KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

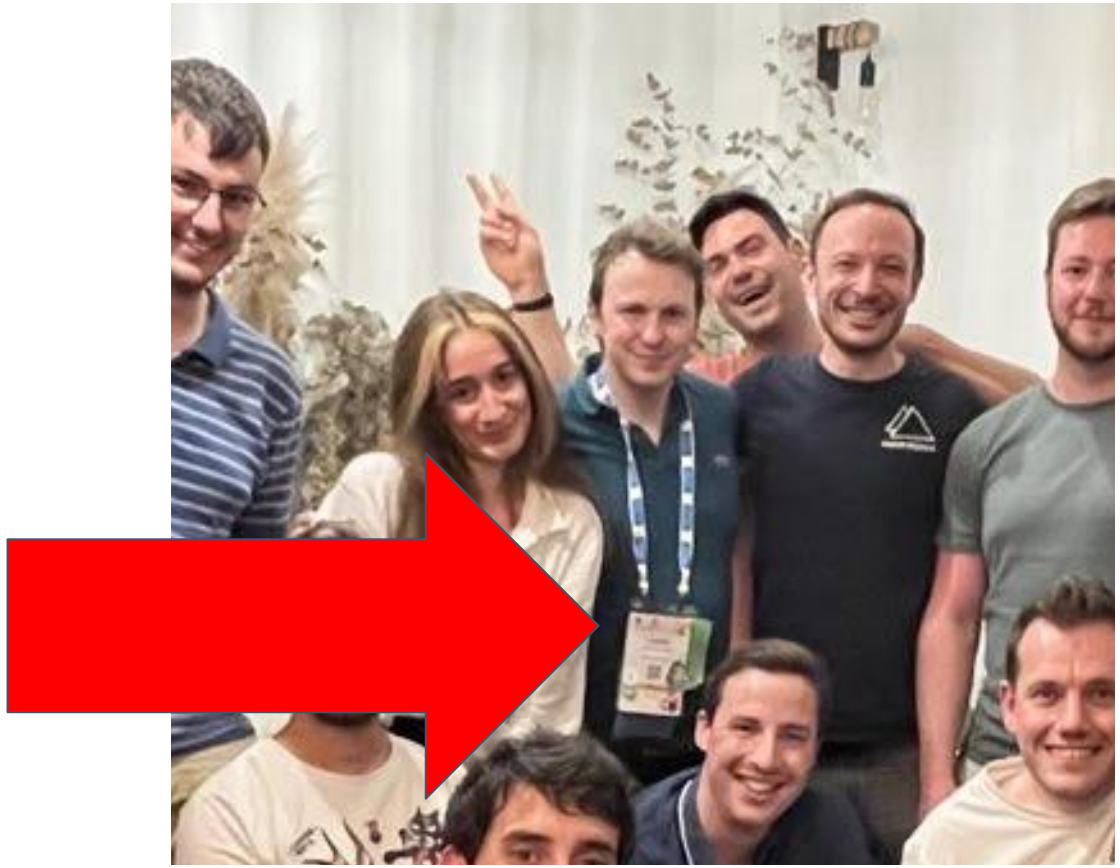
KubeCon CloudNativeCon Europe 2022

Threat Modelling Kubernetes: A Lightspeed Introduction

Lewis Denham-Parry, ControlPlane


controlplane


controlplane



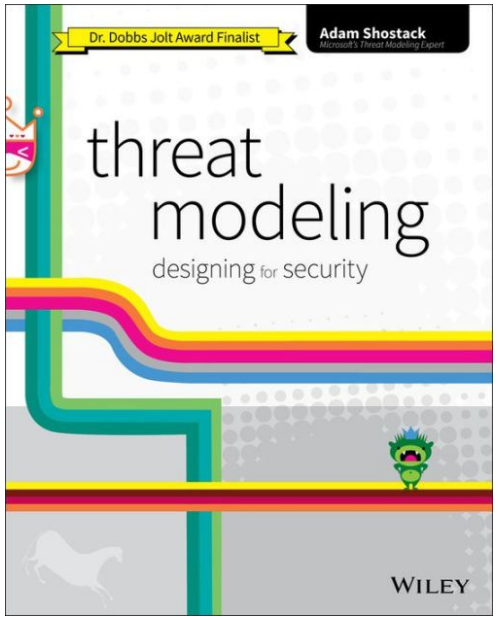

controlplane

Mental Health

Always learning

What is Threat Modelling?

- Threat modelling is:
  - Identifying and enumerating **threats and vulnerabilities**
  - Devising **mitigations**
  - Prioritising **residual risks**
  - **Escalating the most important risks**



- Why Threat Model?
  - Identify **security flaws early**
  - **Save money and time** consuming redesigns
  - **Focus your security requirements**
  - Identify **complex risks and data flows** for critical assets

Everyone can (and should!) Threat Model - **not just security teams**

Diversity

Iain Smart
@smarticu5

“You are stupid if you assume I’ll remember you asking what you think is a stupid question” might be one of the best talk takeaways I’ve heard. @denhamparry dropping life lessons AND an excellent introduction to CTFs in Kubernetes.

How to Threat Model

- Four steps:
  - What are we **building?**
  - What can **go wrong** once it's built?
  - What are we **going to do about** the things that can go wrong?
  - Are we doing a **good job of analysis?**

- Run workshops:
  - Structure your workshops around the four questions
  - Get as many **different views** as possible - development, operations, QA, product, business stakeholders, security
  - Workshops can be run to
    - **document system architecture**
    - **generate threats**
    - **devise controls**

What Are We Building?

