Securing Kubernetes Deployment @ Scale

Sohini Mukherjee

Senior Security Partner

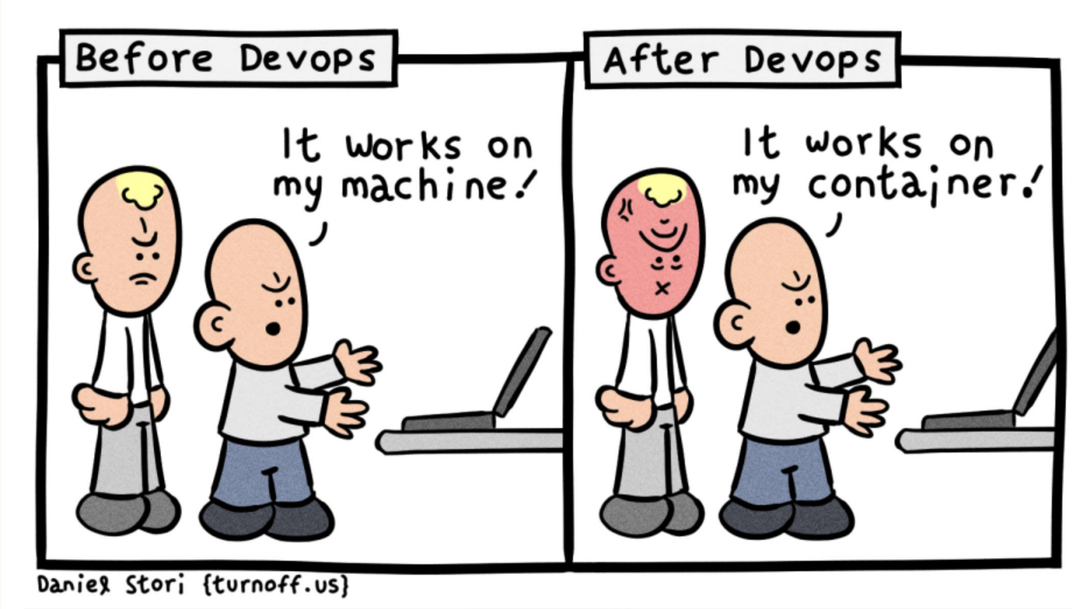
LinkedIn

Agenda

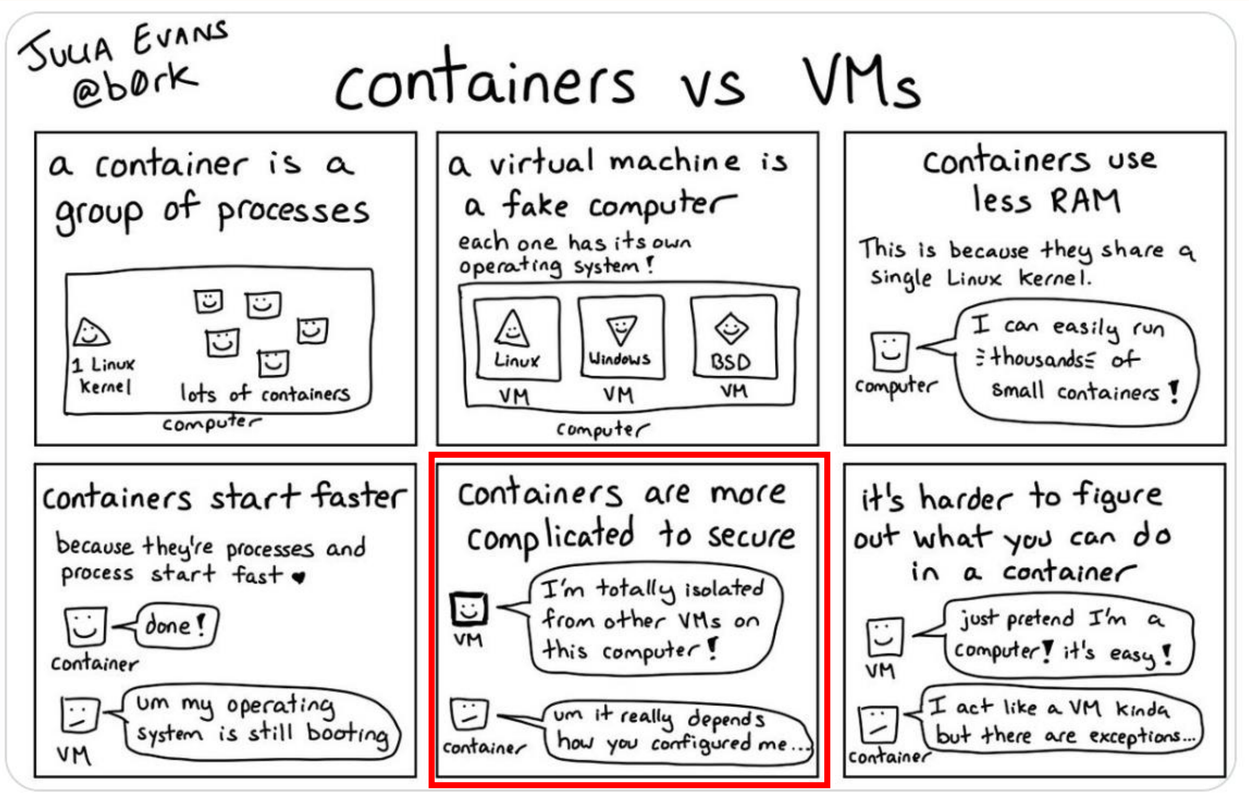
1. Kubernetes Architecture
2. Anatomy of Attacks
3. Securing Kubernetes Deployment
4. Conclusion

