KubeCon

North America 2017

Shipping in pirate-infested waters:
Practical attack and defense in Kubernetes

Greg Castle, CJ Cullen: Kubernetes/GKE Security @ Google

@mrgcastle @cj_cullen

Security in Kubernetes

- Community is working hard on security controls
- Lots of defensive options, where to start?
- How to prioritize?
- Can’t cover all security best practices
- Today’s focus:
  - Helping prevent attacks with existing controls
  - Cluster admin + developer tasks
  - Kubernetes (see blogpost for GKE)
- Documentation has the *how*
- Takeaway: *what*, *why*, and priority

The application code is owned

- K8s threat model assumes app compromise
- Bugs happen
- After code exec is interesting
- Goal: Secure by default, often opt-in first for backwards compat

Demos…tharr be 3!

Attacker lands in clusters at different stages of security evolution

Crawl: App owned == cluster compromise

Walk: App owned + breakout + priv esc == kubelet powers

Run: App owned, no easy escalations: propagate?

Skipper shipper Pirate pew- pew

PyramidSchemeCorp BadSweepstakesApp

- $50 lifetime membership!
- Every 5th member triggers a $100 giveaway!
- Join now or get left behind!
- Get paid in bitcoin?

PyramidSchemeCorp

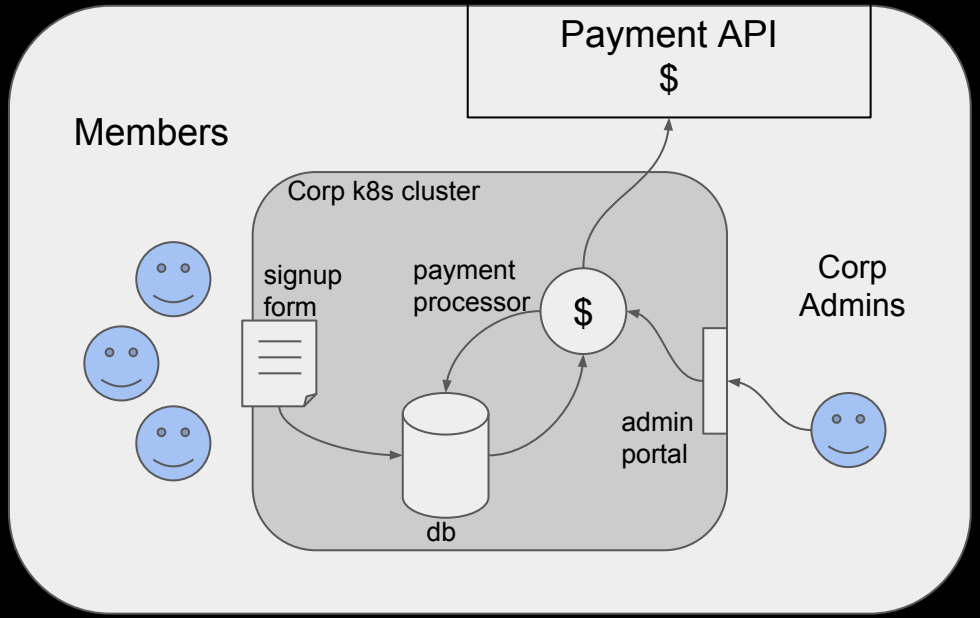
Bad$weepstakesApp

PyramidSchemeCorp BadSweepstakesApp

signup-form
- New member webpage
- Stores info in db

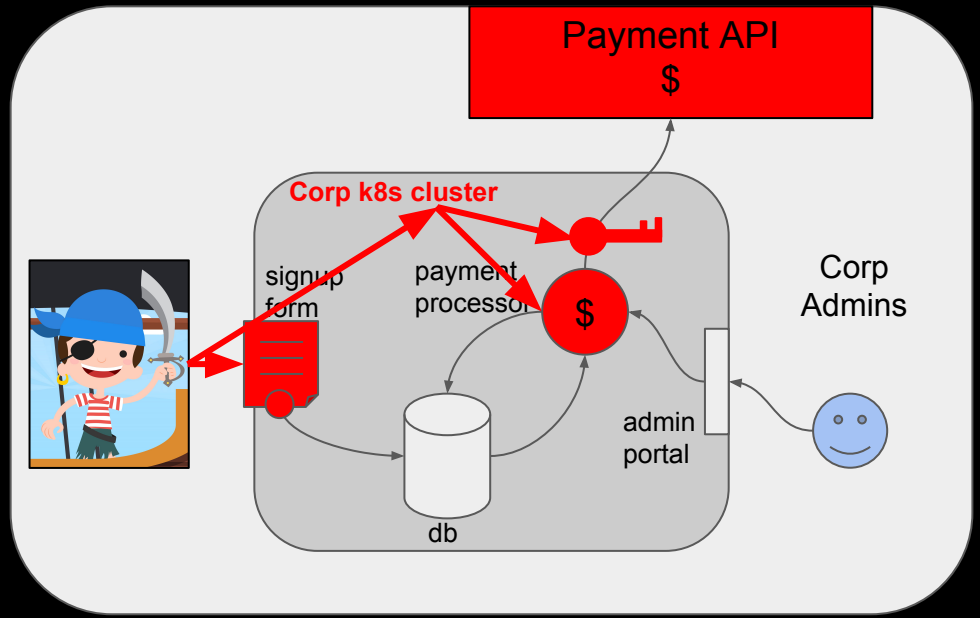
payment-processor
- Charges new members
- Pays winners
- Calls 3rd party API