Process - Documenting your system

- Understand your business's **data and value**
- Understand your **adversary**
- Decide on the **Threat Model's scope**
- Agree on the desired **threat mitigation level**
- Identify applicable **infosec policies, compliance standards**
- Agree on the **architecture**
  - **Architecture diagrams**
  - **Sequence diagrams**
  - **Data flow diagrams**

Understand your data

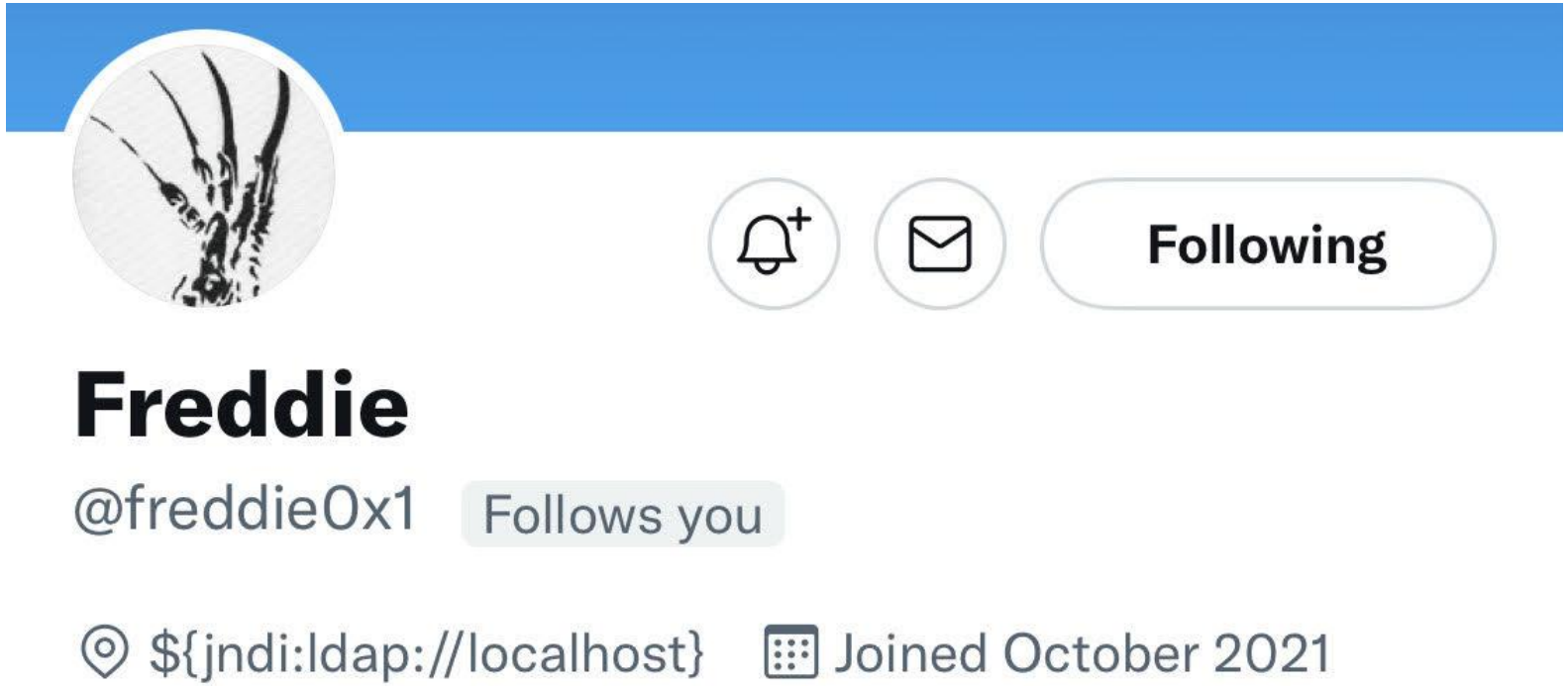
- It is important to understand:
  - The **business value of the data** you're trying to protect
  - Who your **attackers** may be
  - Your potential attackers' **capability** and **motivation** to attack your data
- Impact should be considered for **Confidentiality**, **Integrity** and/or **Availability**
- **Impact levels vary** between organisations
  - An example three tier impact level system for C/I/A:
    - Level 1: minimal reputational damage or financial loss of <£X (e.g. £10,000)
    - Level 2: some negative press or financial loss of <£Y (e.g. £100,000)
    - Level 3: compromise affects future of BCTL or financial loss of >£Z (e.g. £1,000,000)

Understand your data - case studies

- Impact variance examples:
  - a **financial institution** would be likely to grade impact levels based on **financial loss**
  - in a **military context**, impact could be related to **loss of life** and **operational failures**
- Impact may be different across the **fundamental properties**, e.g. confidentiality vs integrity
  - When providing bank details to **receive payment**, the **integrity of the information** is essential
  - When **making a card payment** to someone else, **confidentiality** is key

Adversary Matrix

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 / ”codebombs”
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


Following
Freddie
@freddie0x1
Follows you
${jndi:ldap://localhost}
Joined October 2021

Adversary Matrix

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 / ”codebombs”
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