Standing on the shoulder of Giants

Containerization

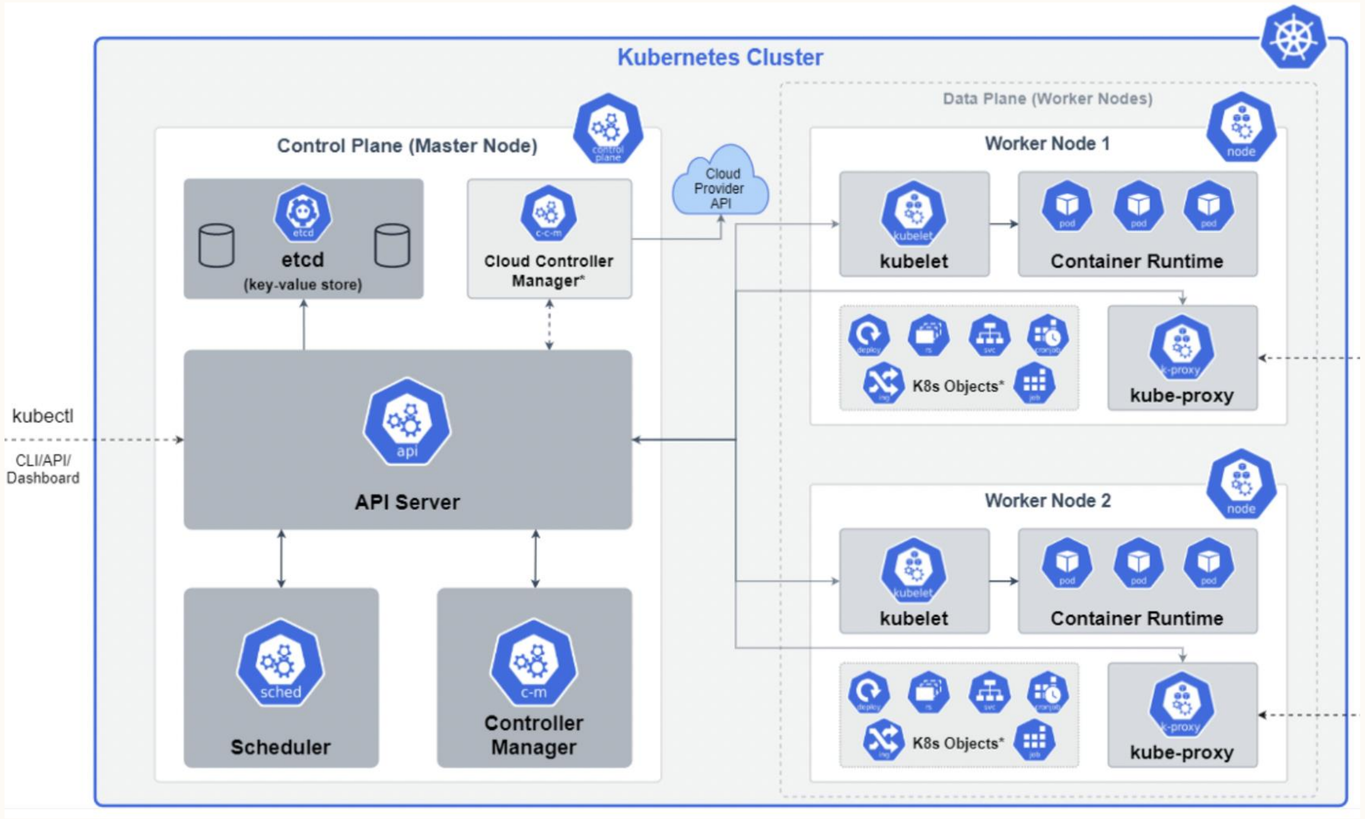


Virtualization vs Containerization



The Architecture

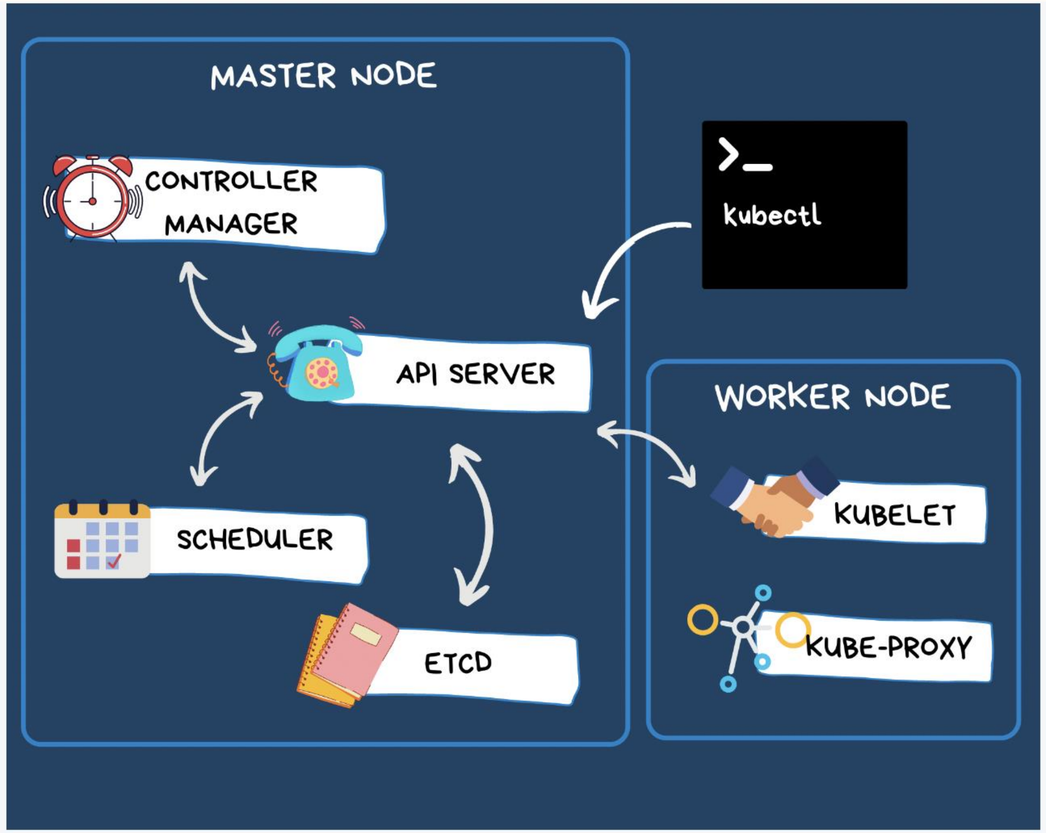
Kubernetes Architecture



Source: https://medium.com/devops-mojo/kubernetes-architecture-overview-introduction-to-k8s-architecture-and-understanding-k8s-cluster-components-90e11eb34ccd

