Threat Modelling & Kubernetes Security Training

sig-security / Financial Services User Group, CNCF

FSUG – Financial Services User Group

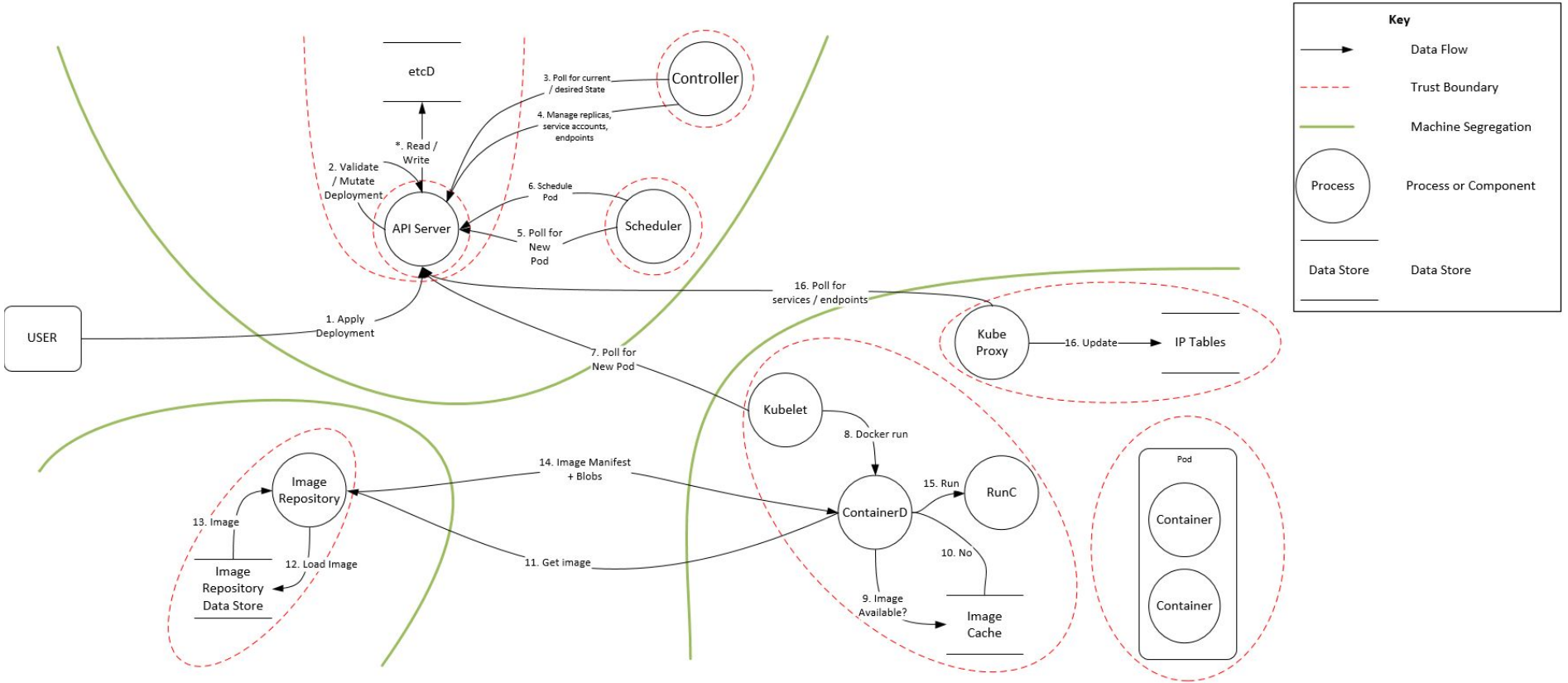
- For those interested in running Kubernetes in highly regulated environments, particularly financial services (https://github.com/cncf/financial-user-group)
- 15-20 financial institutions
- Identified key issues from group perspective
- Top two items
  - What controls should be applied to Kubernetes & how can we test them?
  - Skills gap – Need to train security engineers in Kubernetes

1: What controls to use, how to test them?