Game 00: The Apprentice

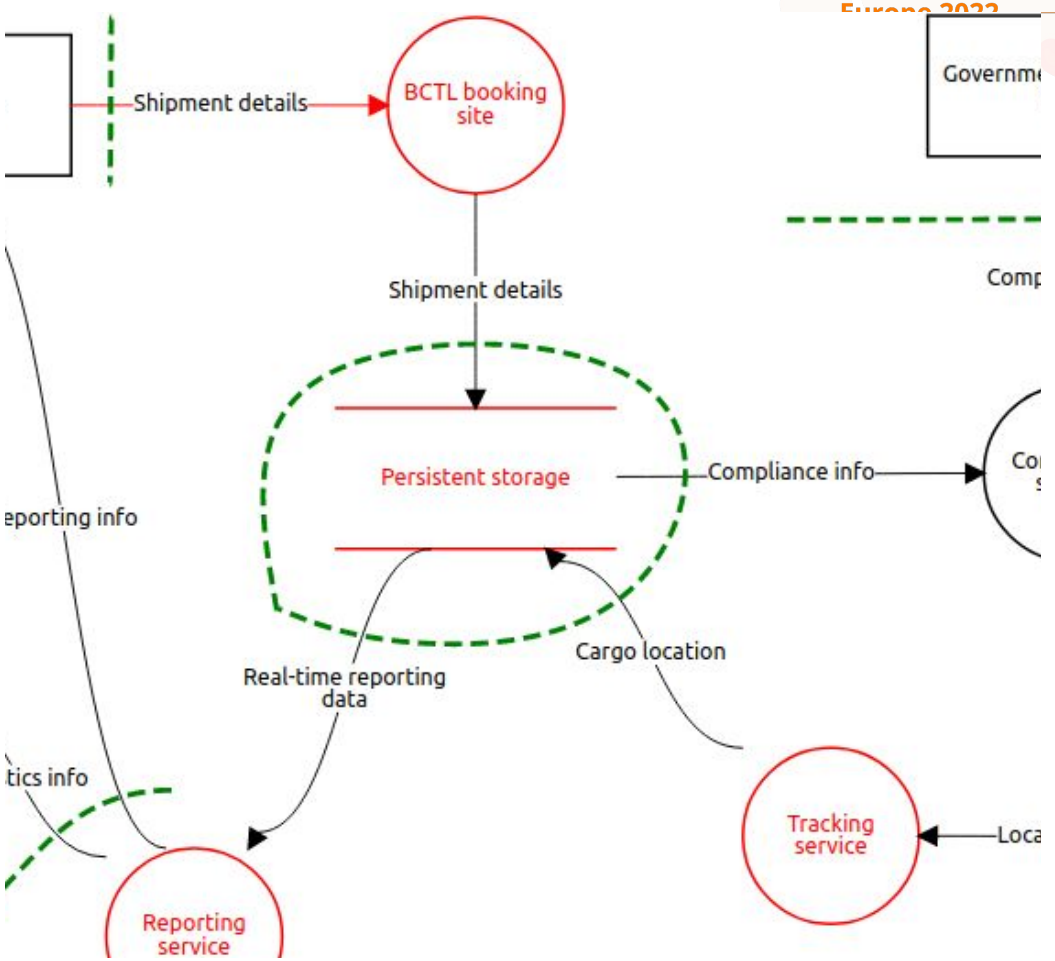
Adversary Matrix

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 / ”codebombs”
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

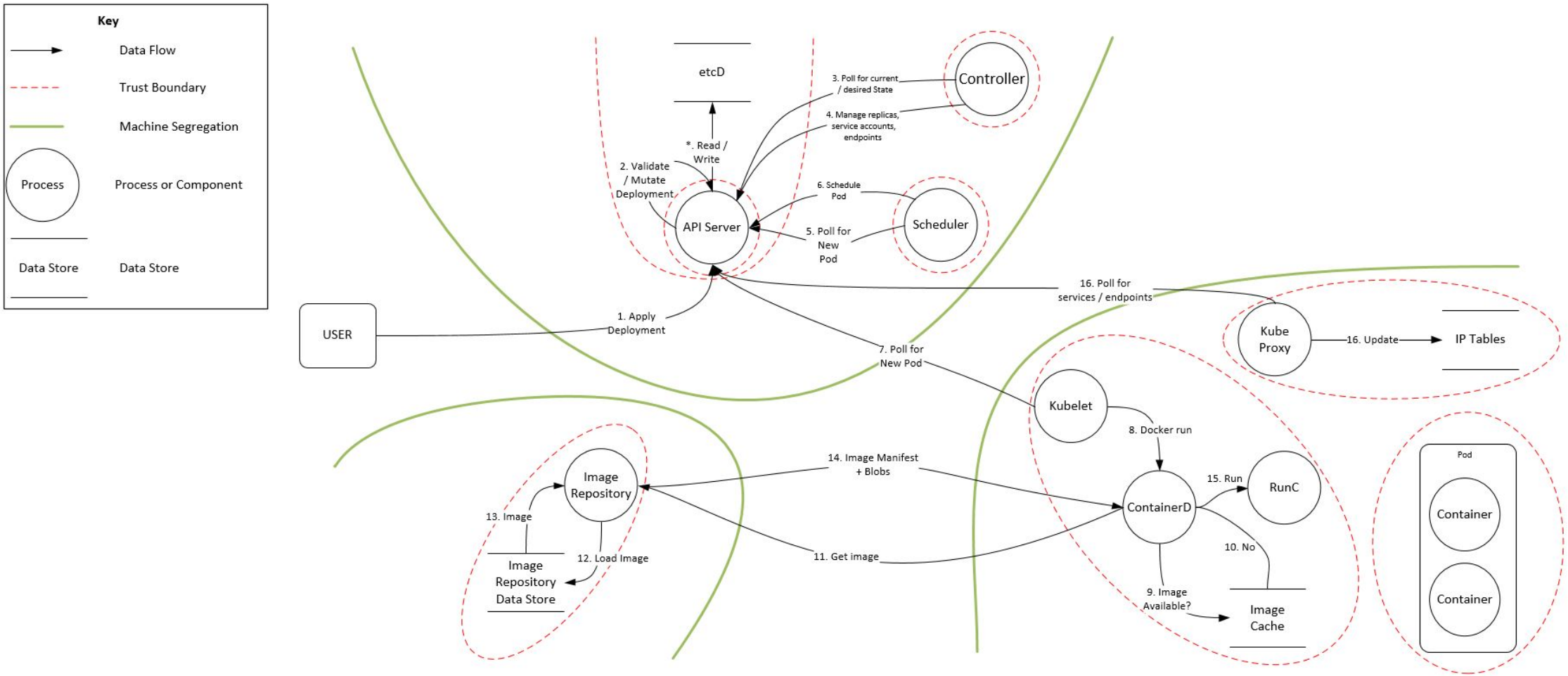
Game 01: Whose ~~line~~ sat next to you anyway

Scope the Threat Model

- Multiple techniques for **documenting a system,** e.g. Data Flow Diagrams (DFDs) and information matrices
- DFDs should:
  - Describe the **complete set of data flows**, where **process logic** occurs & **data stores**
  - Describe **trust boundaries**, all **user roles** and **network interfaces**
  - Be **self contained** and most importantly, **accurate**.
- DFDs can be drawn at various levels
  - Level 0: high-level system view
  - Subsequent levels (L1/L2 etc.) drill down into more detail on system components
- Tools for diagrams/documentation:
  - [OWASP Threat Dragon] etc.