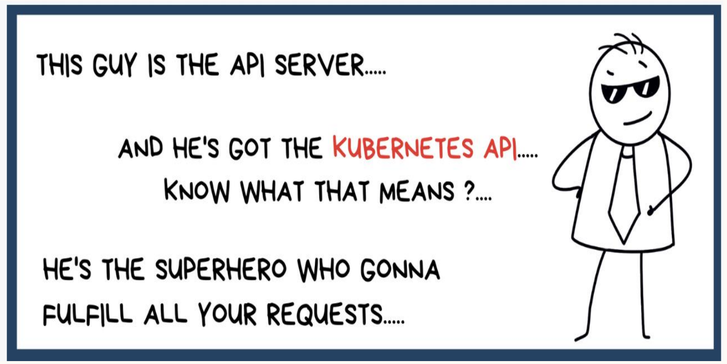
Kubernetes Architecture

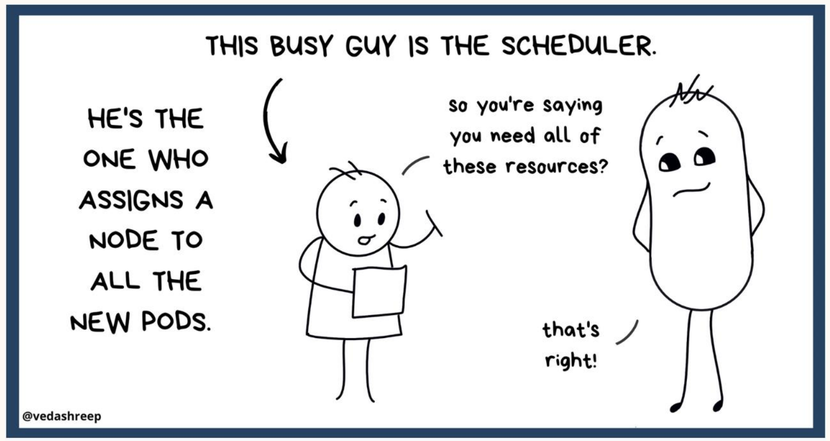
The Cartoon Guide

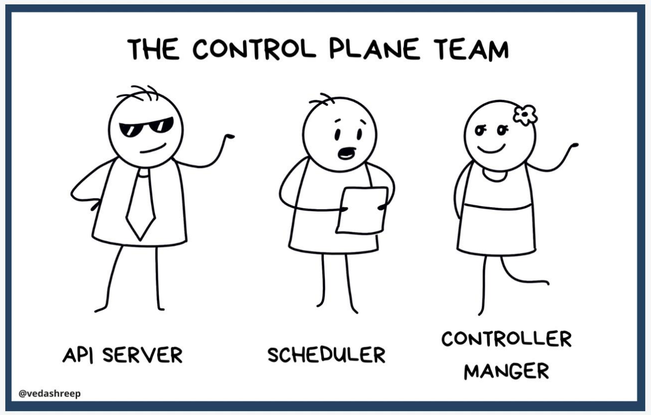


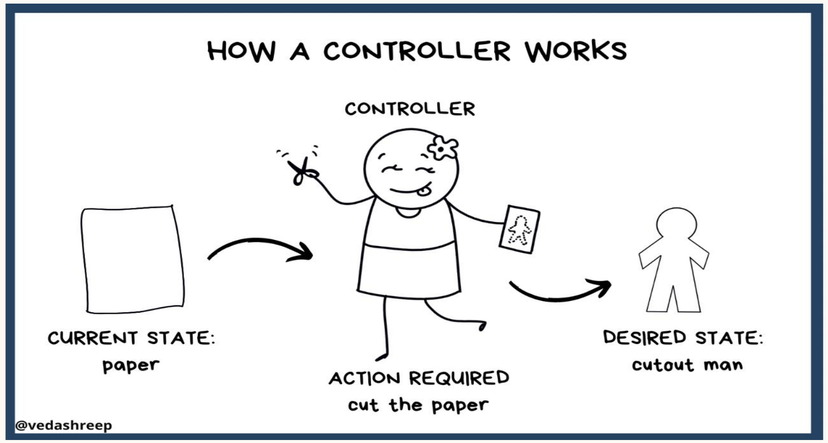
Source: https://gochronicles.com/kubernetes-architecture/

