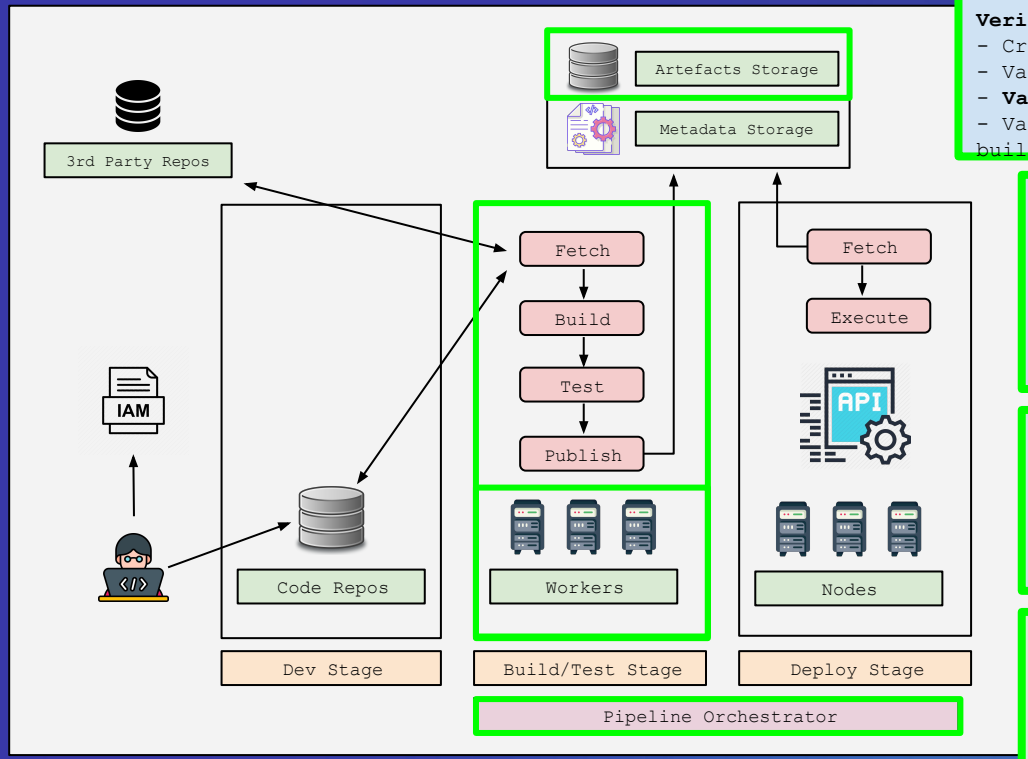


Other:
- Secure EUDs!
- Store keys properly

Securing the Materials



Securing the Build Pipelines



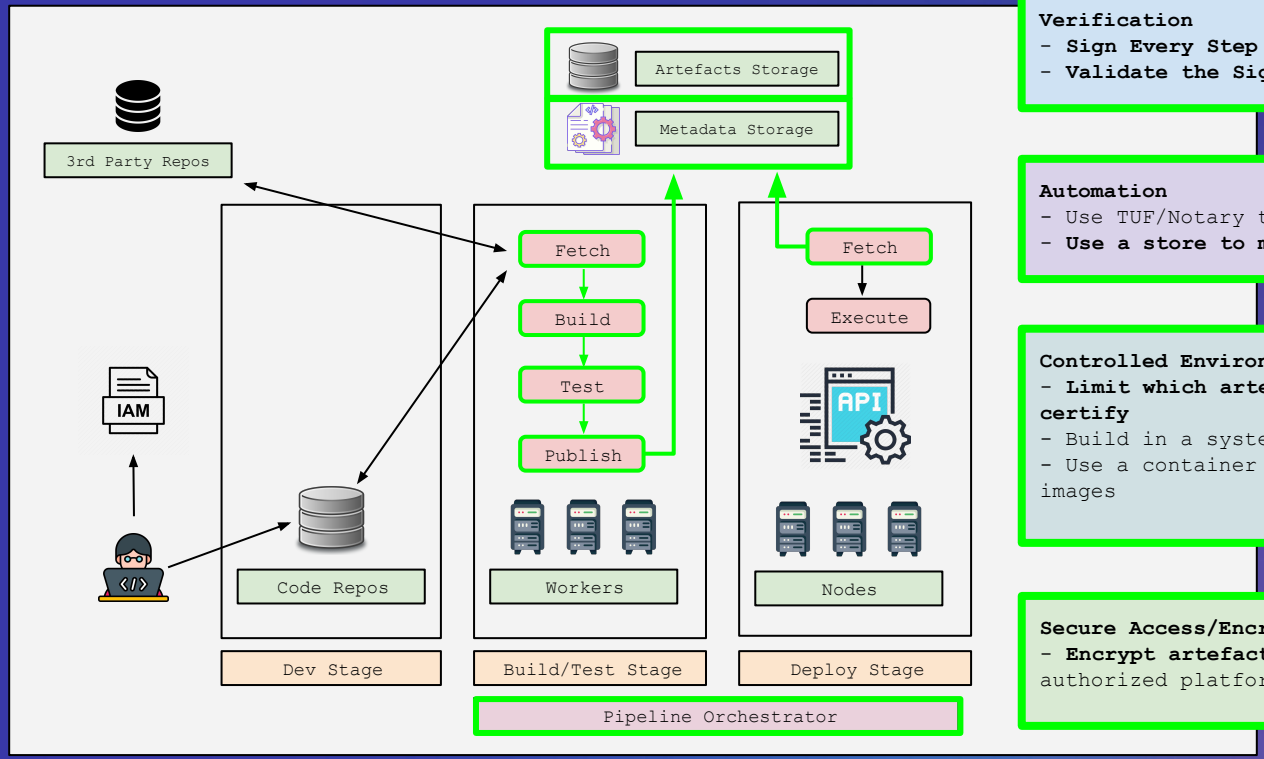
Verification
- Cryptographically guarantee policy adherence
- Validate environments and dependencies before usage
- **Validate runtime security of build workers**
- Validate Build artefacts through verifiably reproducible builds