Threat Modelling with Attack Trees - Once Attack Trees are created one can use them to:

- Map security controls and standards onto the tree in order to understand their coverage
- Automate Security regression tests
- Enable the SOC to develop a protective monitoring policy and provide situational awareness in the event of a suspected breach
- Add colour to discussions between security (SOC, IR, Forensics) and project teams

Kubernetes Trust Boundaries



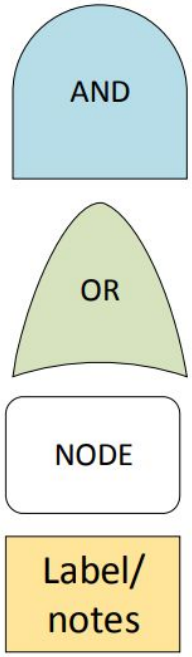
Attack Trees

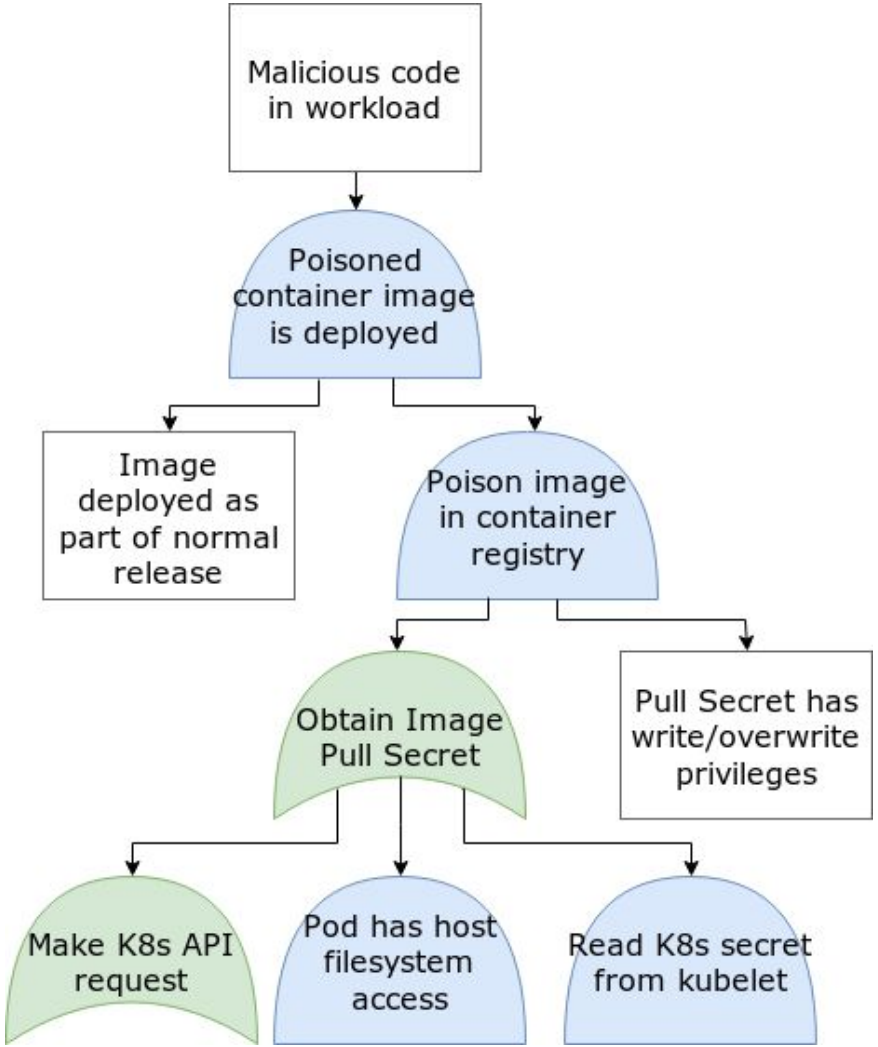
*“Attack trees provide a **formal, methodical** way of **describing the security of systems**, based on varying attacks. Basically, you represent attacks against a system in a **tree structure**, with the **goal as the root node** and different ways of **achieving that goal as leaf nodes**.”*

- Bruce Schneier (1999)