Control Plane





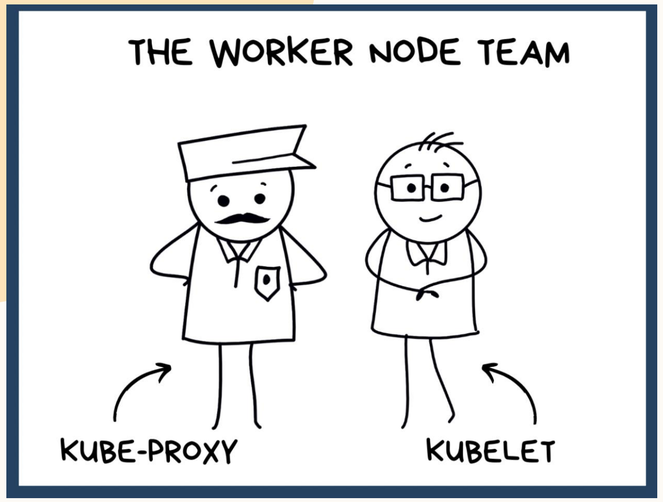


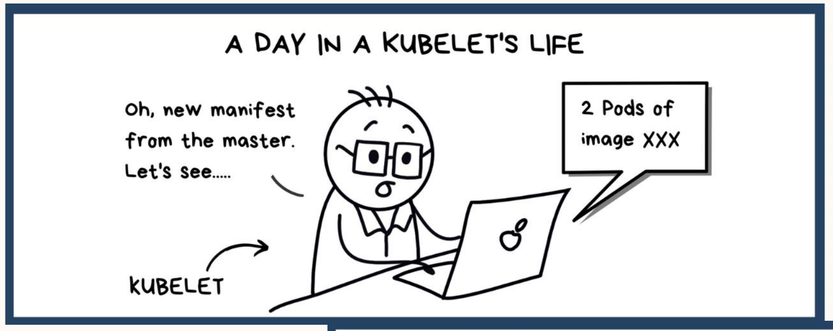


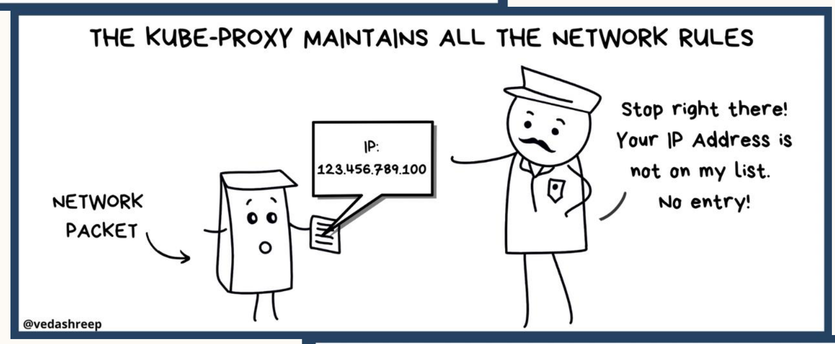
Etcd : key-value store

Source: https://gochronicles.com/kubernetes-architecture/

Worker Nodes







Source: https://gochronicles.com/kubernetes-architecture/

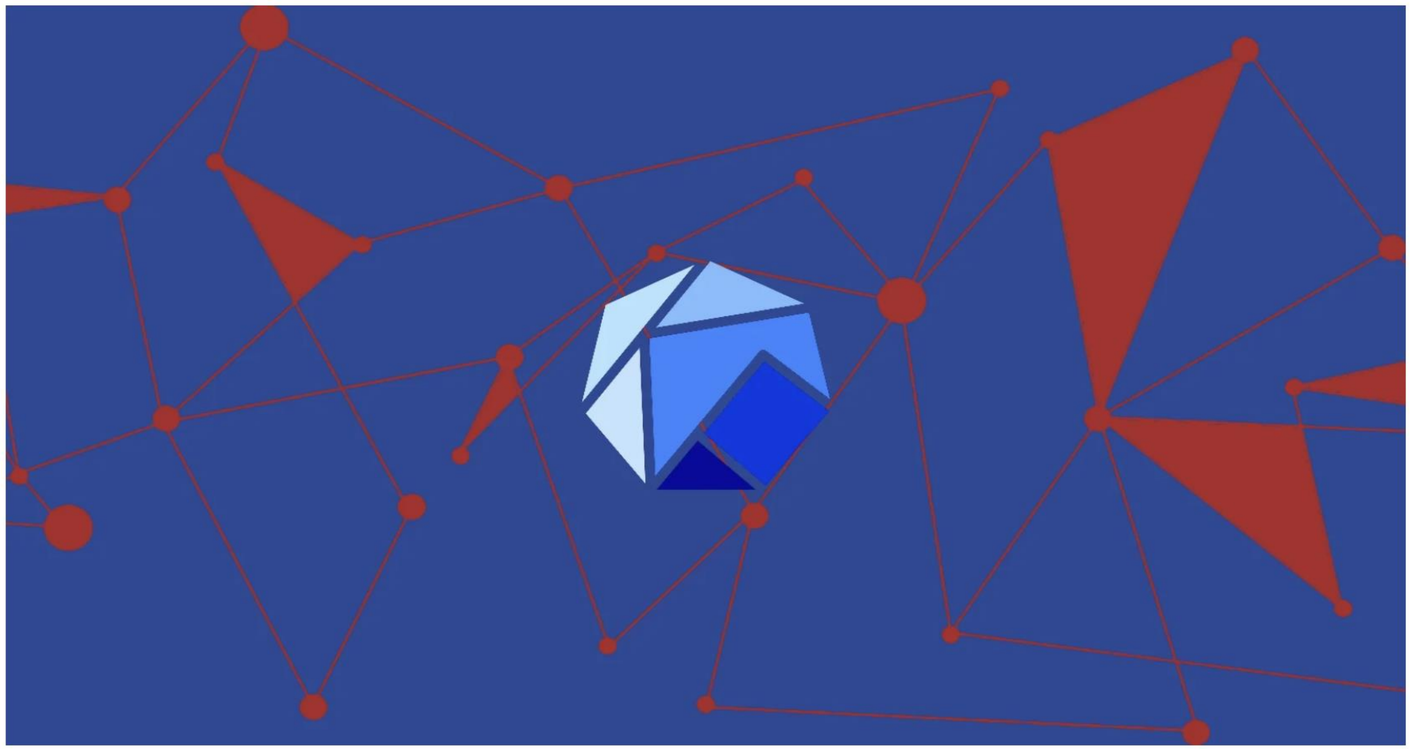
Attack Scenarios

Cryptomining

Microsoft warns of cryptomining attacks on Kubernetes clusters

By **Sergiu Gatlan**

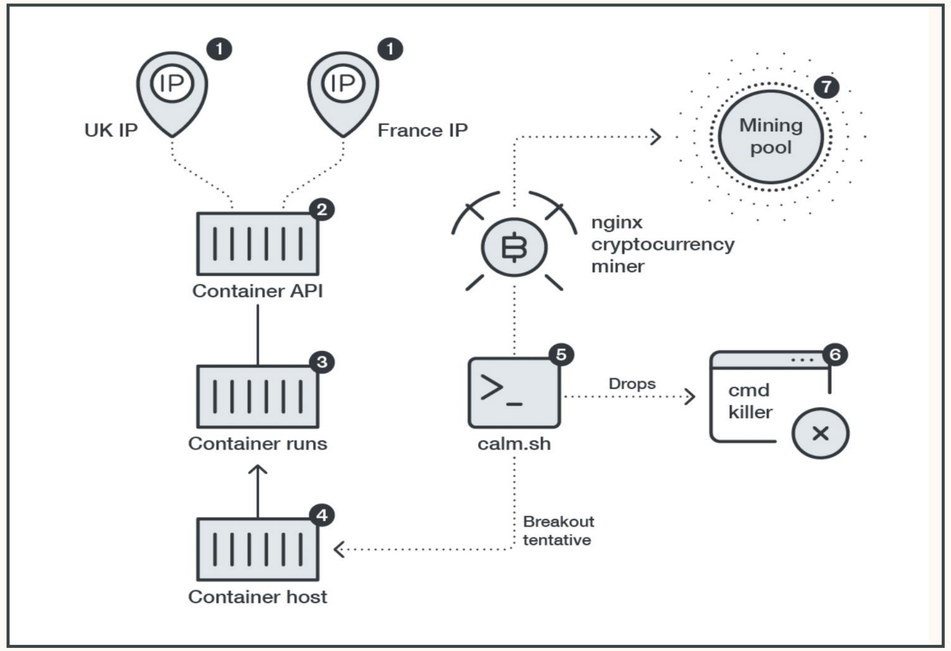
June 9, 2021 01:05 PM 0



Microsoft warns of an ongoing series of attacks compromising Kubernetes clusters running Kubeflow machine learning (ML) instances to deploy malicious containers that mine for Monero and Ethereum cryptocurrency.

