RSAConference™2023

San Francisco | April 24 – 27 | Moscone Center

SESSION ID: CSCO-W08

Attacking and Detecting Attacks on Kubernetes Clusters

#RSAC

Jay Beale

CEO
InGuardians, Inc.
@jaybeale @inguardians

Alana Trimble

Security Analyst
InGuardians, Inc.
@L1ttl3Red

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference™ or any other co-sponsors. RSA Conference does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

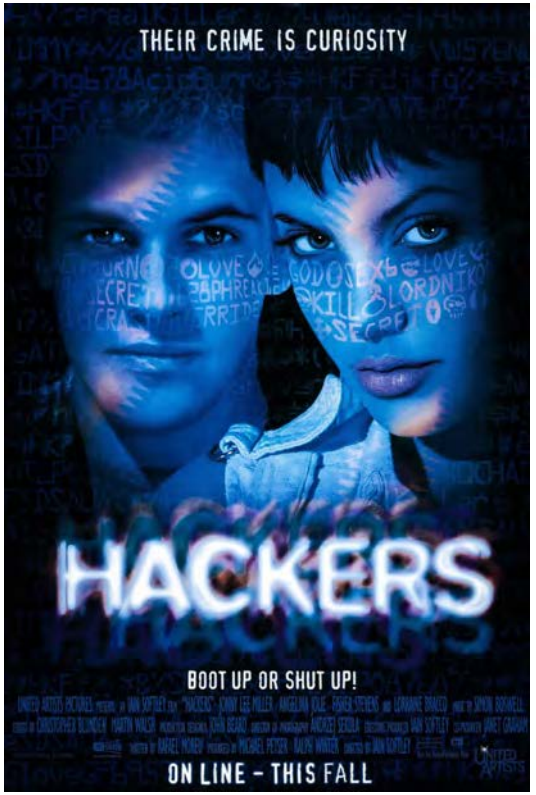
Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

Agenda

- Introducing the DEF CON Kubernetes Capture the Flag (CTF)
- DEMO: The Start-to-Finish Kubernetes CTF Solution
- Defending Against Each Attack Path Step
- Apply: Defending Your Cluster

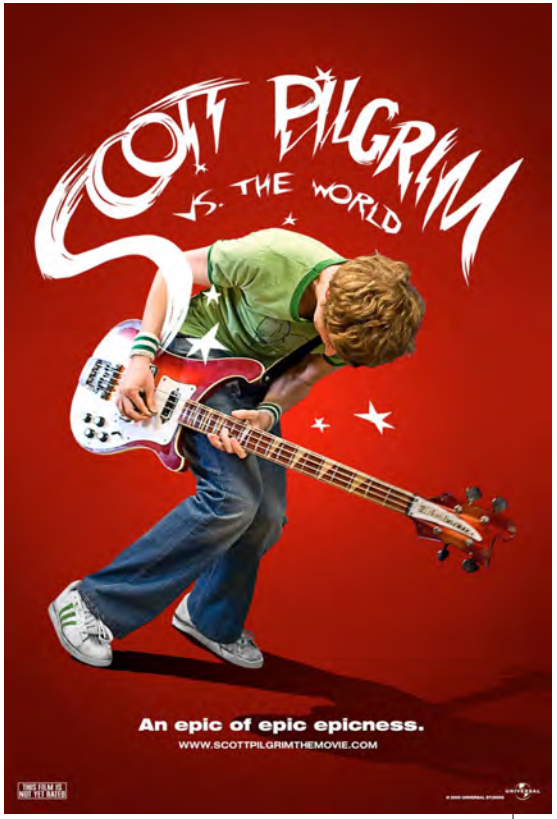
DEF CON Kubernetes Capture the Flag History

- We held the first contest on New Years Eve on December 31, 2020, as part of a special DEF CON NYE 2021.
- We created a Kubernetes Capture the Flag for DEF CON to:
  - give more experienced Kubernetes attackers a chance to compete
  - let newer folks learn on a complex CTF.
  - bring more researcher attention to Kubernetes in the DEF CON community.