Kubernetes Data Flow Diagram



[CNCF financial-services-user-group K8s DFD]

controlplane

Clusters in the wild

- Rory McCune (@raesene) has blogged about finding publicly accessible Kubernetes cluster on the Internet
- Using the censys search facility, valuable information for attackers can be found
- Many clusters will make the **`/version` endpoint available** on the API server **without authentication**
- Armed with the public IP address and the version of Kubernetes being run, this provides information to plan an attack

What Can Go Wrong?

Gathering Techniques and Threat Sources

- You and your team's **experience**, **gut instincts**, and **intuition**!
- Threat intelligence sources
  - MITRE ATT&CK framework
  - Open Web App Security Project (OWASP)
  - Common Weakness Enumeration (CWE)
  - Common Attack Pattern Enumeration and Classification (CAPEC)
  - CNCF financial services attack trees
  - Microsoft Kubernetes attack matrix
- Modelling techniques
  - STRIDE
  - Attack Trees

Microsoft Threat Matrix for Kubernetes

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Images from a private registry	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account		Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed Dashboard	SSH server running inside container				Access managed identity credential	Instance Metadata API	Writable volume mounts on the host		
Exposed sensitive interfaces	Sidecar injection				Malicious admission controller		Access Kubernetes dashboard		
							Access tiller endpoint		
							CoreDNS poisoning		
							ARP poisoning and IP spoofing		

= New technique

= Deprecated technique

controlplane

STRIDE

- For each
  - Process
  - Data flow
  - Store
  - Actor
- Identify what might go wrong

STRIDE - Spoofing

Pretending to be something or someone you’re not

STRIDE - Tampering

Modifying something you're not supposed to modify. This can be on disk, in memory, and/or in transit

ST**R**IDE - Repudiation

Claiming you didn’t do something, whether or not you actually did

STRIDE - Information Disclosure

Exposing information to people who aren't authorised to see it

STRI**D**E - Denial Of Service

Taking actions to prevent the system from providing service to legitimate users

STRID**E** - Elevation of Privilege

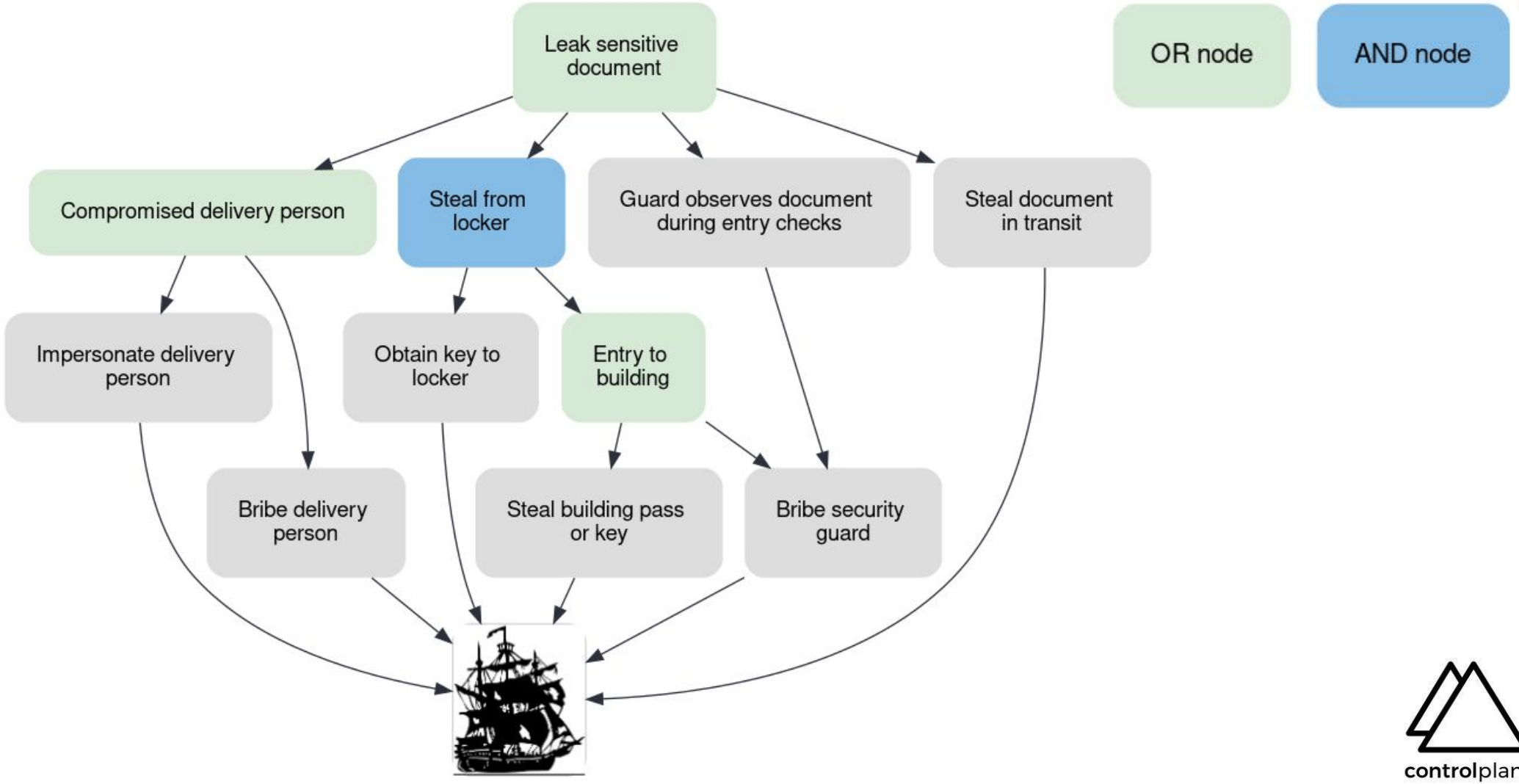
Being able to perform operations you aren’t supposed to be able to perform