Source: https://www.bleepingcomputer.com/news/security/microsoft-warns-of-cryptomining-attacks-on-kubernetes-clusters/

Container Escape



- Attacker's Network Scanner detects Container exposed API
- API uses *create* command to pull a container image called "gin" from its registry and deploys a malicious Container
- One of the options used as a deployment parameter is *privileged*, - this is a requirement for this specific escape technique
- Entry point is shell script *calm.sh*
- *calm.sh* drops another shell script *cmd*
- *calm.sh* calls *nginx*, which is a cryptocurreny miner
- Attacker attempts to fly under the radar by naming the miner '*nginx*'

Source: https://www.trendmicro.com/en_us/research/21/b/threat-actors-now-target-docker-via-container-escape-features.html

Slioscape

Source: https://unit42.paloaltonetworks.com/windows-server-containers-vulnerabilities/

Siloscape

- Windows leverages *silos*, a Windows Kernel feature, to provide isolation
- Microsoft doesn't endorse silos in hostile muti-tenancy environments and instead recommends Hyper-V

- Exploit path:

1. Create a symbolic link for the *host's C: drive*.

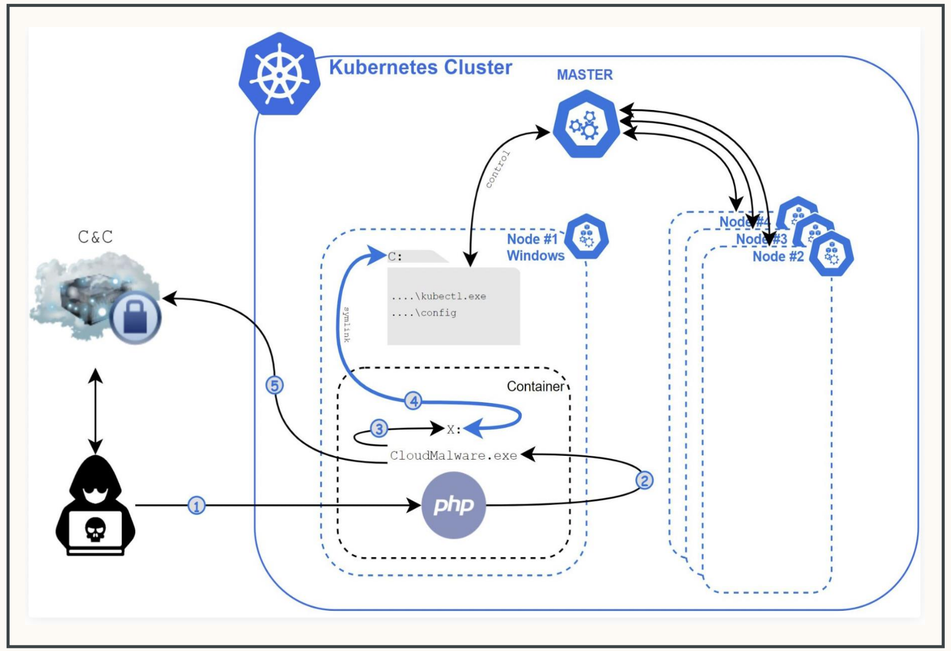
2. Gain *Tcb* privileges by DLL Injection to a special process in Windows Containers called *CExecSvc*