The DEF CON Kubernetes Capture the Flag (CTF)

- DEF CON 30's Kubernetes CTF was themed on the movie "Scott Pilgrim vs the World."
- In the DEF CON event, teams competed and gained experience with a difficult challenge.
- Each year, we also host a non-competitive event using the exact same CTF scenario, but giving everyone an "answer key" like what you're seeing today.
- Watch for the next opportunity via this Twitter handle: @ctfsecurity



The DEF CON Kubernetes Capture the Flag (CTF)

The DEF CON Kubernetes CTF Solution

- Now, we'll demonstrate one solution to the Kubernetes CTF.
- Participants were given an IP address of a Kubernetes node, dedicated to their team.
- This attack starts outside a cluster, gets remote code execution in one pod running in the cluster, then moves laterally until it eventually compromises a cluster node before getting the last flag.
- We'll explain what we're doing at each step in the demonstration.

Welcome to the DEF CON 30 Kubernetes Capture the Flag!

This CTF is themed on Scott Pilgrim vs the World. You play as the movie's protagonists, including Scott Pilgrim, Ramona Flowers, and Knives Chau. You must defeat the League of Evil Exes!

Remember, enumeration is key!

RSAConference™2023

Defending Against Each Attack Path Step

Reference slides are included to summarize the attack step being addressed

Reference: Attack Path Review: Flag 1

- Portscan to discover an HTTP service.
- Connect to it with curl or a browser - it announces that it is Matthew Patel, Ramona's first evil ex-boyfriend.
- Use dirb to find the /matthew path.
- Use wfuzz to find the right username-password combination: scott / pilgrim.
- Find a flag in /flag/flag.txt.
- Start up a reverse shell.

Next, you need to gain two secrets from a Vault instance in the next namespace. The secrets are named "flag" and "next".

Defense: Network Policies

- Prevent the reverse shell by using network policies (Kubernetes-native firewall rules) to block outgoing connections.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-egress
  namespace: matthew-patel
spec:
  podSelector: {}
  policyTypes:
  - egress
  eggress:
  - []
```

Defense: Web App Firewall-enabled Ingress

- Break the automated scanning and reduce the odds of the RCE by using a web application firewall, like a modsecurity-enabled ingress-nginx.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-waf
  annotations:
    nginx.ingress.kubernetes.io/enable-modsecurity: \"true\"
    nginx.ingress.kubernetes.io/enable-owasp-modsecurity-crs: \"true\"
    nginx.ingress.kubernetes.io/modsecurity-snippet: |
      SecRuleEngine On
      SecRequestBodyAccess On
      SecAuditEngine RelevantOnly
      SecAuditLogParts ABIJDEFHZ
      SecAuditLogType Serial
      SecAuditLog /dev/stdout
spec:
  ingressClassName: nginx
  rules:
  - host: matthew-patel-webservice
    http:
      paths:
      - backend:
          service:
            name: matthew-patel
            port:
              number: 80
        path: /
        pathType: Prefix
```

Reference: Attack Path Review: Flag 2

- Discover matthew-patel's service account token in the first pod.
- Determine that the next namespace will be named "lucas-lee".
- Ask Kubernetes what matthew-patel's service account can do.
- List pods in the lucas-lee namespace, then exec into the lucas-lee-vault-0 pod.
- Ask Vault for the flag and next secrets.

You must trick Todd Ingram into drinking half-and-half. If you offer him food or drink, he checks with the service at envy.todd-ingram.svc.cluster.local:80

Defense: Audit RBAC


```
$ kubectl -n lucas-lee get rolebindings -o wide
NAME                           ROLE                   AGE    USERS   GROUPS   SERVICEACCOUNTS
lucas-lee-can-list-secrets     Role/list-secrets      249d                    lucas-lee/lucas-lee-vault
matthew-patel-execs-into-vault Role/pod-exec-vault    249d                    matthew-patel/matthew-patel
matthew-patel-gets-lists-pods  Role/get-list-pods     249d                    matthew-patel/matthew-patel
```

- Audit RBAC to ensure that service account tokens are not over-privileged.

```
$ kubectl -n lucas-lee describe role pod-exec-vault
Name:         pod-exec-vault
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names       Verbs
  ---------  -----------------  --------------       -----
  pods/exec  []                 [lucas-lee-vault-0]  [create]
```

Defense: Deactivate Token Mounts

- Consider deactivating the automatic mounting of service account tokens into pods.

```
apiVersion: v1
kind: Pod
metadata:
  name: lucas-lee-vault-0
  namespace: lucas-lee
spec:
  automountServiceAccountToken: false
  ...
```