Attack Trees

*"**represent attacks against a system in a tree structure**, with the goal as the root node and different ways of achieving that goal as leaf nodes"*

- Can **annotate** in a number of ways to provide **extra context** e.g.:
  - **Attack costs** in order to understand attack **likelihood and required resource**.
  - **Mitigating security controls** to the security of the system and threat events not covered.
- Attack trees should **enhance** the threat modelling process and **not overwhelm**
  - E.g. focus on areas of most interest

Example Attack Tree: carry a briefcase to a locker



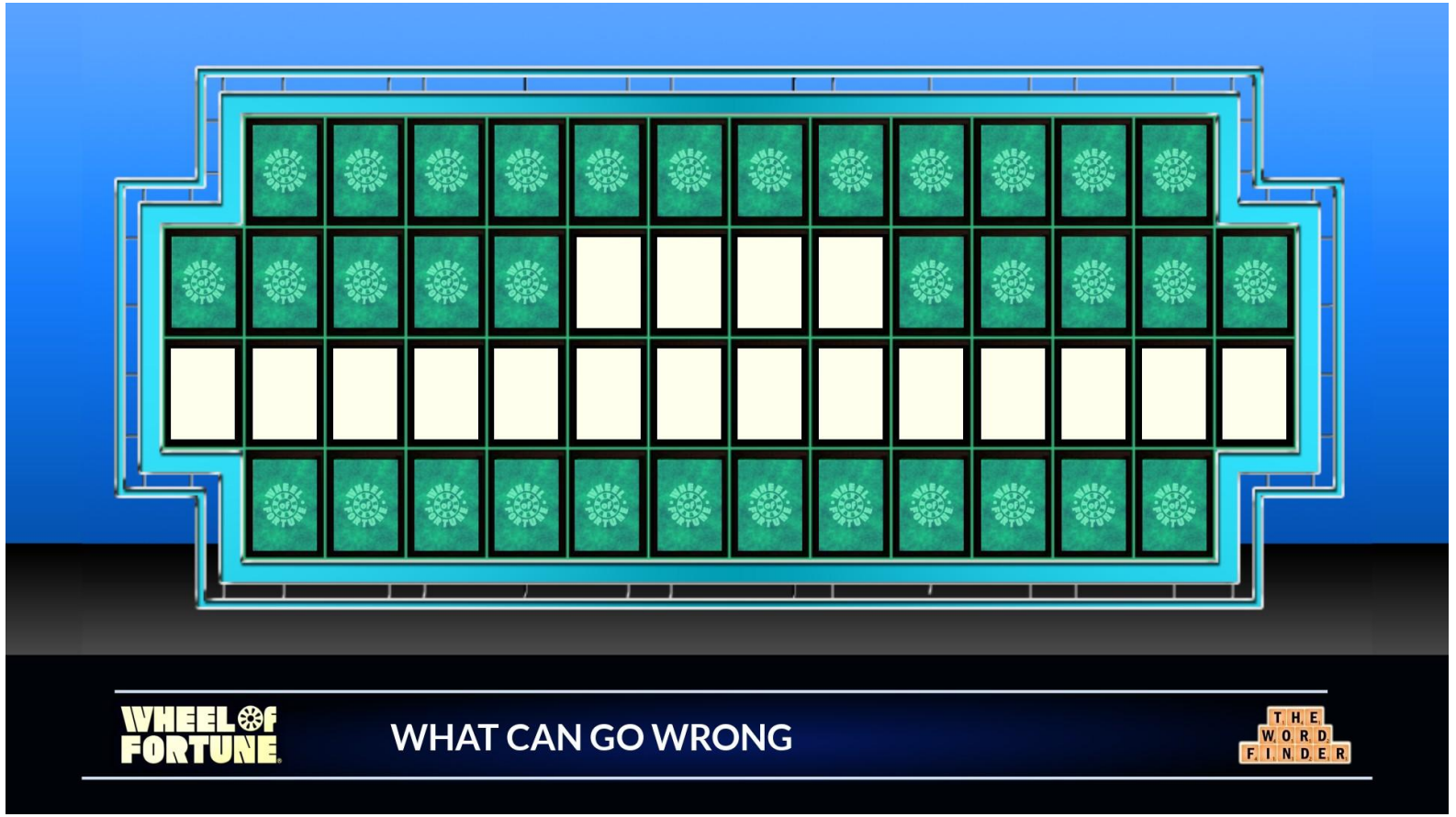
Attack Trees as code

- **Graphviz** can be used to **represent attack trees as code**:
  - Kelly Shortridge leads the way https://swagitda.com/blog/posts/security-decision-trees-with-graphviz/
- **Deciduous** is a web application that makes this easier:
  - https://swagitda.com/deciduous/

