Guia de Boas Práticas para Segurança da API do Kubernetes

Baseado no Trabalho de Conclusão de Curso (TCC) "Análise da Superfície de Ataque da API do Kubernetes" – FATEC São Caetano do Sul – 2024

Autores: Arthur Nolasco, Ian Guimarães, Michele Bueno, Rafael Nascimento

1. Introdução

Este guia apresenta um conjunto de boas práticas para a proteção da API do Kubernetes, com ênfase em ambientes críticos e sensíveis, como os utilizados em cidades inteligentes. As recomendações aqui descritas foram validadas por meio de experimentação em ambiente controlado com o Minikube, permitindo demonstrar, na prática, a eficácia das medidas de segurança adotadas.

2. Autenticação e Autorização

A autenticação e a autorização são pilares fundamentais para proteger o acesso à API do Kubernetes. A seguir, estão descritas as práticas recomendadas para garantir que apenas usuários e serviços autorizados possam interagir com o cluster.

2.1 Autenticação Forte

Utilize mecanismos robustos de autenticação para validar identidades antes de qualquer acesso à API:

- **Tokens de acesso protegidos** vinculados a *ServiceAccounts*.

- **Certificados digitais válidos**, preferencialmente com suporte a **TLS 1.3**, garantindo a integridade e confidencialidade das conexões.

2.2 Implementação de RBAC (*Role-Based Access Control*)

Controle o que cada entidade autenticada pode fazer dentro do cluster por meio do RBAC, respeitando sempre o princípio do menor privilégio.

Exemplo de *Role* restrita:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-viewer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list"]
```

Exemplo de *RoleBinding* associando uma *ServiceAccount*:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: ServiceAccount
  name: my-service-account
roleRef:
  kind: Role
  name: pod-viewer
  apiGroup: rbac.authorization.k8s.io
```

2.3 Boas Práticas

- **Aplique o princípio do menor privilégio**: conceda apenas as permissões necessárias.
- **Revogue acessos não utilizados** regularmente.
- **Realize auditorias** com o comando:

```
kubectl auth can-i --as=system:serviceaccount:<namespace>:<account>
```

3. Proteção de *Secrets*

Os ***Secrets*** armazenam dados sensíveis, como senhas, tokens e chaves de acesso. A manipulação incorreta desses recursos pode expor o cluster a riscos graves. A seguir, são apresentadas as práticas recomendadas para protegê-los adequadamente.

3.1 Criação Segura de Secrets

Crie *Secrets* de forma controlada e evite armazenar dados sensíveis diretamente em arquivos ou comandos históricos:

```
kubectl create secret generic meu-secret \
  --from-literal=usuario=admin \
  --from-literal=senha=123456
```

3.2 Boas Práticas de Armazenamento e Acesso

- **Monte os *secrets* como volumes nos *pods***, em vez de usá-los como variáveis de ambiente, o que reduz o risco de vazamento.
- **Habilite a criptografia de dados no etcd**, configurando um `EncryptionConfiguration` para garantir que os Secrets estejam criptografados em repouso.
- **Restrinja o acesso a *secrets* utilizando RBAC**, permitindo leitura apenas aos serviços estritamente necessários.
- **Evite expor dados sensíveis em logs, arquivos YAML versionados ou imagens de *container***.

4. Proteção TLS e HTTPS

A segurança das comunicações entre os componentes do *cluster* e os clientes externos deve ser garantida por meio de conexões criptografadas utilizando protocolos atualizados e certificados válidos.

4.1 Exigir Conexões com TLS 1.3

Configure o servidor da API para aceitar apenas conexões com **TLS 1.3**, evitando versões obsoletas e vulneráveis do protocolo:

```
--tls-min-version=VersionTLS13 \
--tls-cipher-suites=TLS_AES_256_GCM_SHA384,...
```

4.2 Utilização de Certificados Válidos

Certificados digitais são fundamentais para garantir a identidade do servidor. Utilize certificados emitidos por uma autoridade confiável (como *Let's Encrypt*) ou certificados auto assinados, quando apropriado:

```
openssl req -x509 -nodes -days 365 \
  -newkey rsa:4096 \
  -keyout tls.key \
  -out tls.crt \
  -subj "/CN=example.com"
```

4.3 Validação da Conexão Segura

Verifique se a conexão com a API está utilizando corretamente o TLS 1.3:

```
openssl s_client -connect <IP>:6443 -tls1_3
```

4.4 Boas Práticas Adicionais

- Configure **renovação automática** dos certificados (por exemplo, com `certmanager`).

- Monitore a **data de expiração dos certificados** para evitar indisponibilidade.
- Evite o uso de **certificados fracos**, como RSA menores que 2048 bits.