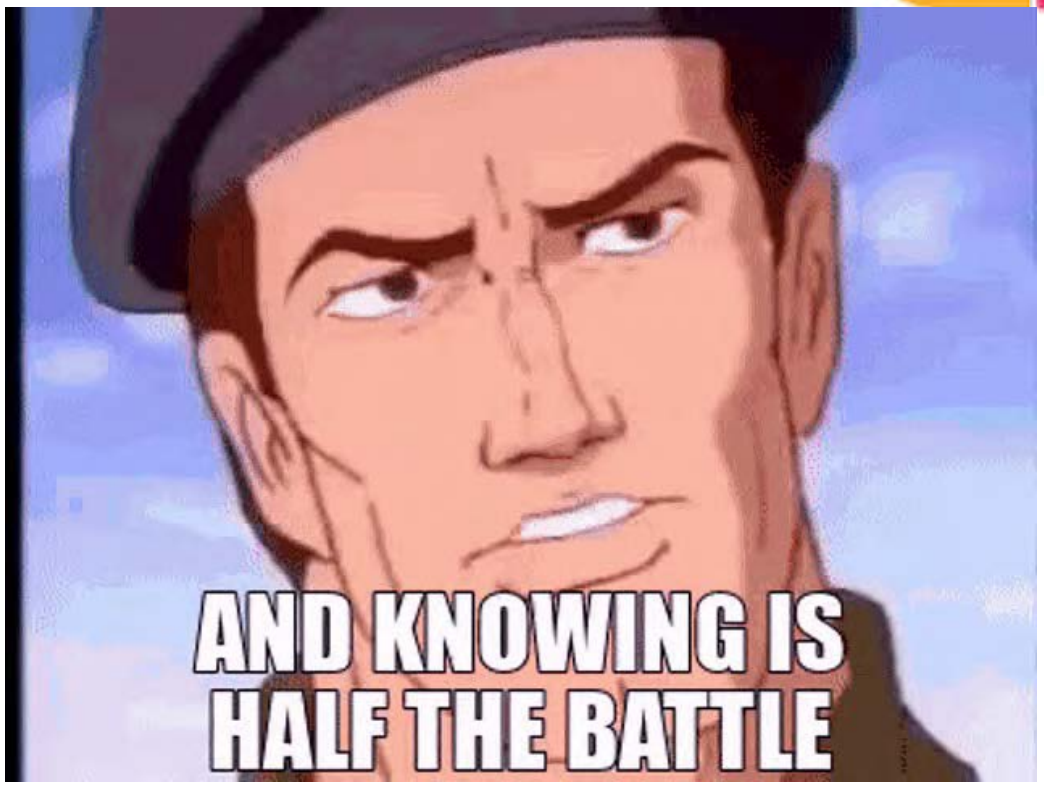
Reference: Attack Path Review: Flag 3

- lucas-lee's service account cannot do anything in the todd-ingram namespace.
- Ask Kubernetes what lucas-lee's service account can do in lucas-lee's namespace.
- Find you cannot request a secret's contents, but you can list secrets, which gives you their contents.
- Get a flag from a secret that tells you to get the stunt team.
- Get the stunt-team service account token, stored as a secret.

You must trick Todd Ingram into drinking half-and-half. If you offer him food or drink, he checks with the service at envy.todd-ingram.svc.cluster.local:80

Defense: Knowing is Half the Battle

- Ensure all platform operators understand that listing secrets discloses those secrets.



Defense: Admission Controllers

- Consider using an out-of-tree admission controller like Kyverno or OPA Gatekeeper, which can prevent creation of RBAC roles that can get/list secrets.