Automation
- Build and related CI/CD steps should all be automated through a **pipeline defined as code**
- Standardize pipelines across projects
- Provision a secured orchestration platform to host software factory
- **Build Workers Should be Single Use**

Controlled Environments
- Ensure Software Factory has minimal network connectivity
- **Segregate the duties of each Build Worker**
- Pass in Build Worker Environment and Commands
- **Write Output to a Separate Secured Storage Repo**

Secure Authentication/Access
- **Only allow pipeline modifications through "Pipeline-a-C"**
- Define user roles
- Follow established practices for establishing a root of trust from an offline source
- **Use short-lived Workload Certificates**

Securing the Artefacts



Verification
- **Sign Every Step in the Build Process**
- **Validate the Signatures Generated at Each Step**

Automation
- Use TUF/Notary to manage signing of artefacts
- **Use a store to manage metadata**

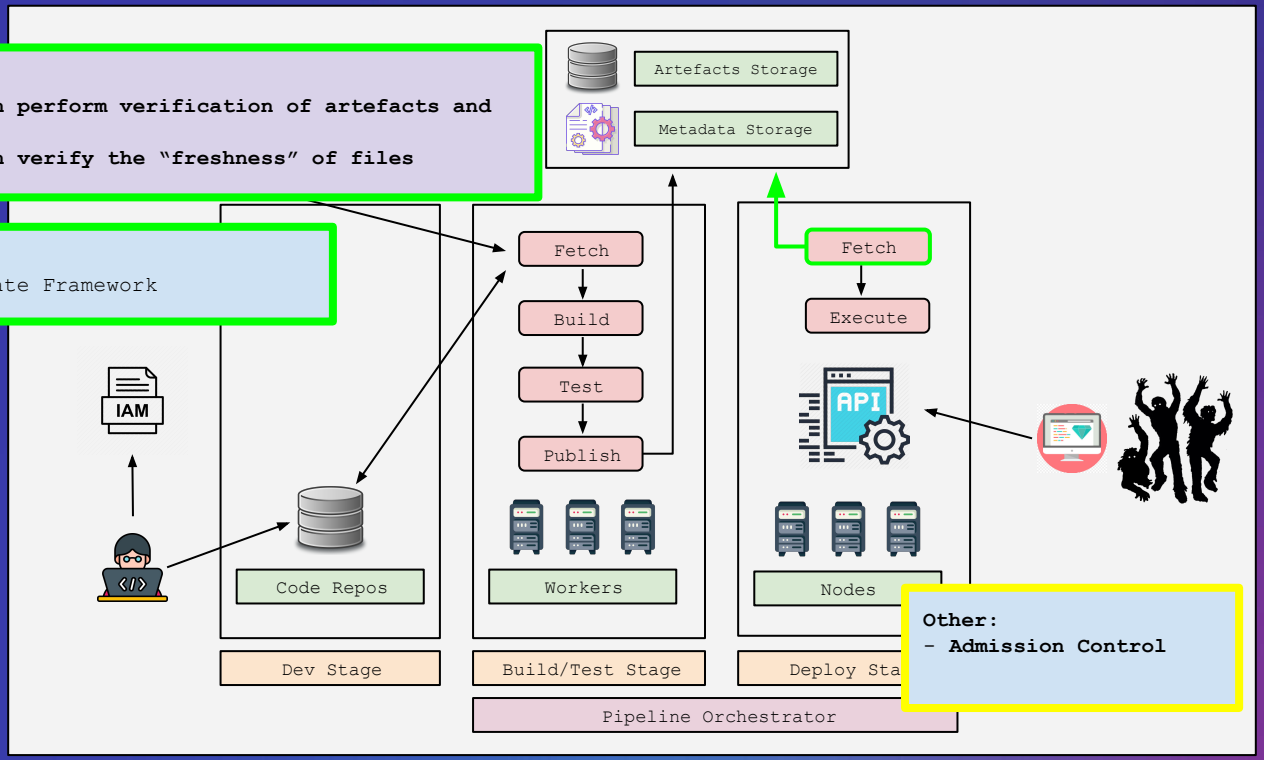
Controlled Environments
- **Limit which artefacts any given party is authorized to certify**
- Build in a system for **rotating** and **revoking** private keys
- Use a container registry that supports OCI image-spec images