https://www.schneier.com/academic/archives/1999/12/attack_trees.html

Attack Tree: Simple Example





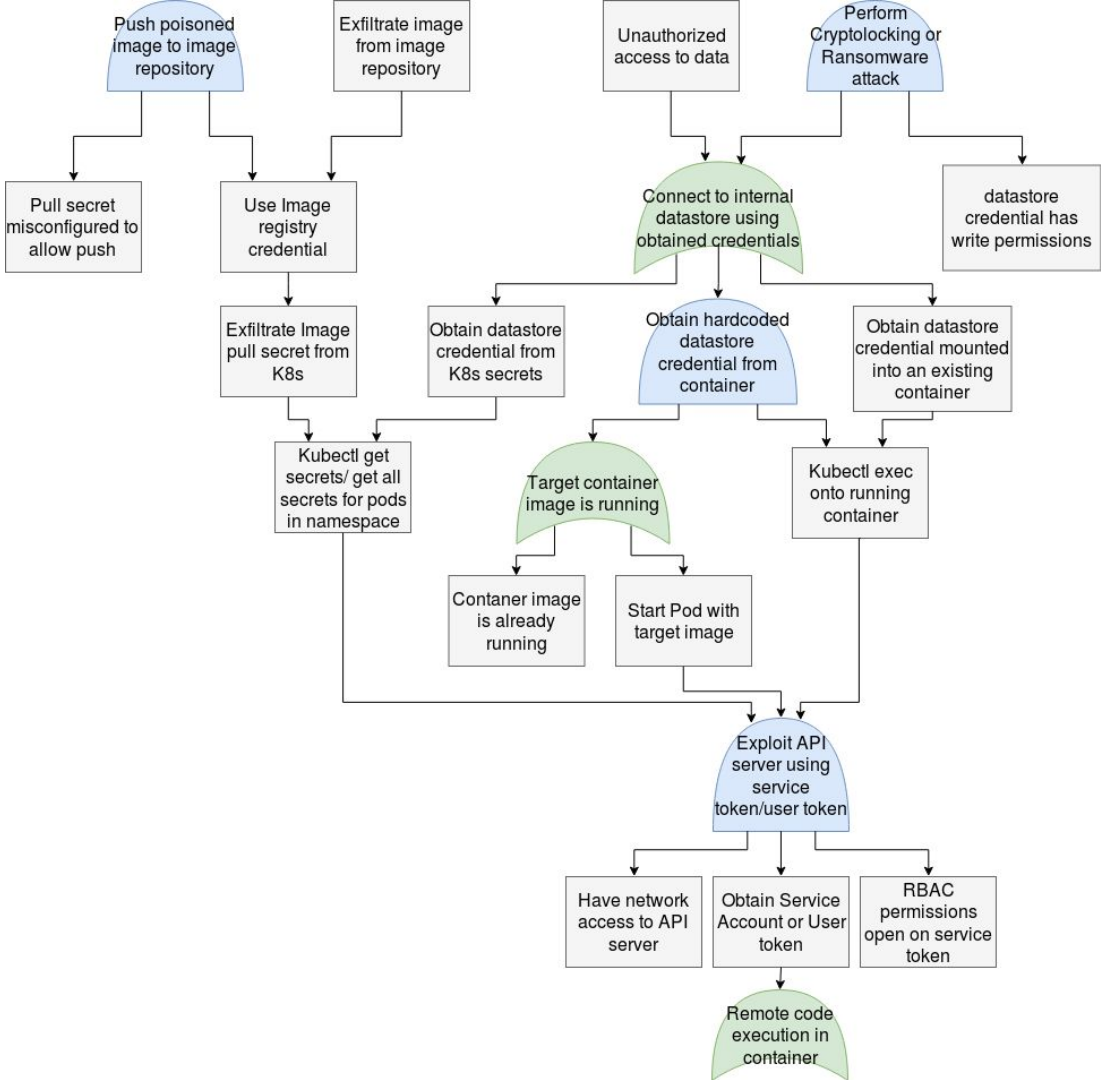
Attack Tree: Bottom Up