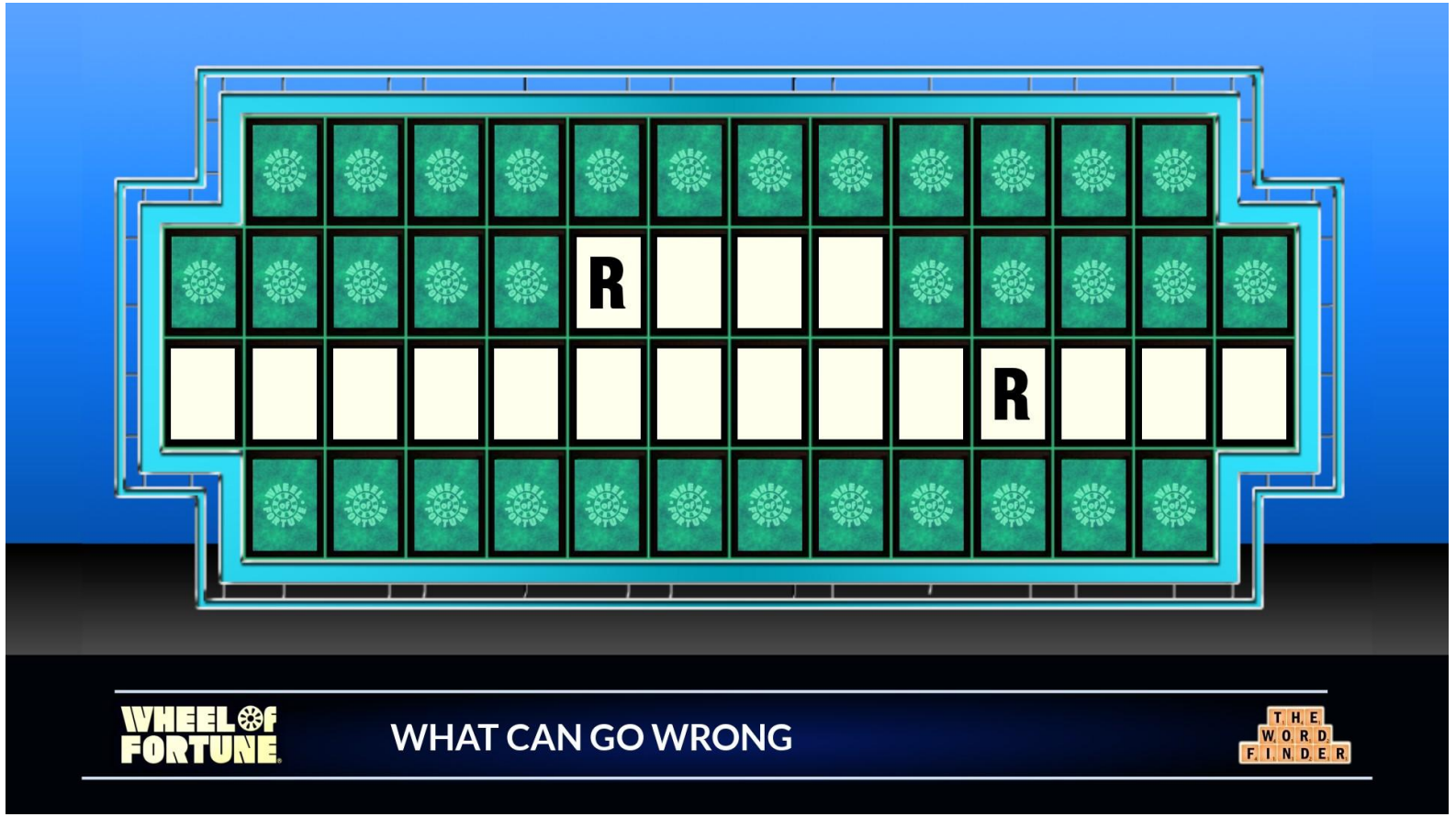
- We have provided a Dockerfile generate trees in PNG format:
  - https://github.com/controlplaneio/threat-modelling-labs

Applying this to Cloud & Kubernetes

Game 02: Wheel of (mis)Fortune



https://www.thewordfinder.com/wof-puzzle-generator



https://www.thewordfinder.com/wof-puzzle-generator

https://www.thewordfinder.com/wof-puzzle-generator

https://www.thewordfinder.com/wof-puzzle-generator

https://www.thewordfinder.com/wof-puzzle-generator

https://www.thewordfinder.com/wof-puzzle-generator

What Can Go Wrong: Workloads

- **RCE**
- **Container breakout**
- **Filesystem**
- **Host Mounts**
- **Hostile Containers**
- **Runtime**
- **Pod configuration**
- **Pod security context and service accounts**

What Can Go Wrong: Networking

- **Intra-pod**
- **Inter-pod**
- **No workload identity by default**
- **No network policy by default**

What Can Go Wrong: Storage

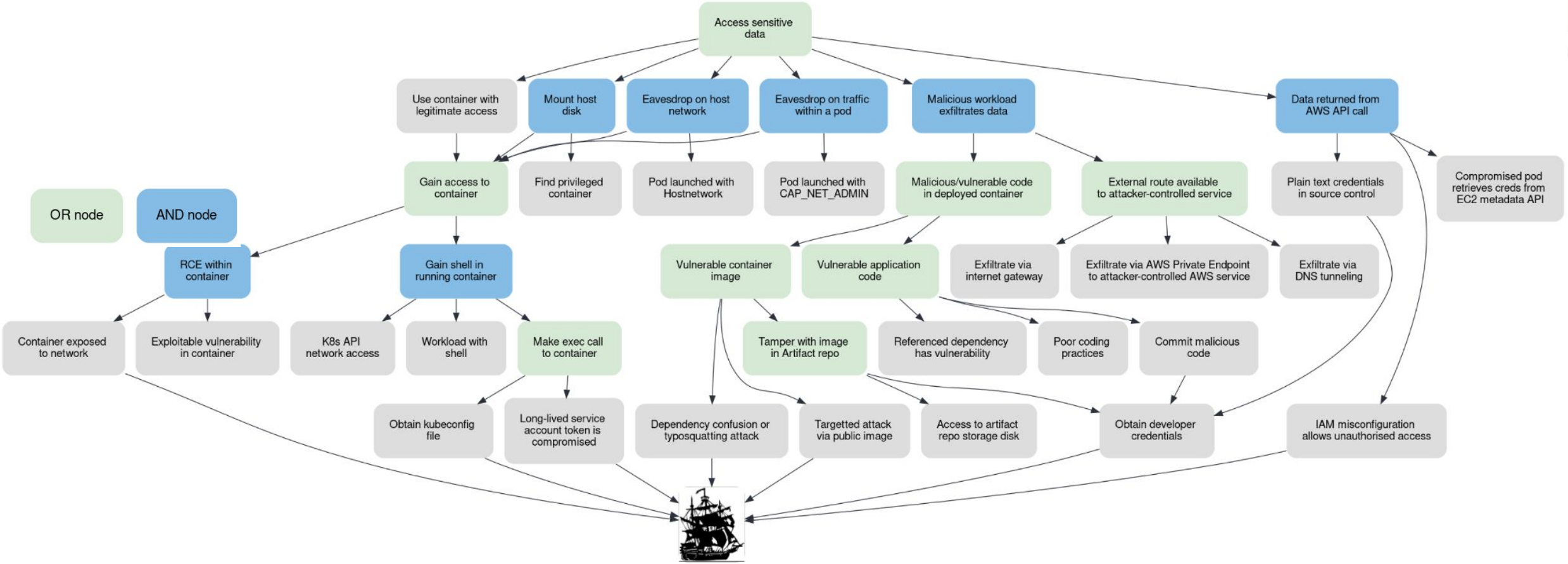
- **Attacking volumes**
- **Dangers of host mounts**
- **No encryption at rest**
  - **Etcd**
  - **Persistent storage**