admin-portal
- Admins grant refunds, pay bribes...




App compromise ==
cluster compromise
Security evolution level: crawl

#1 What happened?

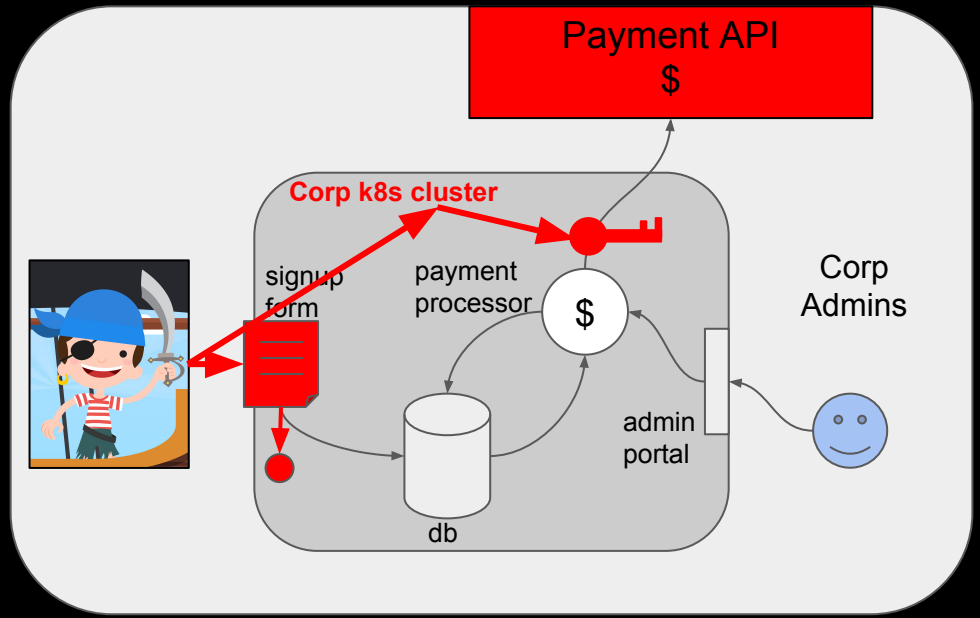


Helping prevent app compromise ➡ cluster compromise

Enable RBAC (disable ABAC), default on GKE for 1.8+. Service accounts no privileges by default. System controllers are least privilege.	Kubernetes 1.6+: start API server with `--authorization-mode=RBAC` GKE 1.6+: `gcloud container clusters create mycluster --no-enable-legacy-authorization`
Use namespaces as boundaries. Payments/frontend different privilege domains. Critical if service account needs API privileges.	`kubectl create namespace payments` `kubectl -n payments run --image=payments`
Force attacker to stay inside the cluster by firewalling access to the master. Makes detecting and evicting attackers easier.	GKE (all versions): `gcloud container clusters update mycluster --enable-master-authorized-networks --master-authorized-networks=8.8.8.0/24`


Root on node ==
kubelet powers
Security evolution level: walk

#2 What happened?



Helping defend against root on node

Limit local escalation
No root
Careful with hostpath mounts
Enforce cluster-wide w/ PodSecurityPolicy (1.8+)
Minimal containers (not fat OS)
	Create PodTemplate with:
	`securityContext:`
	`  runAsUser: 2000`
	`  allowPrivilegeEscalation: false`

