KubeCon CloudNativeCon
North America 2025

No Joke…
Two Security Maintainers Walk into a Cluster

Jackie Maertens and Nilekh Chaudhari, Microsoft

Cluster Inspectors

Jackie Maertens
Senior Software Engineer
Istio Security Maintainer
Co-Lead of Istio's PSWG

Nilekh Chaudhari
Senior Software Engineer
Azure Kubernetes Service (AKS)
Istio Contributor

Security Goals

- Operate with principle of least privilege
- Protect the host
- Isolate environments
- Protect sensitive information
- Minimize open doors

Inspection Checklist

- ❑ Role Based Access Control (RBAC)
- ❑ Encryption at Rest
- ❑ Secret Management
- ❑ Network Policy
- ❑ Pod Security Standards
- ❑ Image Vulnerability


SECURITY
KubeCon | CloudNativeCon
North America 2025

Role Based Access Control (RBAC)

Overview

- Controls *who* can access *what* in your Kubernetes cluster
- Roles are assigned to users, service accounts, and groups
- Follows the *principle of least privilege*
- Critical security mechanism, first line of defense
- Four key components: Roles, RoleBindings, ClusterRoles, ClusterRoleBindings

Role Based Access Control (RBAC)

Common Pitfalls

- **Binding cluster-admin to service accounts**
  - Complete cluster compromise from single pod breach
- **Using wildcard permissions (*)**
  - Grants access to all resources
- **Forgetting to remove test permissions**
  - Temporary debugging access becomes permanent
- **Using ClusterRolesBindings instead of RolesBindings**
  - Breaks namespace isolation, increases blast radius
- **Not disabling auto-mount of service account tokens**
  - Every pod gets credentials attackers can steal

bash

RBAC: The Mistake 💥

ℹ️ A developer creates a ServiceAccount for testing...
ℹ️ They bind it to cluster-admin for 'convenience'

