
بسم الله الرحمن الرحيم

al=naafi

EDUCATION BENEFITS ALL

Certified Kubernetes Security Specialist (CKS)

The Certified Kubernetes Security Specialist (CKS) program provides assurance that a CKS has the skills, knowledge, and competence on a broad range of best practices for securing container-based applications and Kubernetes platforms during build, deployment and runtime.

Pre-requisite

- CKA certification is required to sit for this exam.

Who Is It For

- A Certified Kubernetes Security Specialist (CKS) is an accomplished Kubernetes practitioner (must be CKA certified) who has demonstrated competence on a broad range of best practices for securing container-based applications and Kubernetes platforms during build, deployment and runtime.

About This Certification

- CKS is a performance-based certification exam that tests candidates' knowledge of Kubernetes and cloud security in a simulated, real world environment. Candidates must have taken and passed the Certified Kubernetes Administrator (CKA) exam prior to attempting the CKS exam. CKS may be purchased but not scheduled until CKA certification has been achieved.
- CKA Certification must be active (non-expired) on the date the CKS exam (including Retakes) is scheduled.

What It Demonstrates

- Obtaining a CKS demonstrates a candidate possesses the requisite abilities to secure container-based applications and Kubernetes platforms during build, deployment and runtime, and is qualified to perform these tasks in a professional setting.

Securing Kubernetes Clusters (10)

- Welcome to this course on securing Kubernetes clusters! In this course, we'll delve into the essential security practices for safeguarding your Kubernetes environment.
- We'll cover cluster setup, hardening measures, and system hardening techniques to ensure your clusters are robust and resilient against potential threats.

Cluster Setup

- **Use Network Security Policies to restrict cluster level access:**
  - Limit traffic flow within the cluster.
  - Enhance security posture by controlling pod-to-pod communication.
- **Use CIS benchmark to review the security configuration of Kubernetes components:**
  - Leverage pre-defined security guidelines for various components (etcd, kubelet, kubedns, kubeapi).
  - Identify and address potential security misconfigurations.
- **Properly set up Ingress objects with security control:**
  - Securely expose services to external users.
  - Implement authentication and authorization mechanisms (e.g., basic auth, TLS) for controlled access.
- **Protect node metadata and endpoints:**
  - Mitigate risks associated with exposing node metadata and endpoints.
  - Restrict access using RBAC or Network Policies.
- **Minimize use of, and access to, GUI elements:**
  - Limit reliance on graphical user interfaces (GUIs) for cluster management.
  - Prioritize using secure command-line tools (kubectl) to reduce attack surface.
- **Verify platform binaries before deploying:**
  - Ensure the integrity and authenticity of downloaded binaries.
  - Utilize checksums or trusted repositories to prevent deploying malicious code.

Using Network Security Policies

- Network Security Policies (NPs) define how pods can communicate within the cluster.
- NPs offer granular control over traffic flow, enhancing security.
- We can restrict incoming traffic (ingress) and outgoing traffic (egress) using NPs.
- NPs can be configured to target specific pods, namespaces, or labels.

Utilizing the CIS Benchmark

- The CIS Kubernetes Benchmark is a set of security recommendations for various Kubernetes components.
- It provides guidance on hardening etcd, kubelet, kubedns, kubeapi, and other components.
- Utilizing the CIS benchmark helps identify potential security misconfigurations.
- Tools like kube-bench can be used to assess cluster configuration against the CIS benchmark.

Securing Ingress Objects

- Ingress objects expose services to external users.
- Implementing security controls on Ingress objects is crucial.
- We can utilize authentication mechanisms like basic auth or TLS certificates.
- Authorization mechanisms like RBAC can further restrict access to specific users or groups.

Cluster Hardening (15)

- **Restrict access to Kubernetes API:**
  - Limit access to the Kubernetes API server to authorized users and applications.
  - Utilize authentication and authorization mechanisms (e.g., RBAC, API tokens) to control access.
- **Use Role Based Access Controls (RBAC):**
  - Implement RBAC to grant least privilege to users and service accounts.
  - Define roles, rolebindings, and clusterroles to control access to resources.
- **Exercise caution with service accounts:**
  - Disable default service accounts to minimize potential attack vectors.
  - Grant minimal permissions to newly created service accounts.
- **Update Kubernetes frequently:**
  - Regularly apply security patches and updates to address known vulnerabilities.
  - Stay informed about security advisories and maintain a proactive approach to security.