What Can Go Wrong: IAM

- **Pivot to attack Cloud APIs**
- **Pivot to attack k8s API**
- **Exploit app flows**

What Can Go Wrong: Summary

- Lots!
  - Workloads
  - Networking
  - Storage
- Kubernetes Control Plane
  - Supply chain
  - Insider threats
  - CVEs
  - Misconfiguration
  - Stolen credentials
  - Compromised underlying infrastructure
  - ...and much more!



https://github.com/controlplaneio/threat-modelling-labs

controlplane

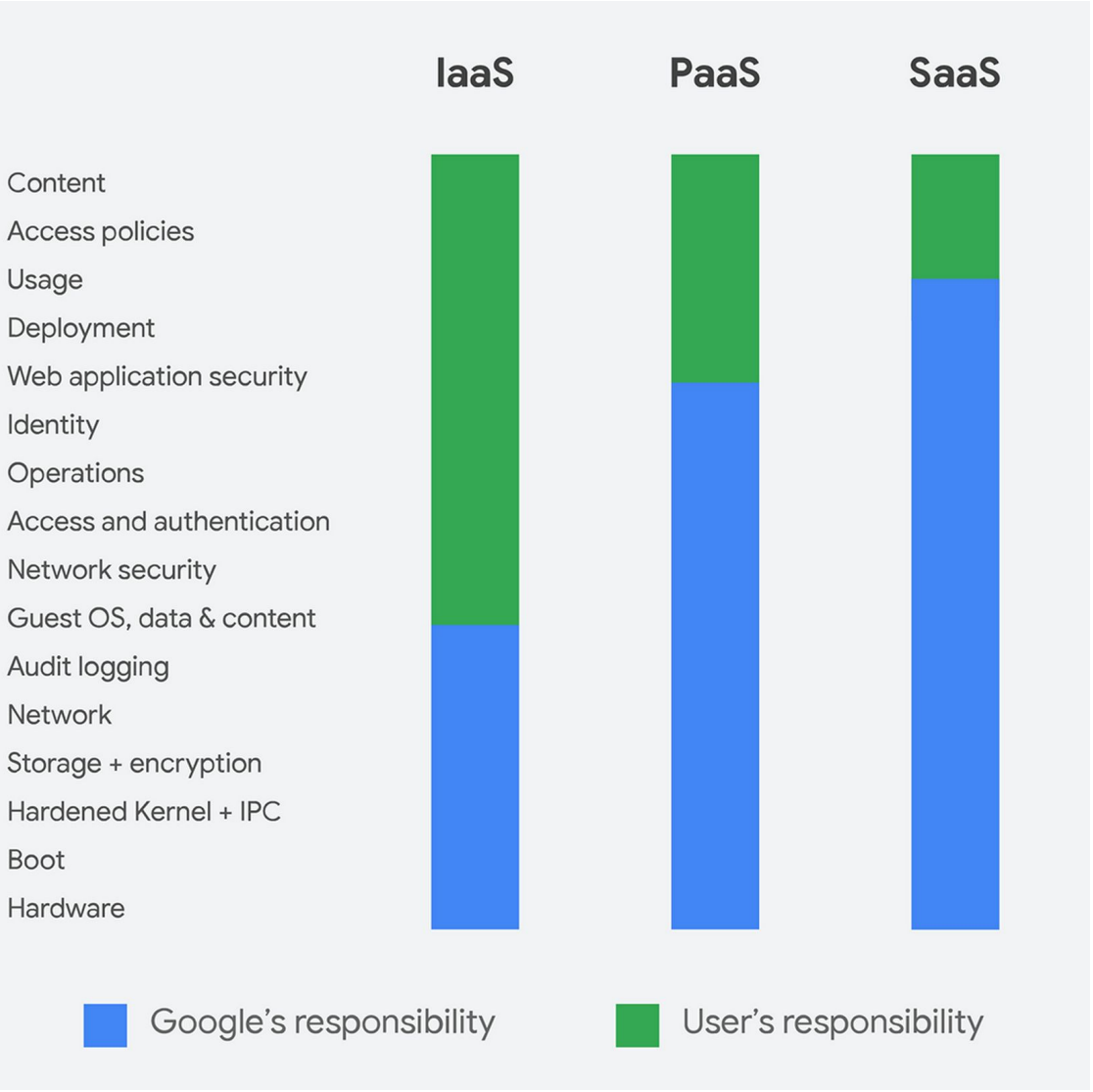
What Will We Do About It?