→ examples

Encryption at Rest

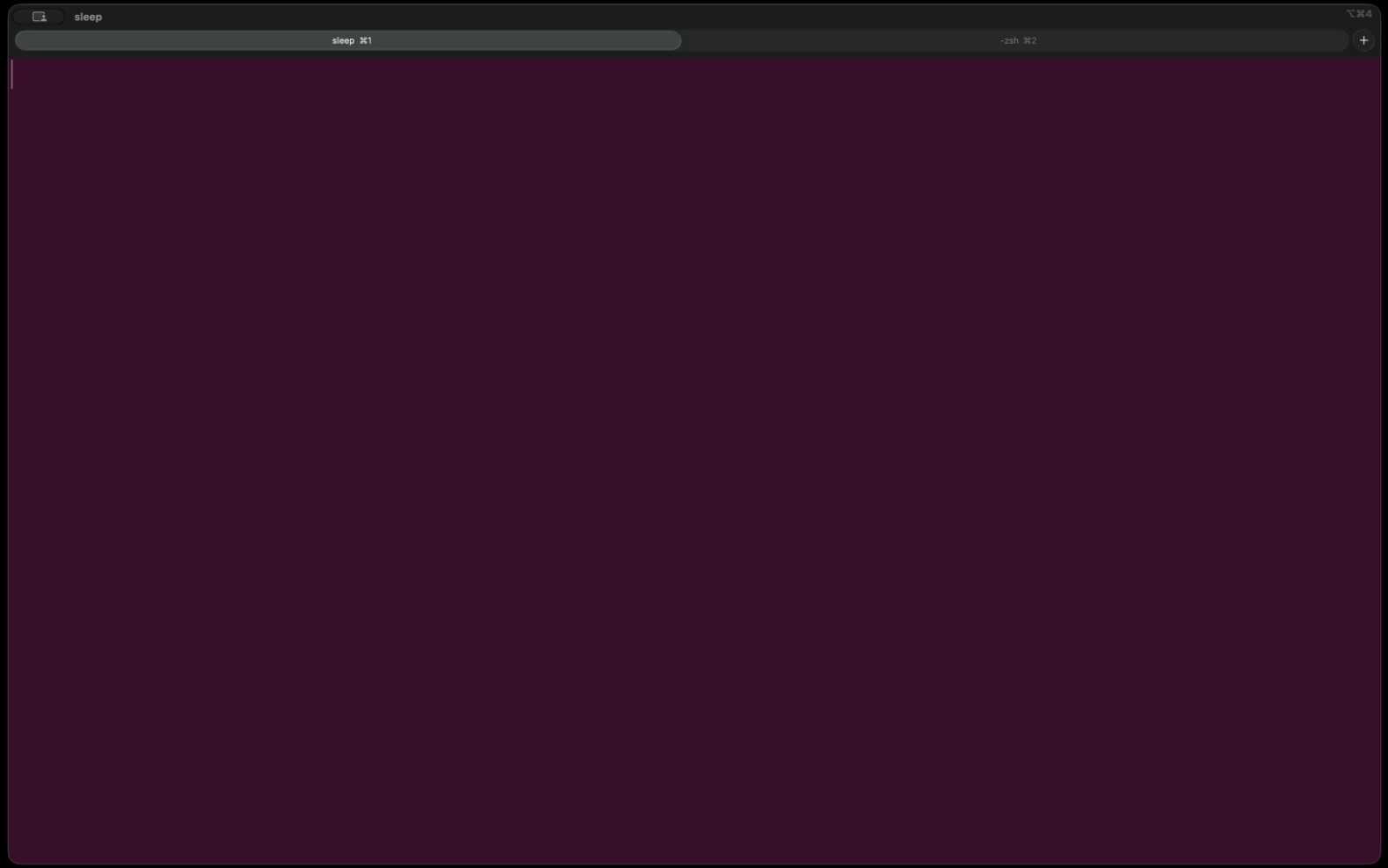
Overview

- By default, Kubernetes stores Secrets in plain text in etcd
- Anyone with etcd access can read ALL your credentials
- Encryption at rest protects data before writing to etcd
- KMS (v2) uses external key management (AWS, Azure, GCP)
- Required for compliance

Encryption at Rest

Common Pitfalls

- **Forgetting to re-encrypt existing secrets**
  - Old secrets remain in plain text after enabling encryption
- **Losing the encryption key**
  - Lost key = permanent data loss for ALL secrets encrypted with that key
- **Using "identity" provider alone**
  - Provides zero encryption despite being in config
- **Using local keys in production**
  - Doesn't protect against node compromise
- **Not testing encryption**