3. Make said symbolic link global

4. Access files on the host's file system

Source: https://unit42.paloaltonetworks.com/windows-server-containers-vulnerabilities/

Siloscape



- Opens backdoor into poorly configured Kubernetes clusters to run malicious containers like cryptojackers

- TTP:
- Targets web servers for initial access, using known vulnerabilities
- Uses Windows Container Escape techniques to escape the container and gain code execution on the underlying node.
- Attempts to abuse the node's credentials to spread in the cluster.
- Connects to its C&C server over the Tor network.
- Waits for further commands

Source: https://unit42.paloaltonetworks.com/windows-server-containers-vulnerabilities/

Attack Matrix

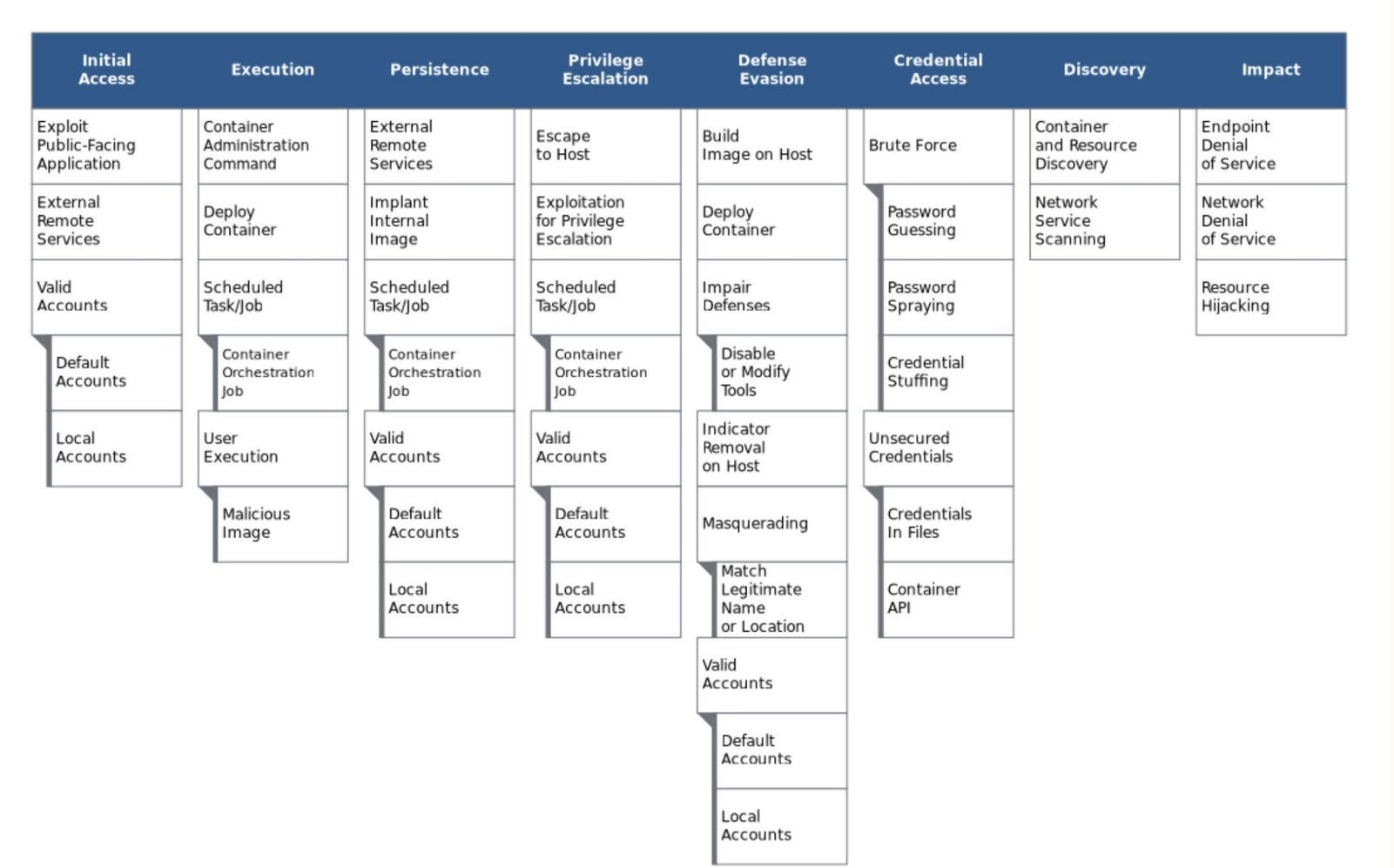
By Microsoft

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	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	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)		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					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

Source: https://www.microsoft.com/security/blog/2020/04/02/attack-matrix-kubernetes/

MITRE ATT&CK

Containers (and K8s)



Source: https://medium.com/mitre-engenuity/att-ck-for-containers-now-available-4c2359654bf1