AND

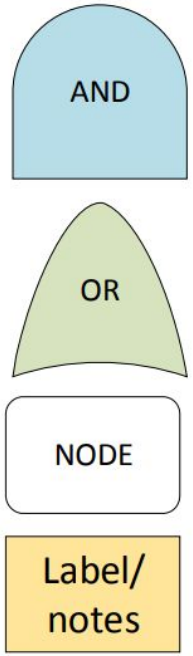
OR

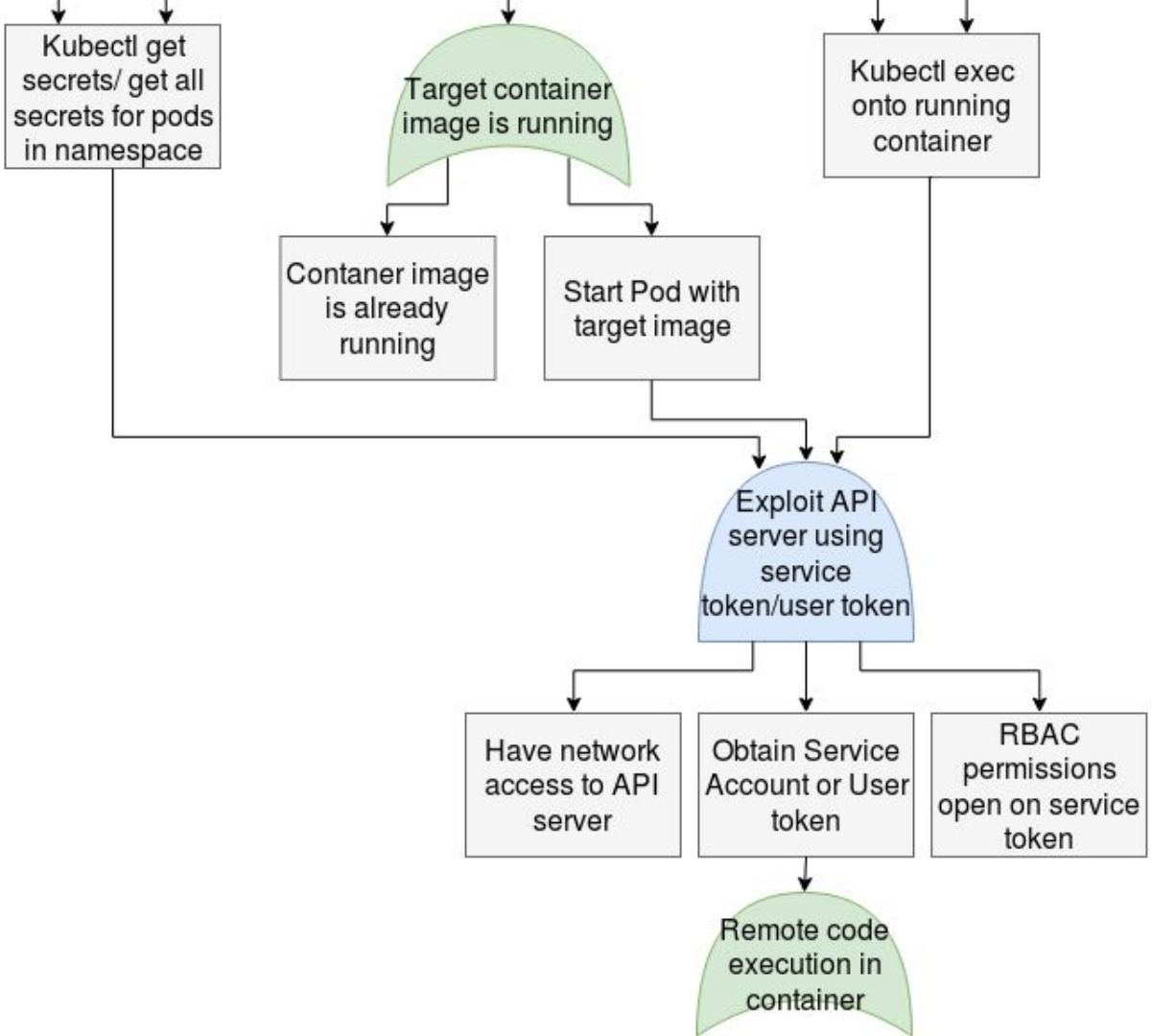
NODE

Label/ notes

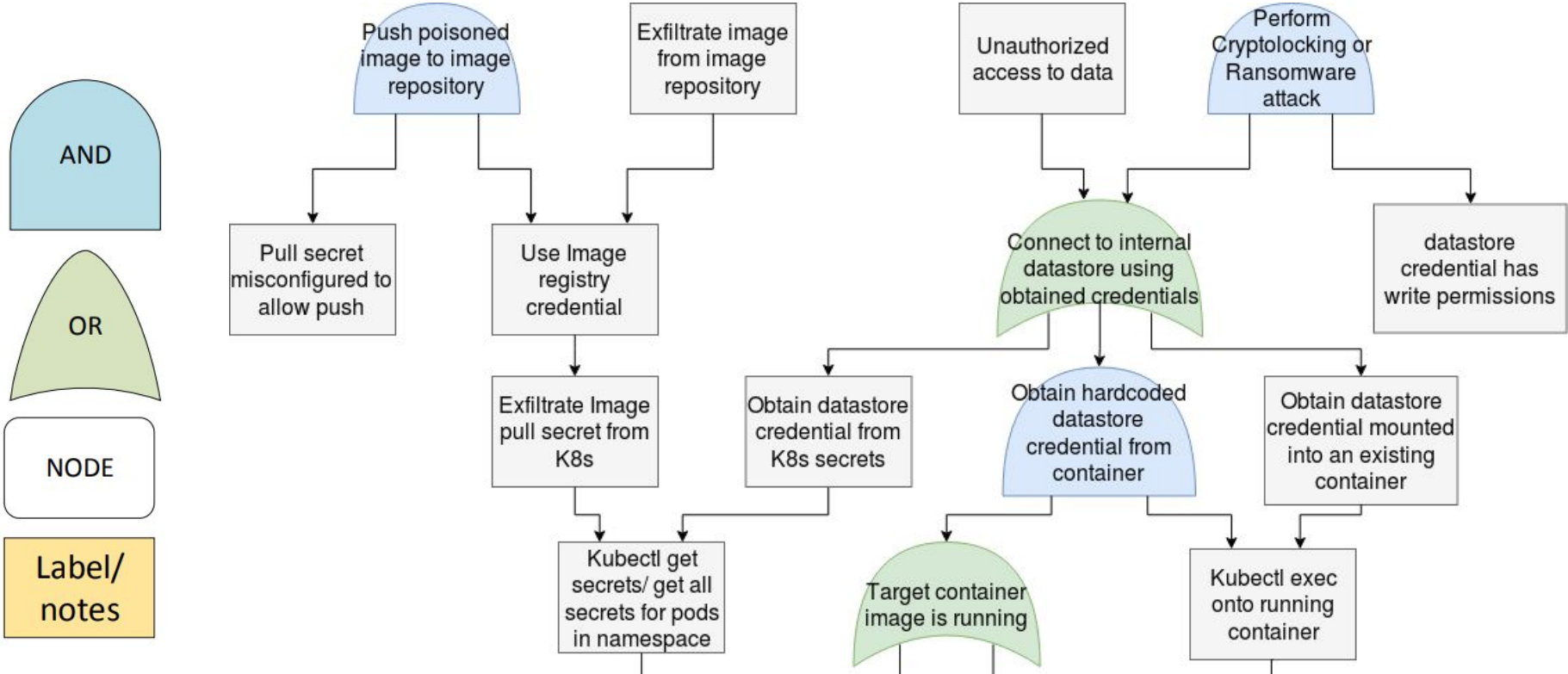