sleep
⌥⌘4
sleep ⌘1
-zsh ⌘2

Secret Management

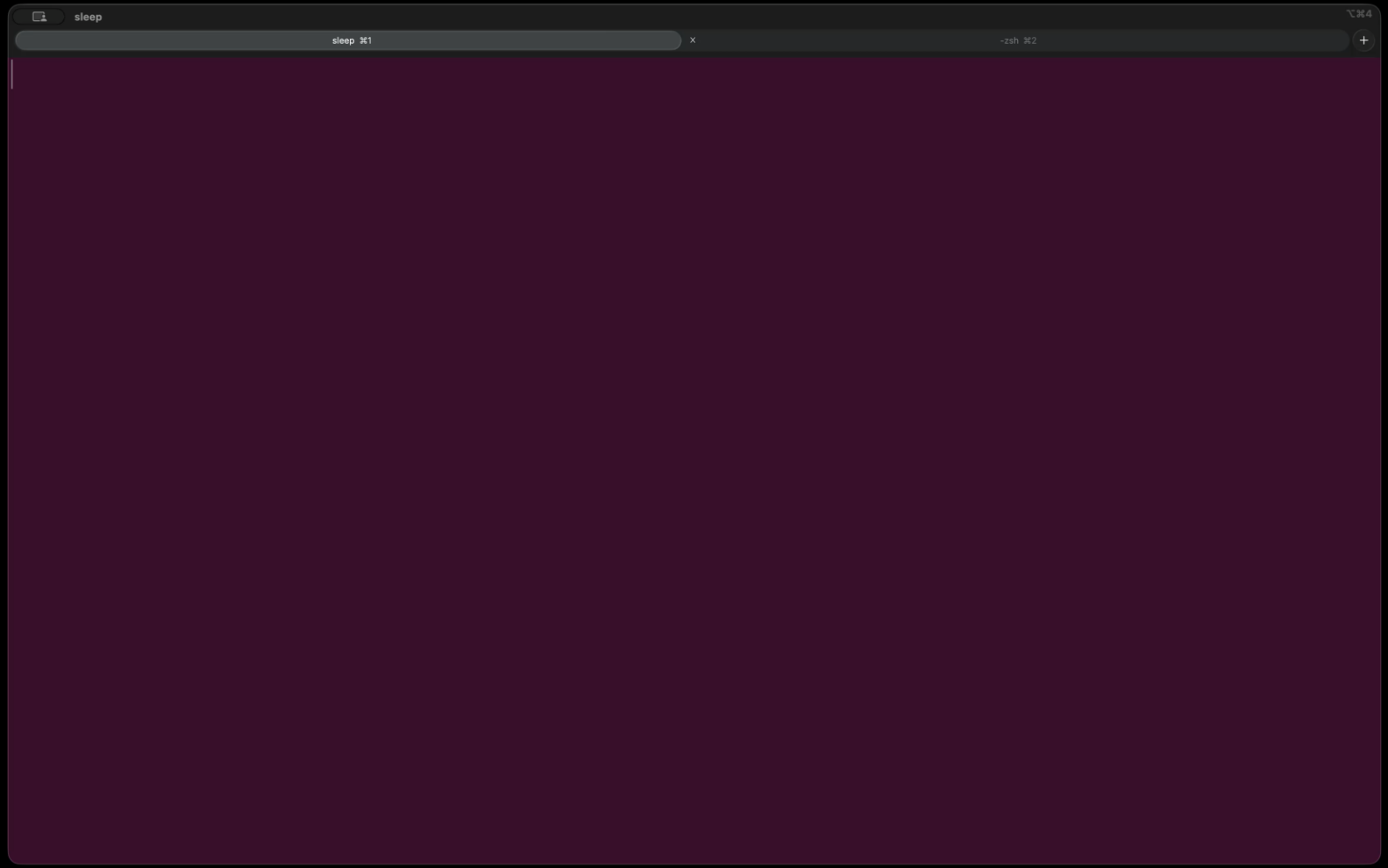
Overview

- Native K8s Secrets exist in etcd, even with encryption
- Secrets Store CSI Driver retrieves secrets from external secrets stores
- Secrets are never stored in etcd, they’re made available to pods only at creation time
- Supports Vault, AWS Secrets Manager, Azure Key Vault, GCP Secret Manager and many more
- Advanced features: automatic rotation, audit logs

Secret Management

Common Pitfalls

- **Mixing native Secrets with CSI driver**
  - Inconsistent security posture
- **Using environment variables for secrets**
  - Leaks through logs and crash dumps
- **Not enabling secret rotation**
  - Stale credentials remain valid indefinitely


sleep
⌥⌘4
sleep ⌘1
-zsh ⌘2

Checkpoint

- [x] Role Based Access Control (RBAC)
- [x] Encryption at rest
- [x] Secret Management
- [ ] Network Policy
- [ ] Pod Security Standards
- [ ] Image Vulnerability

NetworkPolicy

Overview

- Define and control how pods communicate with other entities (pods or external services)
- By defa ress and egress
- Fine-gr