Secure Access/Encryption
- **Encrypt artefacts before distribution** & ensure only authorized platforms have decryption capabilities

Securing Deployments

Verification
- **Ensure clients can perform verification of artefacts and associated metadata**
- **Ensure clients can verify the "freshness" of files**

Automation
- Use (TUF) The Update Framework

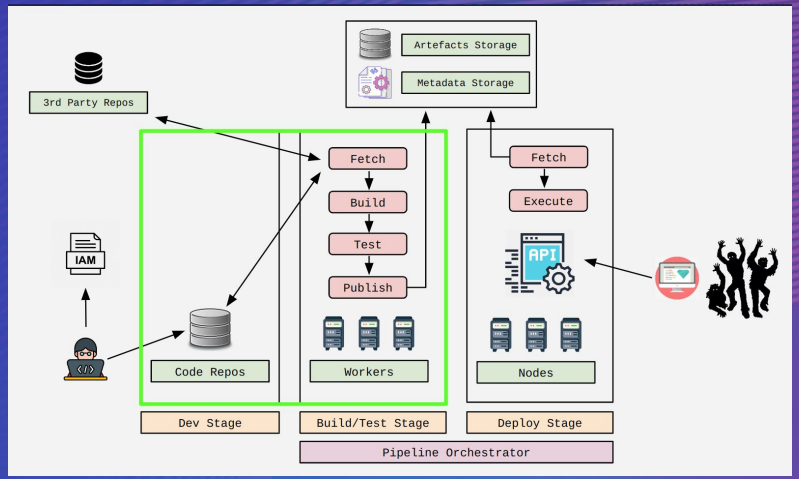


Other:
- **Admission Control**

controlplane

SLSA - Supply chain Levels for Software Artifacts

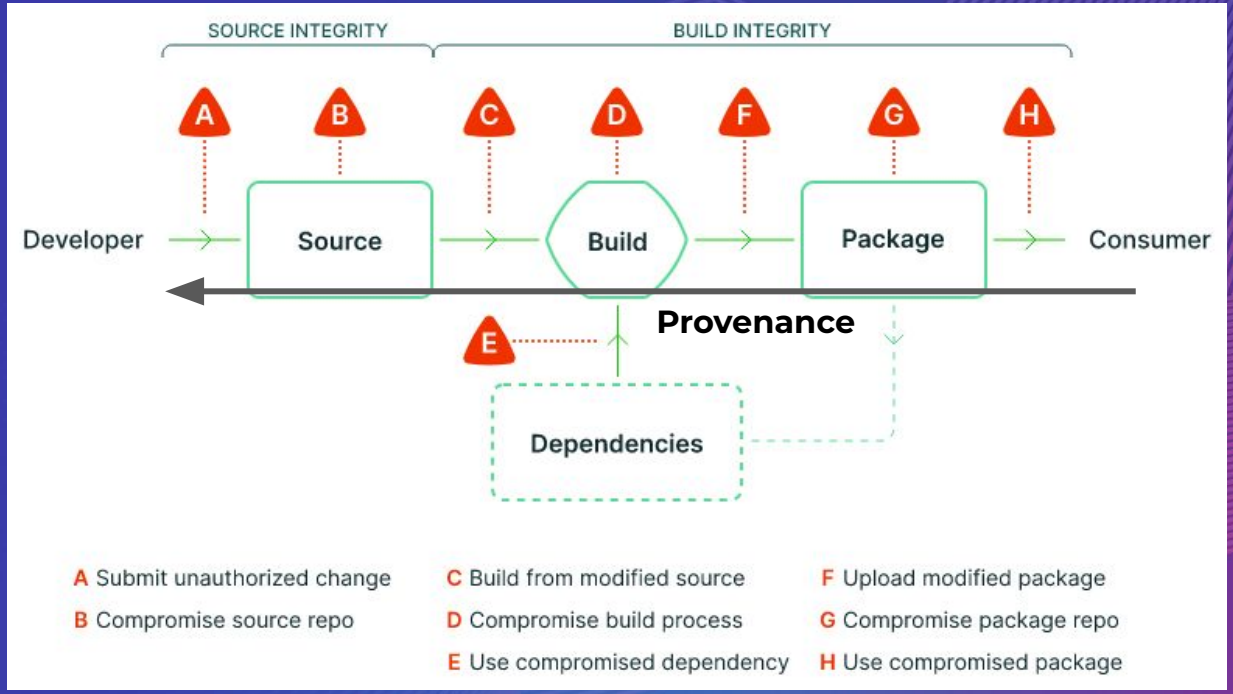
- A security **framework,** with **incrementally** adoptable security **guidelines**, established by industry consensus.
- Check-list of **standards** and **controls** to prevent **tampering**, improve **integrity**, and **secure packages and infrastructure** of your supply chains.
- A common **language** to talk about how **secure** software, **supply chains** and their component parts really are.