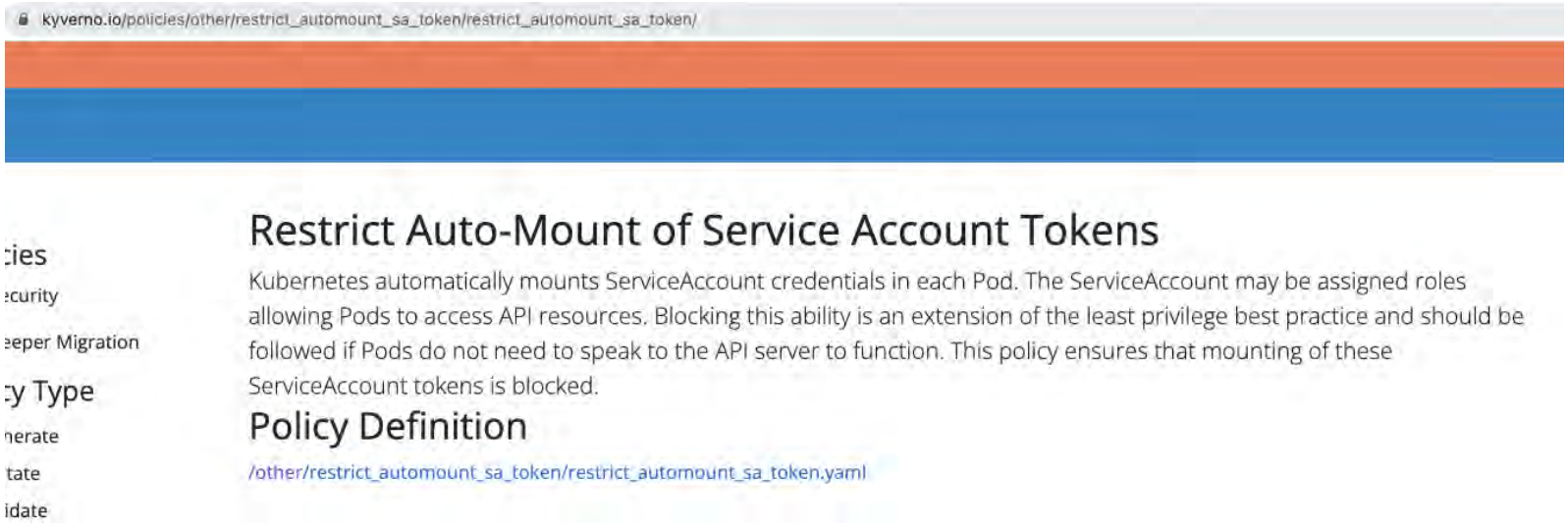
Reference: Attack Path Review: Flag 4

- Ask what stunt-team's service account can do in todd-ingram namespace.
- It read and list pods, and exec into the "spilledcoffee" pod.
- exec into the spillcoffee pod
- find a flag that says, "incorrigible-dont-know-the-meaning-of-the-word"
- find this pod's service account: julie

You must trick Todd Ingram into drinking half-and-half. If you offer him food or drink, he checks with the service at envy.todd-ingram.svc.cluster.local:80

Defense

- Audit RBAC.
- Restrict the automatic mounting of service account tokens into pods (with exceptions) using an admission controller like Kyverno.



Reference: Attack Path Review: Flag 5 – Part 1

- Ask what julie's service account can do in todd-ingram namespace.
- It list, read, and modify services.
- Remember: a service is a load balancer.
- The "envy" service sends traffic to pods whose "app" label is set to "isitvegan".
- We can modify the service so it sends traffic to pods with label "spilledcoffee", like our "spilledcoffee" pod.

You must trick Todd Ingram into drinking half-and-half. If you offer him food or drink, he checks with the service at envy.todd-ingram.svc.cluster.local:80

Reference: Attack Path Review: Flag 5 – Part 2

- Find todd's pod accepts /consume/<food>.
- Todd's pod asks the envy service.
- Investigate the envy pod and service.
- Modify the envy service to send the request to our spilledcoffee pod.
- Use netcat to answer Todd's /isitvegan HTTP request with "yes"
- Send todd /consume/half-and-half.
- Receive: todd's svc acct token and flag.

You must trick Todd Ingram into drinking half-and-half. If you offer him food or drink, he checks with the service at envy.todd-ingram.svc.cluster.local:80

Defense

- Harden RBAC: ensure that principals cannot modify services and avoid cross-namespace privileges.
- Consider deactivating the automatic mounting of service account tokens.
- Consider service meshes (e.g. Istio, Linkerd, Consul, AWS App Mesh, ...) which offer mutual TLS between pods.

Reference: Attack Path Review: Flag 6

- Remember that todd's requests to the envy service were originally going to the envy pod.
- Research nodeJS object serialization.
- Find there is a long-standing RCE vulnerability in the most popular nodejs serialization library.
- Exploit the envy pod's service to get a shell.
- Get a flag and a service account token.

"ERROR: need a Base64-encoded NodeJS-serialized object containing a name..."

Defense

- Detect vulnerable libraries with image scanning and supply chain protections (e.g., SBOM).
- For image scanning, consider open source tools like clair or grype, or use a SaaS solution from your cloud provider, image registry provider, or third party commercial service.

Reference: Attack Path Review: Flag 7

- Find that the envy-adams service account token can get and list configmaps in the roxy-richter namespace.
- List configmaps.
- Read the roxy-richter configmap.
- Get a flag and an item called next, which tells you to exec into a specific pod in the katayanagi-twins namespace.