KubeCon
CloudNativeCon
Europe 2022


controlplane

We have all performed a Personal Threat Model

Shared Responsibility Model



- Cloud provider makes **some security and maintenance guarantees**
  - e.g. AWS Artifact, GCP Compliance
- Clients are responsible for the **usage and configuration of the system**

Risk Management

Risk management strategies: based on your risk tolerance!

- **Avoid**: Remove feature that generates risk
- **Mitigate**: Reducing risk by implementing security controls
  - Enforce design or procedural changes that will reduce risk to an acceptable level
- **Accept**: Risk's impact is minor or its probability of occurrence is very low
- **Transfer**: Disclaimers, insurance policy

All threats and risks should be **explicitly addressed** using one of the risk management strategies. None should be ignored.

Mitigating Risk: Types of Control

- **Preventive** - can be good configuration, or controls in CI/CD e.g. linting/tests that prevent misconfigured/vulnerable clusters from being launched
- **Detective** - detecting bad events within logs
- **Corrective** - e.g. log event triggers corrective Lambda (or equivalent workload), can be a workload/control loop in the clusters
- **“Old school”** controls that are **not effective** in cloud
  - Node segregation: nodes are often multi-tenanted
  - On-prem mindset in cloud
    - Heavily manual change control: restricts deployment
    - Restrictive architectures: difficult to “bin pack” workloads
    - Reliance on detective controls: attacks are implicitly permitted to production!

How to Defend Against our Scoped Attacks?

- **Kubernetes and Cloud IAM** is at the centre of Cloud Native security
  - **Review permissions** regularly
  - **User management** processes are key
  - **Review onboarding/offboarding processes** (Joiners/Movers/Leavers, or JML)
  - Implement **strong access control policies**
- **Default Kubernetes service accounts should not be used**
  - Create a dedicated service account for every workload (it's "workload identity")
  - Workloads will interact with the Kubernetes API, or the cloud via a workload identity
- Cloud providers services can help, e.g **IAM roles for Service Accounts (IRSA)** **Identity** in GKE
  - IRSA uses **Service Account Token Volume Projections**
  - This is just one benefit of using a managed offering, e.g. EKS, GKE, AKS
- For cryptographically strong workload identities, **SPIRE** can be used
  - SPIRE is a production-ready implementation of the **SPIFFE** APIs

How to Defend: Workloads

- Least privilege **securityContext** should be set for pods
- **Admission control should be set up to ensure that non-compliant pods will not run on the cluster**
  - Note: Pod Security Policies (PSPs) are deprecated within K8s v1.21, although they will remain in the API until v1.25
  - Custom policy enforcement can be achieved using technologies such as OPA (e.g. via Gatekeeper) or Kyverno
- Use **container Intrusion Detection Systems (IDS)**, e.g. Sysdig Falco, to spot malicious behaviour in running containers
- Build **automated security testing** into pipelines
  - do not neglect **governance and processes**
  - these must be in place to **make decisions** based on testing output

How to Defend: Workloads - Supply Chain

- **Scan for CVEs** in dependencies using tools like Trivy
- Think about how much an attacker could benefit from seeing your code and **restrict access** appropriately
- Build security
- Organisations can use tools such as **in-toto** to increase trust in their pipelines and artefacts
- The CNCF Security Technical Advisory Group (tag-security) have published a Software Supply Chain Security Paper

How to Defend: Networking

- Enforce **network policy**, e.g. using Cilium
- **Restrict Kubernetes API access**
- Apply **admission control policies** which do not allow dangerous networking configurations, such as `hostNetwork` or the `CAP_NET_RAW` Linux capability
- **Secure Ingress** to allowlist approved networks only
- **Pod-pod encryption** can be configured at the application layer (e.g. making use of an internal PKI orchestrated by CFSSL)
  - Also a **service mesh** (e.g. Istio) could be considered, which also helps with workload identities

How to Defend: Storage

- **Secrets management**, for example, using:
  - Hashicorp Vault
  - Kubernetes-external-secrets
- **Admission control** can prevent host mounts entirely, or enforce read-only mount paths
- Persistent Volume **encryption at rest**
- Persistent Volume **access controls**

How to Choose Controls

- Once threats have been identified, **security requirements** can be derived
- These requirements are called **controls** or **countermeasures**
  - They provide mitigations against the enumerated threats
- 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"**
- Important to **prioritise controls**
  - Assess the **residual risk level** associated with the final selection

Did We Do a Good Job?

Ongoing Threat Modelling Responsibilities

- **Assess identified threats** and security requirements against reference standards and policy
- Conduct **internal reviews**
- Conduct and record the results of **testing**, both automated and external penetration testing
- Identify, record and **confirm that residual risks are at an acceptable level**
- **Regularly review and update** the model with new threats and attacks
  - New threats might be mitigated without additional controls being required

Mapping Controls to Attack Tree Nodes

- **Select security controls** for implementation
- **Map these controls onto an Attack Tree**
  - This provides a direct means of evaluating effectiveness
  - Visual representations are easier to reason about
- Identify how many nodes and “branches”of the attack tree are covered by the mitigating controls
- **Assess the resulting security of the system**
  - If enough security controls are defined, any **new attacks added to the tree should be mitigated by existing controls**

Automated Testing and CI/CD

- **Automated tests** should be devised and run
  - As part of a CI/CD pipeline
  - Every time the **system is deployed** or a **code change made**
- High level test scenarios can be mapped to the **Security Requirements**
- Each test scenario may consist of multiple assertions
  - Document within the test suite itself
  - Add a test pass/fail column if needed
- **Failed tests** indicate a security requirement has not been met
  - **Residual risk** will remain or **remediation** is required
- **Tests should be against threats**
  - Rather than the implementation of a control

*It only takes **one project**, to identify a **thousand threats**,*
*And it only takes **one threat**, to burn a **thousand projects***

[Stereophonics - A Thousand Trees]

Finally…

Rory McCune
@raesene

Replying to @denhamparry and @richburroughs

Now that, is good thinking!

Thank you


KubeCon
CloudNativeCon
Europe 2022


controlplane