Attack Tree: Bottom Up





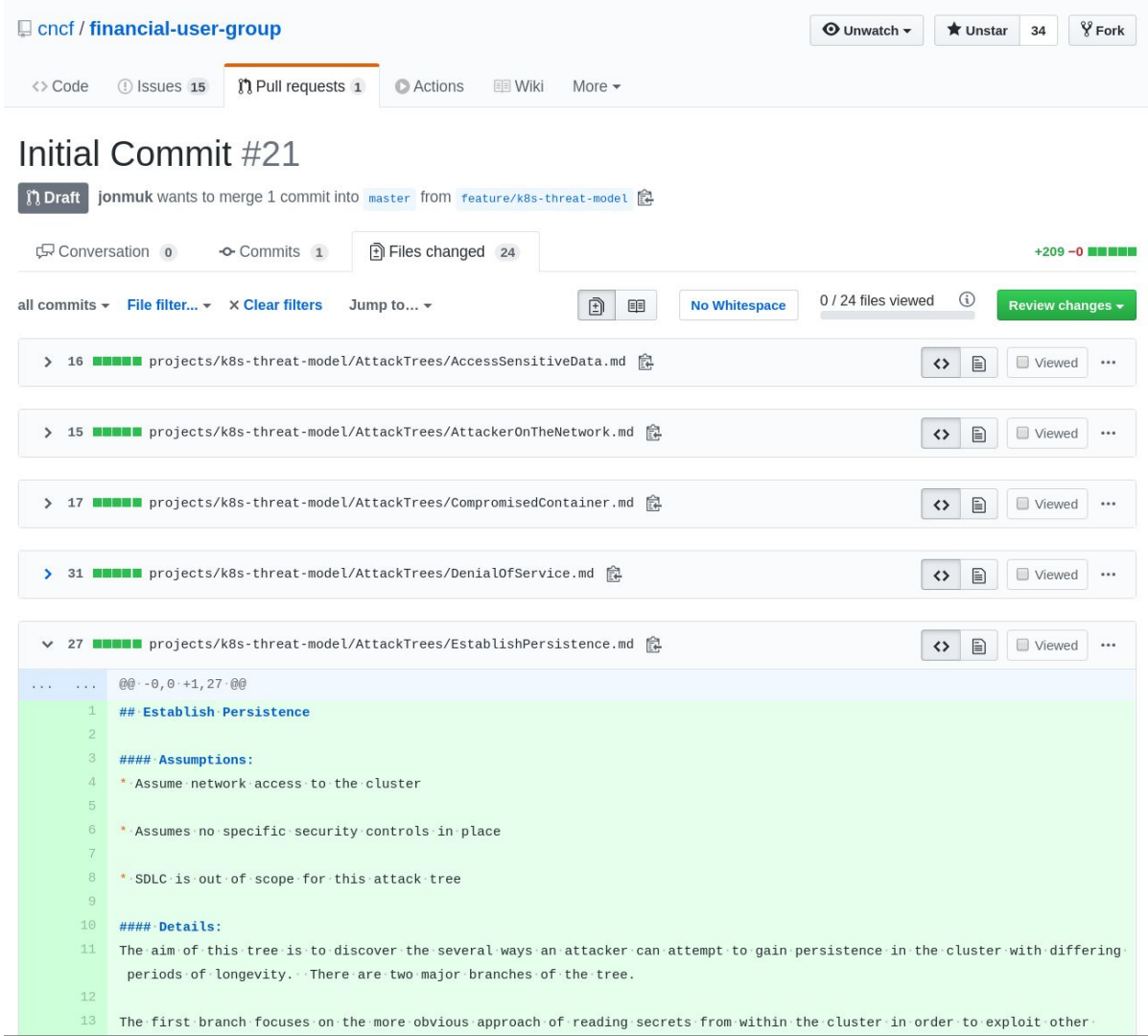
Attack Tree: Bottom Up



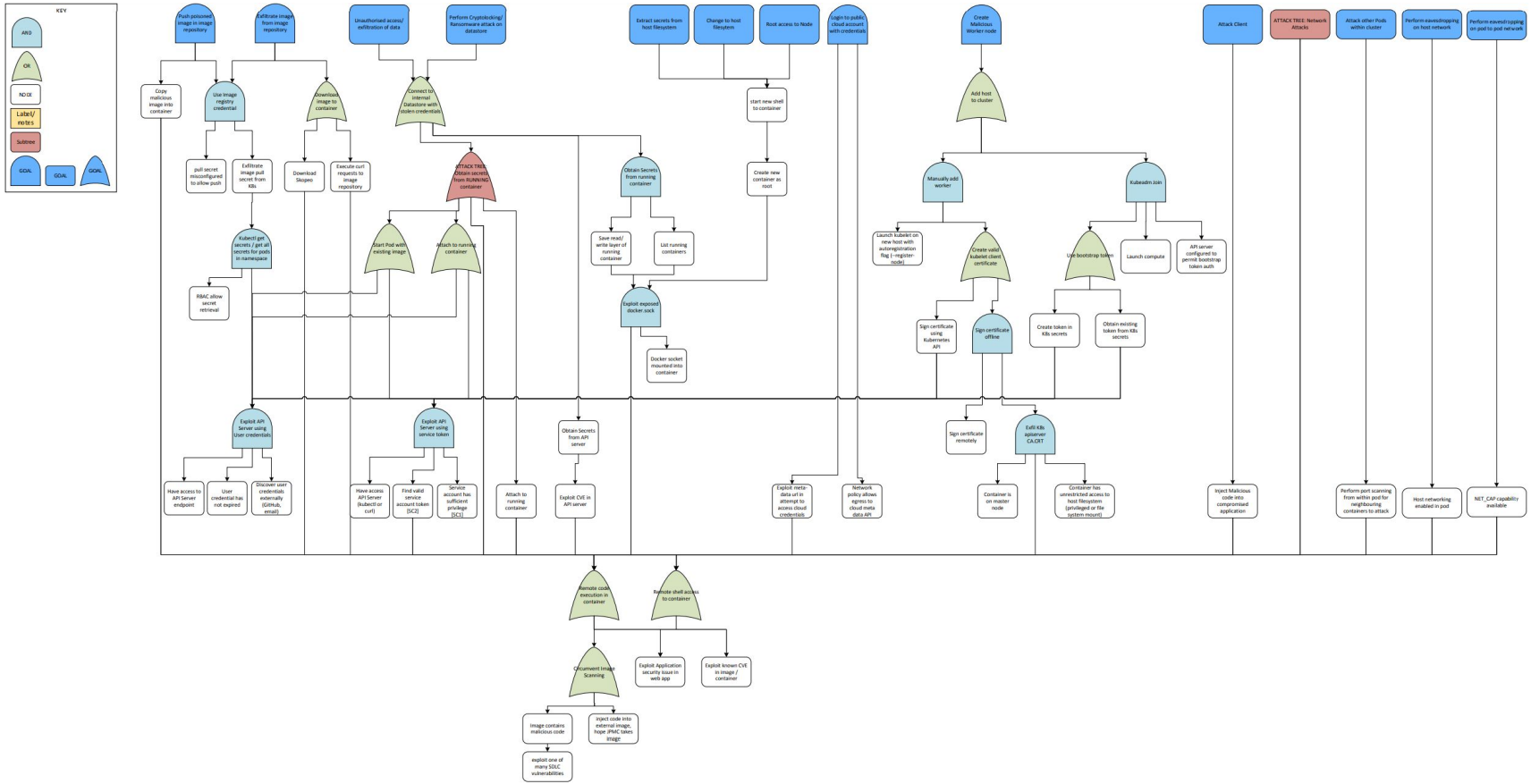
Kubernetes: Attack Trees