@controlplaneio

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

SLSA Threat Model



SLSA Levels for hardening **Source**, **Build**, **Provenance**

Basic security steps

Level 1 means the supply chain is documented, there's infrastructure to generate provenance, and systems are prepared for higher SLSA levels.

After the build

Level 2 shows more trustworthiness in the build, builders are source-aware, and signatures are used to prevent provenance being tampered with.

Back to source

Level 3 shows that a system's builds are fully trustworthy, build definitions come from the source and a system has more hardened CI.

Across the chain

Level 4 means the build environment is fully accounted for, dependencies are tracked in provenance and insider threats are ruled out.

SLSA - Protect Source Code

Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Source - Version controlled		✓	✓	✓
Source - Verified history			✓	✓
Source - Retained indefinitely			18 mo.	✓
Source - Two-person reviewed				✓

SLSA - Protect the Build

Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Build - Scripted build	✓	✓	✓	✓
Build - Build service		✓	✓	✓
Build - Build as code			✓	✓
Build - Ephemeral environment			✓	✓
Build - Isolated			✓	✓
Build - Parameterless				✓
Build - Hermetic				✓
Build - Reproducible				○

SLSA - Protect the Provenance

Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Provenance - **Available**	✓	✓	✓	✓
Provenance - **Authenticated**		✓	✓	✓
Provenance - **Service generated**		✓	✓	✓
Provenance - **Non-falsifiable**			✓	✓
Provenance - **Dependencies complete**				✓

SLSA Minus One

- Security of the build infrastructure, necessary for the SLSA levels to provide value
- Prevents tampering with the build system's controls
- Cloud and on-prem differ in control requirements

A roundup of “Other Good Stuff”

Base Image

- Hardened base images
- Distroless containers
- Minimal OS dependencies
- Linux microcosm, standard security practices
- Make sure developers can debug their images

Code

- Commit signing (GPG, Gitsign)
- Static analysis (ASTs, reachability analysis)
- Dependency analysis (vulnerability scanning, VEX)
- SBOM generation (CycloneDX, SPDX)
- OpenSSF CI ScoreCard

Build

- Build tracing
- Pipeline metadata, workload identity, and build stage signing (in-toto, TUF, SPIFFE/SPIRE, Tekton Chains)
- Software factory (FRSCA, Tekton, Jenkins X)
- Evidence lakes and ledgers (Guac, Rekor)
- Artefact signing (cosign, notation)

Application Image

- Vulnerability scanning (VEX, Trivy et al)
- Config scanning (compliance, best practice)
- Admission control (signing and config tests)
- Runtime hardening (security context, LSMs)
- Least privilege deployment (cluster config)
- Misconfiguration protection (drift analysis)

Why track pipeline metadata?

- Pipeline metadata is rich and varied
  - Initiating user(s) and/or events
  - Installed dependencies and their versions
  - Veracity test data, e.g., unit/integration/acceptance/&c tests
  - Security test data

- Data can be used for recording (audit) and reporting/enforcing (policy)

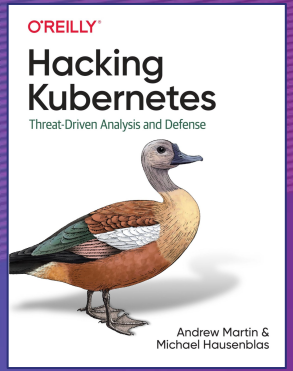
Agenda

- ~~(Software) Supply Chain Problem~~
  ~~Simple Case study: Applications and Supply Chain attack~~
- ~~Problem Space~~
- ~~Threat Modelling Supply Chains~~
- ~~Securing Supply Chains~~

Key Takeaways

- The Problem Space is **LARGE**
- Threat Model **EVERYTHING** (did we mention we're experts?)
- Principles, frameworks, guidelines and controls are **available**
- **Retrofit** and/or slowly **mature** your supply chain security
- You are NOT alone - join & contribute

Thank you!
Q&A



@controlplaneio

controlplane