Next, find a configuration item named roxy-richter-flag to defeat roxy-richter!

Defense

- Ensure that sensitive configuration data is kept in secrets, rather than configmaps.
- Audit RBAC to ensure that service account tokens are not over-privileged.
- Consider deactivating the automatic mounting of service account tokens into pods.

Reference: Attack Path Review: Flag 8

- Find that the envy-adams service account token can get and list pods in the katayanagi-twins namespace.
- Find it can also exec into a specific pod, named "ninth-circle".
- exec into the "ninth-circle" pod.
- Find a flag.

exec-into-ninth-circle-in-katayanagi-twins-namespace

Defense

- Audit RBAC to ensure that service account tokens are not over-privileged.
- Consider deactivating the automatic mounting of service account tokens into pods.

Reference: Attack Path Review: Flag 9

- Realize that the node's directory /etc/kubernetes/manifests is mounted into the ninth-circle pod as /manifests.
- Create a privileged "static pod" by writing a pod manifest into /manifests.
- Mount the node's filesystem to get the kubelet credentials.
- Use the kubelets credentials to find and exec into a gideon-graves-chaos-theater pod in the gideon-graves namespace.
- Read the flag secret from the filesystem.

no hint…

Defense

- Consider file integrity monitoring for sensitive node directories, including /etc/kubernetes/manifests.
- Use Pod Security Standards or another admission controller to prevent hostPath mounts of node directories.

RSAConference™2023

Detection

Activate Kubernetes Audit Logs

- We can detect a number of the vital steps on this attack path with Kubernetes' built-in auditing log.
- To turn it on in the most simple way, we do three things:
  - Write an **audit log configuration file** to the control plane node(s) filesystem.
  - Add **command line flags** to the API server activate use the config file. (*)
  - Mount the **configuration file** and **log directory** into the API server pod. (*)

* If kube-api-server is run as a pod, edit its manifest file in /etc/kubernetes/manifests/.

Write an Audit Log File to the Control Plane Nodes

audit/audit-policy.yaml

```yaml
apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      # Resource "pods" doesn't match requests to any subresource of pods,
      # which is consistent with the RBAC policy.
      resources: ["pods"]
  # Log "pods/log", "pods/status" at Metadata level
  - level: Metadata
```

https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/audit/audit-policy.yaml

Modify API Server Command Line Flags


```
  GNU nano 6.2          /etc/kubernetes/manifests/kube-apiserver.manifest
apiVersion: v1
kind: Pod
metadata:
  annotations:
    dns.alpha.kubernetes.io/internal: api.internal.k8s-1233-322.k8s.local
    kubectl.kubernetes.io/default-container: kube-apiserver
  creationTimestamp: null
  labels:
    k8s-app: kube-apiserver
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - args:
    - --log-file=/var/log/kube-apiserver.log
    - --also-stdout
    - /usr/local/bin/kube-apiserver
    - --allow-privileged=true
    - --anonymous-auth=false
    - --api-audiences=kubernetes.svc.default
    - --apiserver-count=1
    - --audit-policy-file=/etc/kubernetes/audit-policy.yaml
    - --audit-log-path=/var/log/kubernetes/audit/audit.log
```

Create Volumes: Config File and Log Directory


```
GNU nano 6.2                    /etc/kubernetes/manifests/kube-apiserver.manifest *
  name: etcopenssl
- hostPath:
    path: /etc/kubernetes/in-tree-cloud.config
  name: cloudconfig
- hostPath:
    path: /srv/kubernetes/ca.crt
  name: kubernetesca
- hostPath:
    path: /srv/kubernetes/kube-apiserver
  name: srvkapi
- hostPath:
    path: /srv/sshproxy
  name: srvsshproxy
- hostPath:
    path: /etc/kubernetes/kube-apiserver-healthcheck/secrets
    type: Directory
  name: healthcheck-secrets
- name: audit
  hostPath:
    path: /etc/kubernetes/audit-policy.yaml
    type: File
- name: audit-log
  hostPath:
  path: /var/log/kubernetes/audit/
  type: DirectoryOrCreate
```