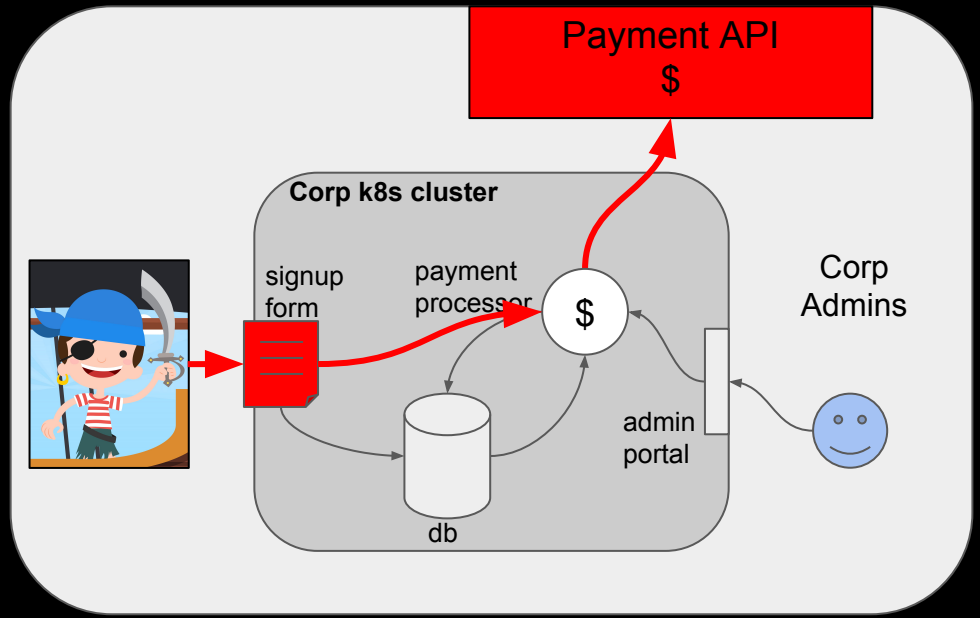
Ensure **least privilege for nodes**:
Enable Node Authorizer/Admission on 1.7+ to protect secrets
	K8s (1.7+): Start kube-apiserver with:
	`--authorization-mode=Node,RBAC`
	`--admission-control=...,NodeRestriction`
	
	GKE (1.7+): automatically enabled

Separate sensitive workloads with anti-affinity, taints, tolerations (1.4+)	`podAntiAffinity:`
	`  requiredDuringSchedulingIgnoredDuringExecution:`
	`  - labelSelector:`
	`  matchExpressions:`
	`  - key: app`
	`  operator: In`
	`  values:`
	`  - signup`
	`  topologyKey: kubernetes.io/hostname`

Kubelet client cert rotation Force attacker to maintain presence, limit time.	K8s 1.8 beta: Start kubelet with: `--rotate-certificates` GKE: Coming Q1 2018


No easy escalations...
Propagate?
Security evolution level: run

#3 What happened?



Make propagation harder

NetworkPolicy (1.7+)
 	
Microservices = natural boundaries
 	
Ingress: Only admin-portal ➡ payments API
 	
Egress: Need other services? Internet? No ➡ block it off
 	
Istio authz also an option for services
	kind: NetworkPolicy
	...
	  podSelector:
	  matchLabels:
	  app: "payment"
	  ingress:
	  - from:
	  - podSelector:
	  matchLabels:
	  app: "admin-portal"

Enforce authn/authz on kubelet (1.5+) Access to kubelet port ➡ execute inside any container.	See docs goo.gl/XumrAd GKE: enabled by default

Summary: Helping prevent attacks

Update: Keep up with K8s releases, enable RBAC

Minimal Containers: Small container OS, no root, no hostpath/network

Segregation: Namespaces, dedicated nodes, network policies

Get involved

- Great security engineer expertise at sig-auth
- Help us make future production of the world rock solid
- Meet Wednesdays every 2 weeks: goo.gl/7DzJJY

- Google Kubernetes/GKE security team is hiring in Seattle :)

Links

- GKE hardening 1.8 blogpost: goo.gl/88Nzbk
- Securing a cluster k8s doc: goo.gl/Qmhsw9
- Using RBAC: goo.gl/XkuEuU, RBAC on GKE: goo.gl/o1BkQf
- audit2rbac for semi-automated RBAC policy generation: goo.gl/d3W5h2
- Using namespaces to separate privileges: goo.gl/SHi3w1
- GKE master firewall: goo.gl/ZVRJzf
- PodSecurityPolicy: goo.gl/J5kmVL
- Anti-affinity: goo.gl/BzYbFk, taints/tolerations: goo.gl/HTQcBf
- Node authorizer: goo.gl/12J2U2
- Kubelet client cert rotation: goo.gl/yQ3rP7
- Network policy: goo.gl/1cjtgx (also see ahmetb's talk: goo.gl/PdLwE6)
- Kubelet authn/z: goo.gl/XumrAd

- Security features roadmap: see Jordan Liggitt's Sig Auth Update talk
- Sig-auth meeting: goo.gl/7DzJJY

- Metasploit (used in demos) is available under a BSD license: github.com/rapid7/metasploit-framework