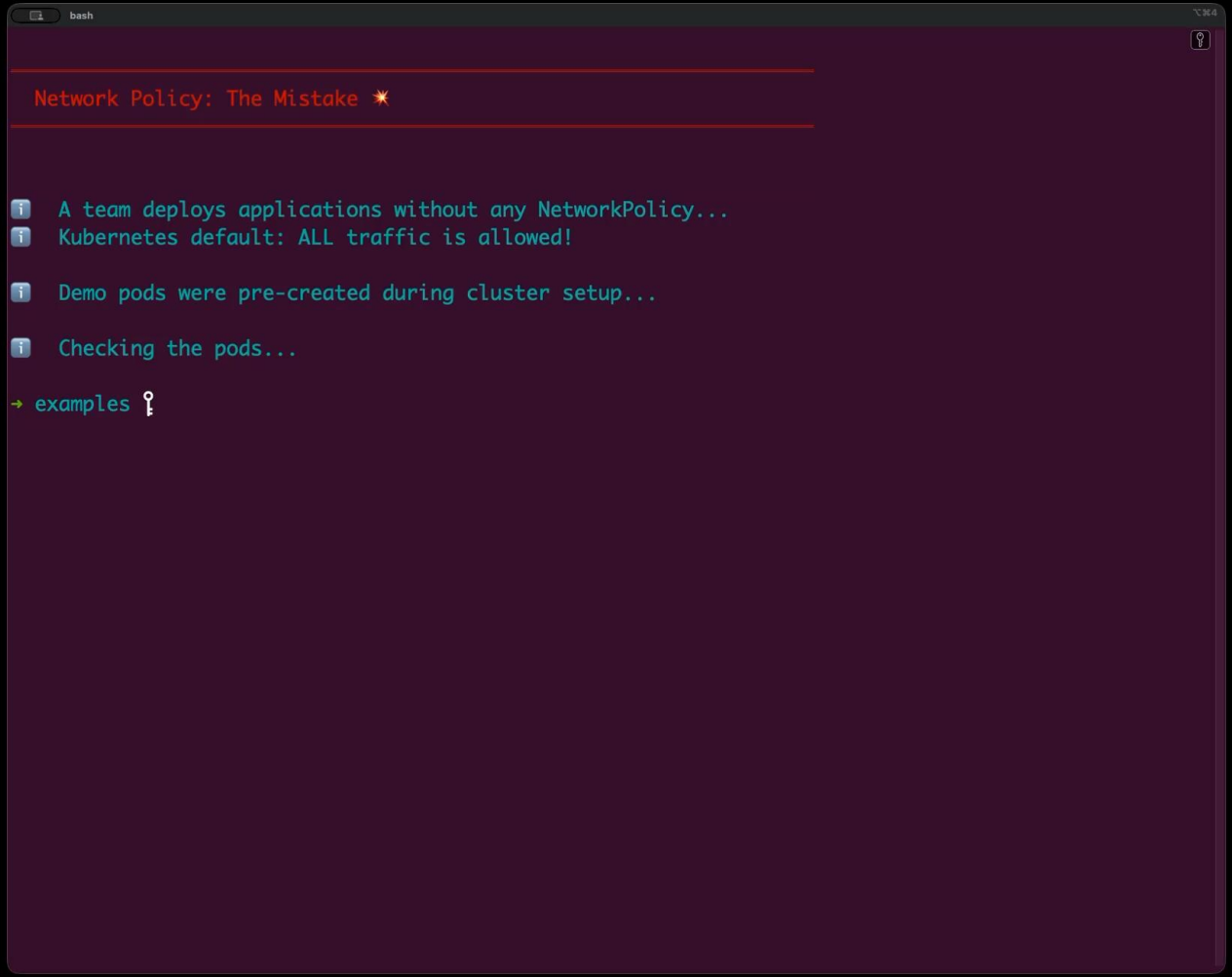
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: example-network-policy
spec:
  podSelector: # Selects pods the policy applies to
    matchLabels:
      role: db
  policyTypes: # Define the types of traffic to be controlled
  - Ingress
  - Egress
  ingress:
    # Define allowed ingress traffic
  egress:
    # Define allowed egress traffic
```

```yaml
ingress:
- from:
  - namespaceSelector:
      matchLabels:
        project: myProject
egress:
- to:
  - ipBlock:
      cidr: 10.0.0.0/24
```

NetworkPolicy

Common Pitfalls

- Misunderstanding defaults
- Forgetting egress rules
- Misconfiguring additive policies
- Using IPs instead of selectors
- Not testing policy


bash
⌥⌘4
Network Policy: The Mistake
A team deploys applications without any NetworkPolicy...
Kubernetes default: ALL traffic is allowed!
Demo pods were pre-created during cluster setup...
Checking the pods...
→ examples

Pod Security Standards

Overview

- Define security profiles for pods to control privileged capabilities and configurations
  - Privileged
  - Baseline
  - Restricted

Pod Security Standards

Capabilities/Settings to Restrict

- Namespace sharing
- Privileged escalation
- Linux capabilities
- Volume types
- Host port access
- Proc mounts
- Running as non root
- And more…

Pod Security Standards

Common Pitfalls

- Allowing overly permissive controls
- Not enforcing pod security standard