Restricting Access to the Kubernetes API Server

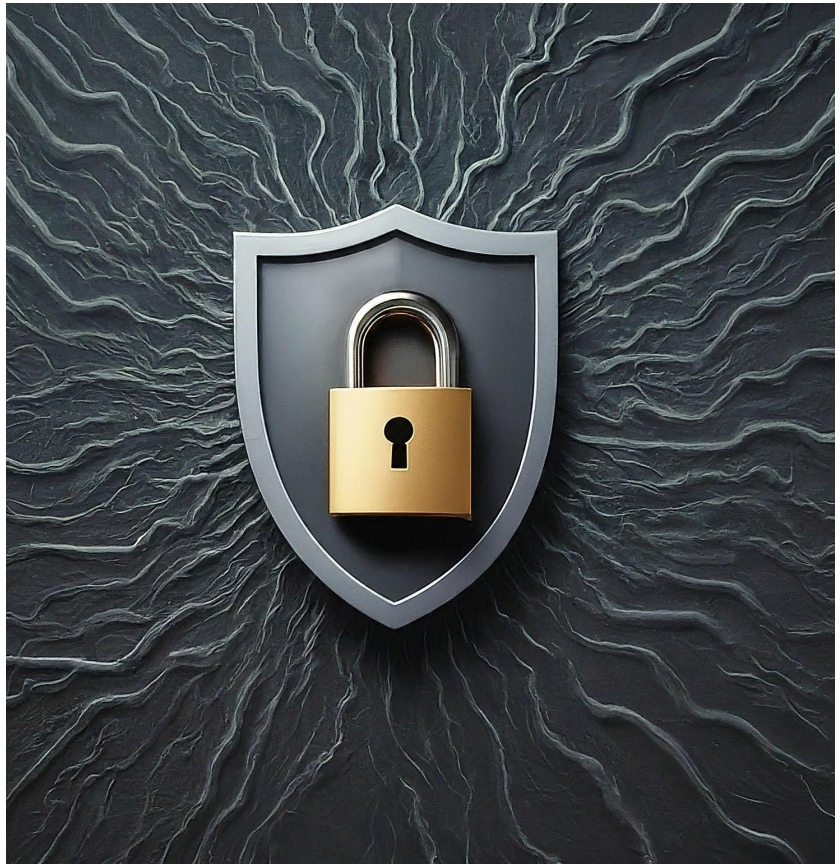
- The Kubernetes API server is the central point of communication for cluster management.
- Unrestricted access to the API server poses a significant security risk.
- Utilize authentication mechanisms (e.g., client certificates, tokens) to verify user identities.
- Implement authorization mechanisms (e.g., RBAC) to control access to API resources.

Role Based Access Controls (RBAC)

- RBAC is a security framework that grants access to resources based on roles.
- It enables defining roles (e.g., admin, editor, viewer) with specific permissions.
- Rolebindings associate users or service accounts with roles.
- Clusterroles and clusterrolebindings manage cluster-level access.

System Hardening (15)

- **Minimize host OS footprint:**
  - Reduce the attack surface by removing unnecessary packages and services from the underlying host OS.
  - Utilize minimal base images for containers and systems to minimize potential vulnerabilities.
- **Minimize IAM roles:**
  - Grant only the minimum IAM roles and permissions required for specific tasks.
  - Regularly review and update IAM policies to maintain appropriate access control.
- **Minimize external access to the network:**
  - Implement firewalls or security groups to restrict inbound and outbound network traffic.
  - Adhere to the "deny all, allow specific" approach for network access control.
- **Appropriately use kernel hardening tools:**
  - Leverage tools like AppArmor and seccomp to enforce security profiles and restrict system calls.
  - Consider consulting kernel hardening documentation for advanced configuration options.



Minimize Host OS Footprint

- **Reduce attack surface:**
  - Limit the number of installed packages and services to minimize potential vulnerabilities.
  - Remove unnecessary software components that are not essential for cluster operations.
- **Utilize minimal base images:**
  - Employ containers and systems built upon minimal base images.
  - These images contain only the essential components needed for functionality, reducing the attack surface.