Mount Config File and Log Directory into Pod


```
GNU nano 6.2                    /etc/kubernetes/manifests/kube-apiserver.manifest
  - mountPath: /usr/lib/ssl
    name: usrlibssl
    readOnly: true
  - mountPath: /usr/local/openssl
    name: usrlocalopenssl
    readOnly: true
  - mountPath: /var/ssl
    name: varssl
    readOnly: true
  - mountPath: /etc/openssl
    name: etcopenssl
    readOnly: true
  - mountPath: /etc/kubernetes/in-tree-cloud.config
    name: cloudconfig
    readOnly: true
  - mountPath: /srv/kubernetes/ca.crt
    name: kubernetesca
    readOnly: true
  - mountPath: /srv/kubernetes/kube-apiserver
    name: srvkapi
    readOnly: true
  - mountPath: /srv/sshproxy
    name: srvsshproxy
    readOnly: true
  - mountPath: /etc/kubernetes/audit-policy.yaml
    name: audit
    readOnly: true
  - mountPath: /var/log/kubernetes/audit/
    name: audit-log
    readOnly: false
```

Detecting the Attacks

Which events result in audit logs that tell us an attack is underway?

Attack Path Action	API Event
exec –it pod /bin/bash	create pod/exec
get secrets -o yaml	list secrets
auth can-i --list	create selfsubjectrulesreviews
edit service envy	update service
get configmap roxy	get configmap
(create a pod)	create pod

Event: Exec Into a Pod


```
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Request",
  "auditID": "7c16c609-d17e-4d03-8059-e36f8d31b1a2",
  "stage": "RequestReceived",
  "requestURI": "/api/v1/namespaces/lucas-lee/pods/lucas-lee-vault-0/exec?command=%2Fbin%2Fsh&container=vault&stderr=true&stdin=true&stdou
e",
  "verb": "create",
  "user": {
    "username": "system:serviceaccount:matthew-patel:matthew-patel",
    "uid": "e7b0c8cc-5d1f-4bbe-ab8d-6a1bd91e1d1f",
    "groups": [
      "system:serviceaccounts",
      "system:serviceaccounts:matthew-patel",
      "system:authenticated"
    ],
    "extra": {
      "authentication.kubernetes.io/pod-name": [
        "matthew-patel-microservice"
      ],
      "authentication.kubernetes.io/pod-uid": [
        "4e358f0b-fd7e-431b-a77e-0cdf3a240f57"
      ]
    }
  },
  "sourceIPs": [
    "10.0.16.2"
  ],
  "userAgent": "kubectl/v1.24.0 (linux/amd64) kubernetes/4ce5a89",
  "objectRef": {
    "resource": "pods",
    "namespace": "lucas-lee",
    "name": "lucas-lee-vault-0",
    "apiVersion": "v1",
    "subresource": "exec"
  },
  "requestReceivedTimestamp": "2023-04-26T08:03:27.440372Z",
  "stageTimestamp": "2023-04-26T08:03:27.440372Z"
}
```

Event: Exec Into a Pod


```
$ cat exec-audit-log.json | jq 'select(.objectRef.resource == "pods") | select(.objectRef.subresource == "exec") | { request : .requestURI , us
er: .user.username , timestamp: .requestReceivedTimestamp }'
{
  "request": "/api/v1/namespaces/lucas-lee/pods/lucas-lee-vault-0/exec?command=%2Fbin%2Fsh&container=vault&stderr=true&stdin=true&stdout=true",
  "user": "system:serviceaccount:matthew-patel:matthew-patel",
  "timestamp": "2023-04-26T08:03:27.440372Z"
}
```

Lessons Learned

- Overarching Lesson:
  - Small configuration changes make all the difference!
- Next week you should:
  - Audit the RBAC authorizations for service accounts and users.
  - Ensure that your clusters use Pod Security Standards or an out-of-tree admission controller.
- In the next three months, you should:
  - Deploy an out-of-tree admission controller, like Kyverno or OPA Gatekeeper.
- Longer term:
  - Review the other defenses covered here.

References

- Kyverno Policy for Disabling Service Account Token Mounting

`https://kyverno.io/policies/other/restrict_automount_sa_token/restrict_automount_sa_token/`

- Service Mesh Comparison

`https://www.toptal.com/kubernetes/service-mesh-comparison`

- Image Scanning with Quay's Clair and Anchore's Grype (Open Source)

`https://github.com/quay/clair/`

`https://github.com/anchore/grype`

- Activating Auditing in Kubernetes

`https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/`

RSAConference™2023

Thank You.

Please follow us on Twitter:

@jaybeale
@L1ttl3Red
@inguardians
@ctfsecurity

Learn more about Kubernetes security:

https://inguardians.com/kubernetes