5. Controle de Acesso à API

A API do Kubernetes é o principal ponto de administração do *cluster*. Portanto, seu acesso deve ser rigidamente controlado para reduzir a superfície de ataque e prevenir acessos não autorizados.

5.1 Restringir Exposição Externa

- Utilize **VPNs**, ***firewalls*** e redes privadas para garantir que somente usuários autorizados, dentro de uma rede segura, possam acessar a API.
- **Bloqueie o tráfego externo** diretamente direcionado ao `kube-apiserver`.

5.2 Não Expor Diretamente o Kube-APIserver

- O `kube-apiserver` nunca deve estar acessível diretamente pela internet.
- Prefira o uso de **jump boxes (bastion hosts)** ou túneis SSH para administração remota, quando necessário.

5.3 Utilizar Ingress com TLS

- Ao disponibilizar serviços expostos via API, utilize ***Ingress Controllers*** com suporte a **TLS**, para garantir comunicação segura.
- O *Ingress* atua como um ponto centralizado de controle, facilitando a aplicação de políticas de segurança como autenticação, limitação de IP e auditoria.

5.4 Reforço de Segurança Adicional

- Aplique **restrições por endereço IP** no nível de firewall ou proxy reverso.
- Implemente **rate limiting** para evitar abusos por força bruta ou DoS.

6. Ferramentas e Auditoria

A adoção de ferramentas específicas para monitoramento e auditoria é fundamental para garantir a detecção proativa de falhas de segurança e o cumprimento de políticas de conformidade no ambiente Kubernetes.

6.1 Falco

Ferramenta de **detecção de intrusão em tempo real** que monitora atividades anômalas em containers e no host. Permite configurar regras para alertar sobre comportamentos suspeitos, como acesso a arquivos sensíveis, execução de comandos fora do padrão, ou modificação de configurações críticas.

6.2 Kubeaudit

Utilitário que realiza **auditorias automatizadas de segurança** na configuração do cluster Kubernetes. Detecta práticas inseguras como permissões excessivas em RBAC, containers executando como root, ou ausência de políticas de segurança.

6.3 cert-manager + ClusterIssuer

O `cert-manager` automatiza a emissão, renovação e gerenciamento de **certificados TLS** no cluster. Combinado com um `ClusterIssuer`, permite a integração com autoridades

certificadoras públicas ou privadas, garantindo criptografia de ponta a ponta para APIs e serviços expostos.

6.4 Postman com Token de Autenticação

O uso do **Postman** com tokens de autenticação válidos permite realizar testes manuais e controlados nos *endpoints* da API do Kubernetes. Essa prática é útil para validar o funcionamento de mecanismos de autenticação e autorização, além de simular acessos conforme as políticas de RBAC configuradas.

7. Pontos de Atenção

Mesmo com a aplicação das boas práticas anteriores, alguns aspectos de segurança costumam ser negligenciados. Abaixo estão os principais pontos de atenção que merecem monitoramento contínuo:

7.1 Ausência de Autenticação Multifator (MFA)

A utilização de autenticação baseada apenas em senhas ou tokens expõe o ambiente a riscos em caso de comprometimento de credenciais. A implementação de **MFA** (autenticação multifator) adiciona uma camada adicional de segurança para o acesso ao painel administrativo ou a sistemas de controle do cluster.

7.2 Falta de Controle de Taxa (Rate Limiting)

Sem a implementação de ***rate limiting***, a API do Kubernetes pode ser alvo de ataques por força bruta ou negação de serviço (DoS). É recomendável configurar limites de requisições por IP, usuário ou token, especialmente em ambientes expostos.

7.3 Ausência de Controle por IP

Permitir o acesso irrestrito à API a partir de qualquer endereço IP representa um risco significativo. O ideal é restringir o acesso por meio de **listas de controle de acesso (ACLs)**, firewalls, ou proxies reversos que limitem os IPs permitidos a interagir com o cluster.

8. Conclusão

A adoção sistemática das boas práticas apresentadas neste guia contribui significativamente para a redução da superfície de ataque da API do Kubernetes, mitigando riscos de acesso não autorizado, vazamento de dados e comprometimento do ambiente.

No entanto, é importante destacar que **segurança não é um estado final, mas um processo contínuo**. À medida que o ambiente evolui e novas ameaças surgem, é fundamental revisar periodicamente as configurações, aplicar atualizações de segurança e manter uma postura proativa na proteção do cluster.

9. Acesso ao Guia

Este conteúdo está disponível para consulta e download no repositório oficial do projeto, onde também podem ser encontrados exemplos práticos, arquivos YAML e atualizações futuras do guia:

Repositório oficial:

https://github.com/CyberKube/Guia-de-Boas-Praticas-Kubernetes