Minimize IAM Roles

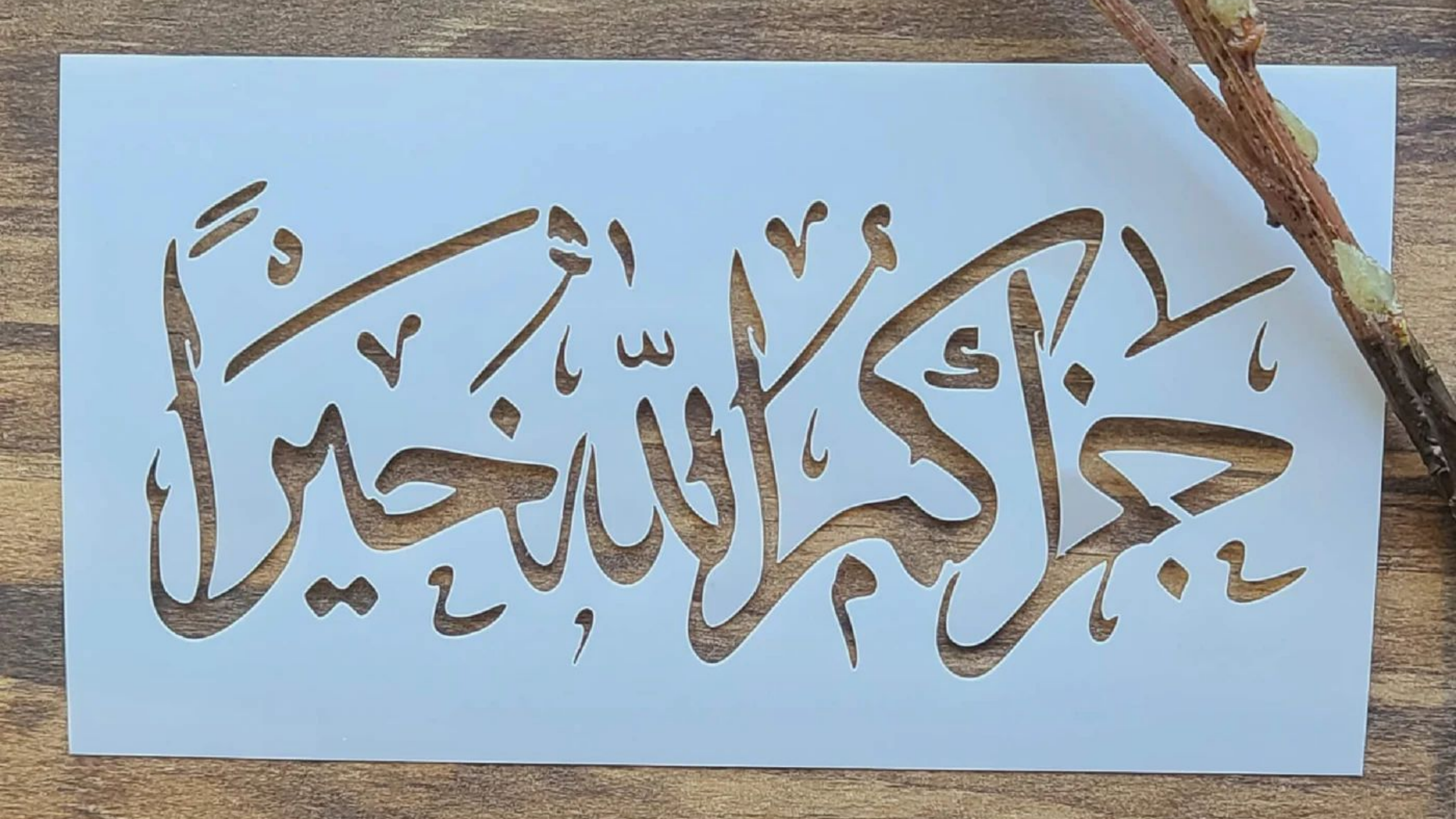
- **Grant least privilege:**
  - Assign only the minimum IAM roles and permissions required for specific tasks.
  - Avoid granting excessive permissions that could be exploited if compromised.
- **Regular review and update:**
  - Regularly review IAM policies to ensure they remain aligned with current needs.
  - Update policies promptly to revoke unnecessary permissions and address any changes in user roles or responsibilities.

Minimize External Access to the Network

- **Restrict network traffic:**
  - Implement firewalls or security groups to control inbound and outbound network traffic.
  - Define rules to allow only authorized communication and block all other traffic by default.
- **Deny all, allow specific:**
  - Adhere to the "deny all, allow specific" approach for network access control.
  - Start by denying all traffic and explicitly permit only the necessary communication channels.

Appropriately Use Kernel Hardening Tools

- **AppArmor:**
  - Provides application profiling and restricts system calls to enforce security policies.
  - Defines profiles that limit what applications can do on the system.
- **Seccomp:**
  - Allows fine-grained control over system calls allowed to processes.
  - Can significantly limit the attack surface and prevent unauthorized actions.


جزاكم الله خيرا