Securing Kubernetes

Guardrails

Securing Kubernetes Hosts and Kubernetes Components

Securing Kubernetes Hosts

- Harden underlying host by installing the latest version of operating system
- Patch Management & Configuration Management System
- Running the latest version of Kubernetes

  - Kubelet was de-privileged in V 1.22

Control Plane Security

- Access to various Control Plane components (*etcd, Scheduler, API Server*) should be restricted
- Access should be logged and monitored to detect any unauthorized access attempts

Securing Kubernetes Components

- Deny/Restrict SSH access to *Kubernetes Nodes.* Use *kubectl*
- Controlling/Restricting Access to *Kubelet*
- TLS to encrypt in-Cluster communication. Service Mesh like Istio can ensure mTLS

Guardrails

Securing Kubernetes : Build , Deploy & Runtime Phase

Image Management

- Base Images should be signed and controlled by a central deployment pipeline from Artifactory
- Distroless Images
- Continuous Scanning for vulnerabilities

Secret Management

- No hard-coded secrets in Config. Scan for any exposed secrets
- Encrypt secrets at rest

RBAC

- Configure least privilege

Implement Network Policies

- Control Traffic between Pods and Cluster
- Celium, Calico etc.

Pod Security Policies

- Running of privileged containers
- Escalations of root privileges
- Usage of host namespaces
- Usage of host networking and ports
- Usage of the host filesystem
- Requirements for use of a read only root file system
- Security Context: Linux capabilities, SELinux , AppArmor, SecComp etc.

Pod Security Policies

Pod Security Policies are being replaced by Pod Security Standards

Restrictive PSP

```
+ kind: PodSecurityPolicy
+ metadata:
+   name: restrictive-psp
+ spec:
+   privileged: false
+   hostNetwork: false
+   allowPrivilegeEscalation: false
+   defaultAllowPrivilegeEscalation: false
+   hostPID: false
+   hostIPC: false
+   runAsUser:
+     rule: RunAsAny
+   fsGroup:
+     rule: RunAsAny
+   seLinux:
+     rule: RunAsAny
+   supplementalGroups:
+     rule: RunAsAny
+   volumes:
+   - 'configMap'
+   - 'downwardAPI'
+   - 'emptyDir'
+   - 'persistentVolumeClaim'
+   - 'secret'
+   - 'projected'
```

- Pod Security Standards: https://kubernetes.io/docs/concepts/security/pod-security-standards/

Permissive PSP

```
+ kind: PodSecurityPolicy
+ metadata:
+   name: permissive-psp
+ spec:
+   privileged: true
+   hostNetwork: true
+   hostIPC: true
+   hostPID: true
+   seLinux:
+     rule: RunAsAny
+   supplementalGroups:
+     rule: RunAsAny
+   runAsUser:
+     rule: RunAsAny
+   fsGroup:
+     rule: RunAsAny
+   hostPorts:
+   - min: 0
+     max: 65535
+   volumes:
+   - '*'
```

Open Policy Agent (OPA)

Centralized Policy Management

- Domain-agnostic & Open Source Policy Enforcement Tool
- Unified method of enforcing security policy in the entire stack
- OPA can be used for Admission Control
- OPA can be used to build policies to ensure all container images are from trusted sources, pods are not run as root etc.
- OPA integrates directly into the Kubernetes API server
- Policies can be implemented early in the Dev lifecycle (CICD Pipeline) or run ad-hoc to monitor compliance
- OPA can also be used to regulate service mesh architecture
  - e.g. implement policies in Service Mesh to limit lateral movement

Reference: https://github.com/open-policy-agent/opa

Detections

Securing Kubernetes : Runtime

- Shell is run inside a container
- Reverse shell connection detected
- Container mounts a sensitive path from the host such as /proc
- A Sensitive file is unexpectedly read in a running container such as /etc/shadow
- A standard system binary, such as ls, is making an outbound network connection
- A process is spawning an unexpected child process
- First seen privileged process is spawned
- Track unexpected syscalls that can lead to priv escalation

- Know your environment. What is an anomaly?

- https://github.com/falcosecurity/falco
- https://github.com/aquasecurity/kube-bench

Conclusion

Securing Kubernetes @ Scale

- Kubernetes is complex
- Security is hard
- Kubernetes : Security an afterthought
- Know your ecosystem – How secure are you today?
- Data driven approach : pre-fail and post-fail
- Defense-in-Depth

References

- https://github.com/magnologan/awesome-k8s-security
- https://github.com/aquasecurity/kube-bench
- https://github.com/falcosecurity/falco
- https://owasp.org/www-project-kubernetes-top-ten/
- https://www.cisecurity.org/benchmark/kubernetes
- https://github.com/open-policy-agent/opa
- https://kubernetes.io/docs/concepts/security/pod-security-standards/
- https://github.com/madhuakula/kubernetes-goat

Thank you

@sohamaze

LinkedIn