- Collaborative work effort with the CNCF Financial User Group
- The initial set of Attack Trees are now open sourced and available on GitHub:
  - https://github.com/cncf/financial-user-group/tree/master/projects/k8s-threat-model

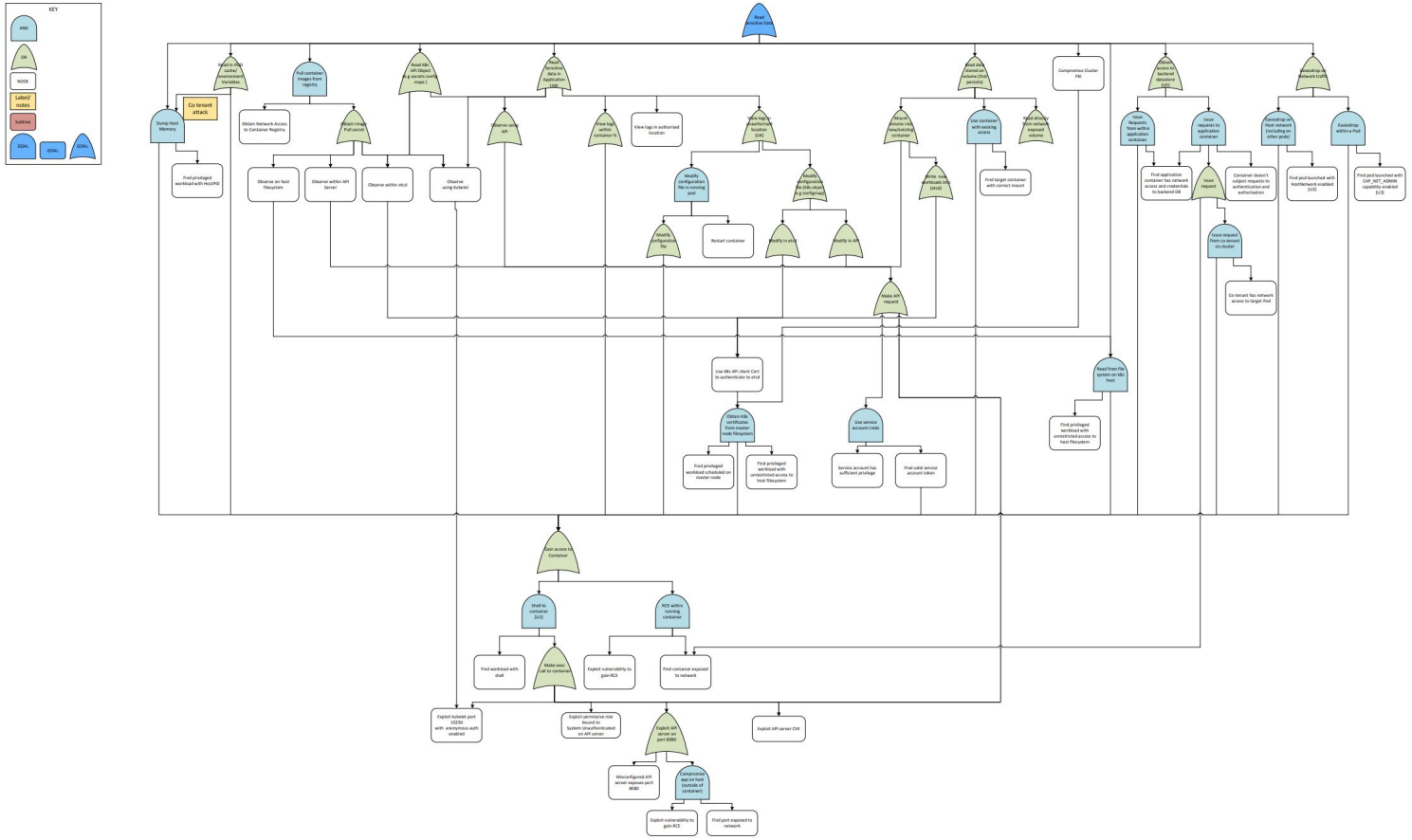
- **Next Steps: More automation, less visio**
- **Breach Simulation?**