```yaml
apiVersion: v1
kind: Pod
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: my-baseline-namespace
  labels:
    pod-security.kubernetes.io/enforce: baseline # Required to enforce Baseline profile
    pod-security.kubernetes.io/enforce-version: v1.34
    pod-security.kubernetes.io/audit: restricted # Adds an audit annotation to Restricted profile violations
    pod-security.kubernetes.io/audit-version: v1.34
    pod-security.kubernetes.io/warn: restricted # Surfaces a user-facing warning on Restricted profile violations
    pod-security.kubernetes.io/warn-version: v1.34
```

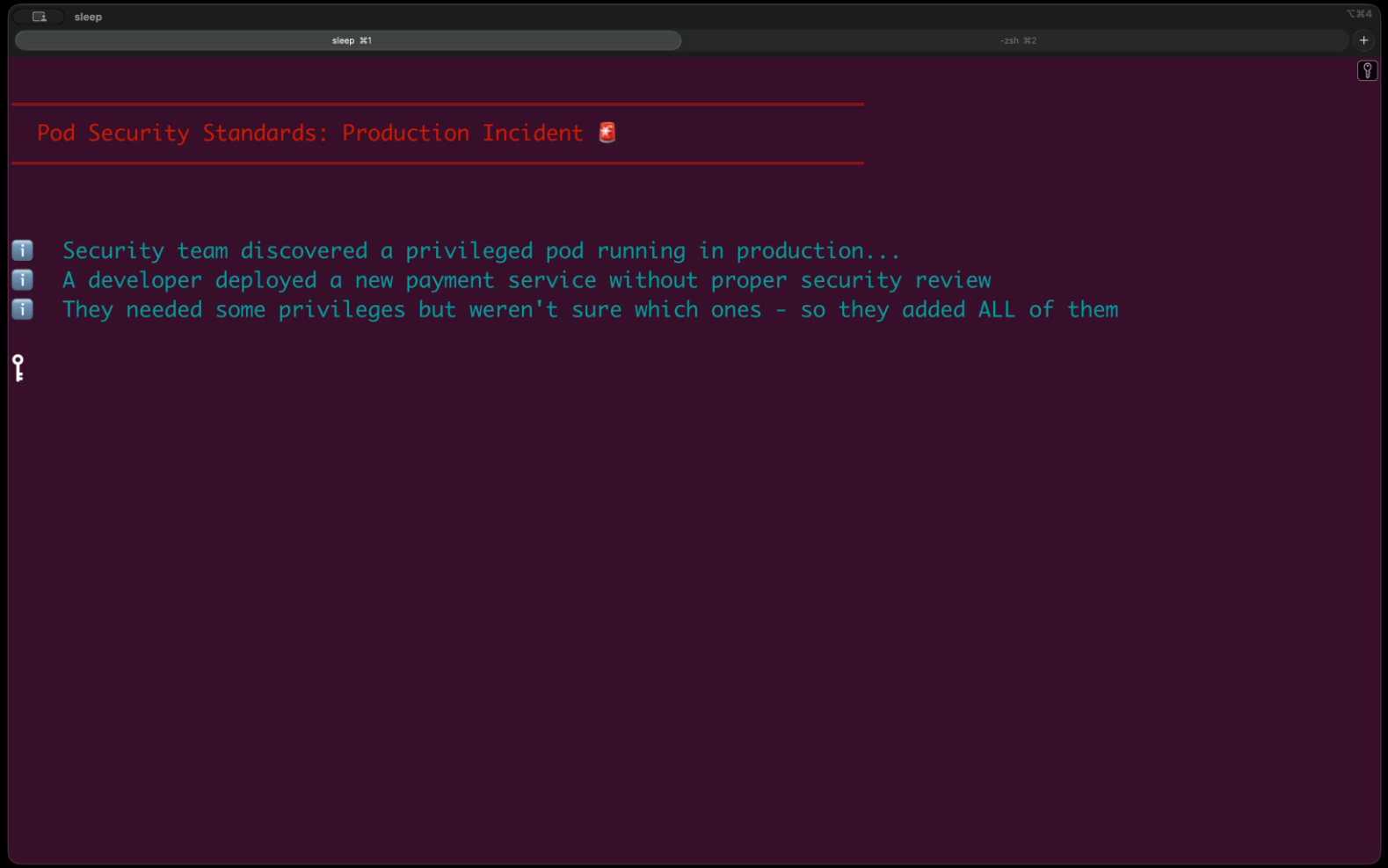

sleep
sleep ⌘1
-zsh ⌘2
Pod Security Standards: Production Incident
Security team discovered a privileged pod running in production...
A developer deployed a new payment service without proper security review
They needed some privileges but weren't sure which ones - so they added ALL of them

Image Vulnerability

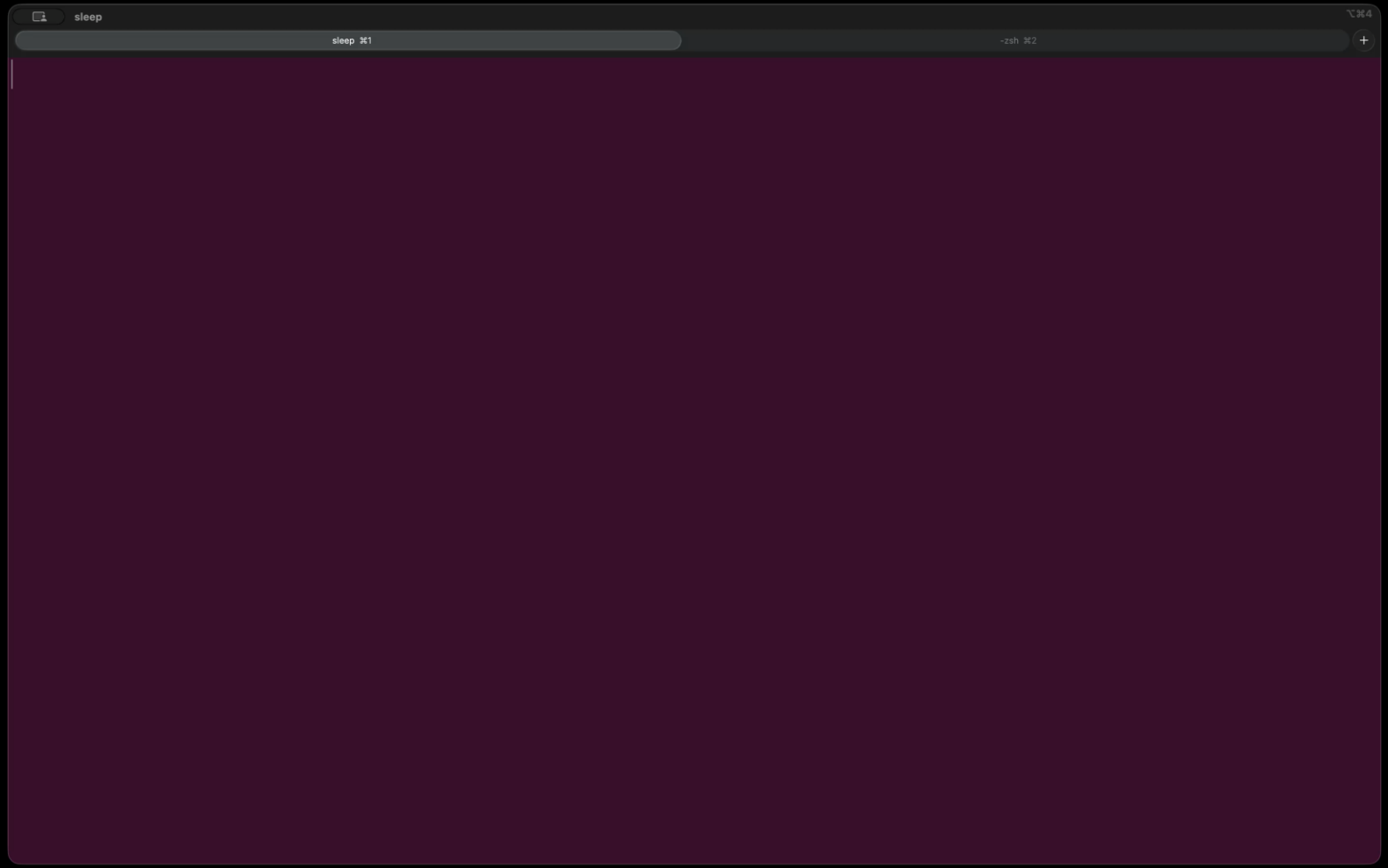
Overview

- Images provide portable, immutable, and reproducible application packages
- Improper image utilization and maintenance makes users vulnerable to attacks

Image Vulnerability

Common Pitfalls

- Utilizing outdated and unsupported images
- Referencing images by tag not digest
- Not utilizing vulnerability scanning tools
  - Trivy, Grype, and more


sleep
⌥⌘4
sleep ⌘1
-zsh ⌘2

Inspection Review

- [x] Role Based Access Control (RBAC)
- [x] Encryption at Rest
- [x] Secret Management
- [x] Network Policy
- [x] Pod Security Standards
- [x] Image Vulnerability

What Next?

KubeCon CloudNativeCon North America 2025

- Check out our guide on GitHub at https://aka.ms/security-at-kubecon
- Review the Kubernetes security checklist
- Integrate security checklists and safeguards at each lifecycle stage
- Lookout for new projects and guidelines!
- Please take a moment to provide feedback

nilekhc / jaellio

KubeCon | CloudNativeCon

North America 2025

Thank you!