Kubernetes Attack Trees - Compromised Container Scenario



Kubernetes Attack Trees - Read Sensitive Data



2: Addressing the skills gap

- Threat models are great, but how do we get started if we don’t know anything about Kubernetes?
- Think like an attacker
- **Hands on, less slideware**
- **Ultimate Goal** – Multi-user experience. Read Team, Blue Team, Forensics

kubesim.io

- **infrastructure** deployment
- **cluster** provisioning and workload configuration
- **scenario** runner with challenges, hints, and scoring
- raw **command line experience**
- **open source core** at https://github.com/kubernetes-simulator/simulator

Select Scenarios

- `container-ambush`
- `container-defeat-in-detail`
- `container-phalanx-formation`
- `etcd-inverted-wedge`
- `master-bait-and-bleed`
- `master-encirclement`
- `network-feint`
- `network-hammer-and-anvil`
- `network-hedgehog-defence`
- `network-swarming`
- `node-amphibious-operations`
- `node-raiding`
- `node-shock-tactics`
- `policy-echelon-formation`
- `policy-fire-support`
- `policy-force-dispersal`
- `policy-vertical-envelopment`
- `rbac-contact-drill`
- `rbac-sangar`
- `rbac-flanking-maneuver`
- `rbac-shoot-and-scoot`
- `secret-high-ground`
- `secret-tank-desant`

Scenario: rbac-sanger

Scenario: After a change to your cluster's RBAC, you reviewed the audit logs and noticed the default service account in the rbac-sanger namespace is making successful calls to the API server. As per your organisation's policy, the default service account in every namespace should not have permissions to query the API server. The suspect calls have been traced back to the frontend container.

Starting Point: frontend pod in the rbac-sangar namespace

Task 1: Can you use the rbac-sangar default service account to retrieve a secret from the application pods in the rbac-sangar namespace?
Task 2: Find out how the default service account gained these elevated permissions and fix the issue.
Task 3: What change can you make to the frontend deployment to mitigate this issue?
Task 4: What change can you make to the application deployment to mitigate this issue?

Setup

Using your own AWS keys:

- Git clone https://github.com/kubernetes-simulator/simulator
- ```
  AWS_DEFAULT_REGION=us-west-1 \
    AWS_PROFILE=my-profile \
    make run
  ```
- `$ simulator ssh attack`
- Follow the instructions!
  ```

K8S Breach Lessons

- Application workloads
- Workload configuration
- Cluster configuration - developer
- Cluster configuration - operations
- Cluster deployment

-rum-animations.html

K8S Breach Lessons

- Application workloads
  - container images (app, OS, deps, filesystem conf, user, baked-in secrets), malicious images
- Workload configuration
  - pod spec (privileged/PSP config, env vars, service accounts, file mounts, runAsUser, seccomp/selinux), runtime (docker, crio), priv workloads, identity, datastores, traffic sniffing/nmap
- Cluster configuration - developer
  - authn/RBAC, NetPol/networking
- Cluster configuration - operations
  - admission control, etcd encryption, control plane TLS, local/remote registry, dashboard/API, federated identity
- Cluster deployment
  - control plane config, user access, topology, transport & TLS, infrastructure/cloud, OS

-rum-animations.html

How to Train your Red Team

- FSUG: https://github.com/cncf/financial-user-group
- Open source: https://github.com/kubernetes-simulator/simulator
- Attack Trees: https://github.com/cncf/financial-user-group/tree/master/projects/k8s-threat-model