CKS Exam preparation

PodAnnotations of interest

KubeArmor

- `container.apparmor.security.beta.kubernetes.io/hello: localhost/k8s-apparmor-example-deny-write`: Load app armor profile. If not set, it doesn't work in the node. If set but deployed in a node with not that app armorprofile it fails with error `Error: failed to get container spec opts: failed to generate apparmor spec opts: apparmor profile not found k8s-apparmor-example-deny-write`

Config file strucure (examples)

Api Server

- Encryption at Rest: Used for specifying encryption configuration in the apiserver:

```
kind: EncryptionConfig
apiVersion: v1
resources:
  - resources:
      - secrets
    providers:
      - aescbc:
          keys:
            - name: key1
              secret: ${ENCRYPTION_KEY}
      - identity: {}
```

- Audit Logging

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: Metadata
```

- Admission control config file: ImagePullPolicyWebhook sets the image policy. Normally the kubeconfig file is set to use the kube-apiserver credentials in `/etc/kubernetes/pki`

TODO: Determine where can you restrict the docker repositories you can download from

```
apiVersion: apiserver.config.k8s.io/v1
kind: AdmissionConfiguration
plugins:
```

```
- name: ImagePolicyWebhook
  configuration:
    imagePolicy:
      kubeConfigFile: /etc/kubernetes/controlconf/webhook.kubeconfig
      allowTTL: 50
      denyTTL: 50
      retryBackoff: 500
      defaultAllow: true
```

Config files of interest path

seccomp

- `grep syscall /var/log/syslog`: Seccomp loging
- `/usr/include/asm/unistd_64.h`: Syscall ids

falco

- `/etc/falco/falco.yaml`: yaml with all the falco config. Amongst other config, you have there the rule files to apply

containerd

- `/etc/containerd/config.toml`: containerd config

ETCD

- `var/lib/etcd`: Normally where `etcd` storage is

Api Server

- `/var/lib/kubernetes/encryption-at-rest.yaml`: Normally where the encryption configuration for the apiserver is stored (it's determined by the parameter `--encryption-provider-config`)

Kubelet

- `/var/lib/kubelet/config.yaml`: Normally where the kubelet config is stored
- `/var/lib/kubelet/seccomp/`: Seccomp profiles

Config files options of interest

Kubelet

- In kubelet config (`/var/lib/kubelet/config.yaml`) `authentication.anonymous.enabled` controls authentication in `kubelet`
- In kubelet config (`/var/lib/kubelet/config.yaml`) `authorization.mode: Webhook|AlwaysAllow|AlwaysDeny` controls authorization in `kubelet`

Command options of interest

tracee

- `docker run --name tracee --rm --privileged --pid=host -v /lib/modules:/lib/modules:ro -v /usr/src:/usr/src:ro -v /tmp/tracee:/tmp/tracee aquasec/tracee:0.4.0 --trace container=new`

ETCD

- `--endpoints`: URL of etcd server
- `--insecure-skip-tls-verify`: Skip server cert validation
- `--insecure-transport=false`: Use TLS
- `ETCDCTL_API=3`: Variable to specify the API version to use
- `--cert`: Client certificate
- `--key`: Client key
- `--authorization-mode=RBAC|AlwaysAllow|AlwaysDeny|Node`: authorization mode

Containerd

- `containerd config default`

seccomp

Not implemented by kubernetes by default.

- `grep -i seccomp /boot/config-$(uname -r)`: check if seccomp is configured
- `grep Seccomp /proc/1/status`: Seccomp: 2 means that seccomp is implemented (0 disabled, 1 strict, 2 filtered)

  whitelist.json:

```
{
  "defaultAction": "SCMP_ACT_ERRNO",
  "archMap": [
    {
      "architecture": "SCMP_ARCH_X86_64",
      "subArchitectures": [
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
      ]
    },
    {
      "architecture": "SCMP_ARCH_AARCH64",
      "subArchitectures": [
        "SCMP_ARCH_ARM"
      ]
    }
  ],
  "syscalls": [
    {
      "names": [
```

```
          "read",
          "write",
          "exit",
          "exit_group"
        ],
        "action": "SCMP_ACT_ALLOW"
      }
    ]
  }
```

Api Server

- `--tls-cert-file`: Cert to enable TLS in API server
- `--tls-private-key-file`: Key of the TLS certificate for the API server
- `--token-auth-file`: Path to csv to enable token based authentication
- `--client-ca-file`: Path to CA certificate to authenticate client requests (mTLS)
- `--encryption-provider-config`: Path to encryption provider config to encrypt data before writting to ETCD
- `--audit-policy-file`: Path where the policy file config is stored
- `--audit-log-path`
- `--enable-admission-plugins`: Enable kube-apiserver admission-plugins
- `--admission-control-config-file`: Path to admission control config file

`kubectl`

- `--token`: Pass token to aexamplesuthenticate to API server

Certificates

- `openssl genrsa -out key.key 2048`: Generate 2048 bits RSA key
- `openssl req -new -req -key key.key -subj="/CN=COMMON-NAME -out certificate.csr`: Generate certificate signing request
- `openssl x509 -req -in ca.csr -signkey ca.key -CAcreateserial -out ca.crt .days 1000`: Self-sign certificate signing request
- Creating configuration for CSR:

```
cat > some.cnf <<EOF
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
IP.1 = 123.123.123.123
IP.2 = 127.0.0.1
EOF
```

- `openssl req -new -key key.key -subj "/CN=some" .out some.csr -config some.cnf`: Generate CSR with config file
- `openssl x509 -req -in some.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out some.crt -days 100`: Sign certificate with CA
- `openssl x509 -req -in some.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out some.crt -days 100 -extfile some.cnf -extensions v3_req`: Sign certificate with CA and config file
- `openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ingress.key -out ingress.crt -subj="/CN=example.internal/O=security"`: Create self-signed certificate for an ingress

ETCD

- `etcdctl put key value`: Write key value to etcd
- `etcdctl get key`: Get value from etcd
- `etcdctl del key`: Remove key from etcd

AppArmor

- `aa-genprof script.sh`:
  1. Create a security profile for the script, you need to run it once to do so
  2. Profile stored in `/etc/apparmor.d/root.apparmor.script.sh`
  3. Only affects tp the script you marked `script.sh`, if I copy its content to another, this other is not affected by this apparmour policy.
- `aa-status`: App armor status. Shows the profiles loaded
- `ln -s /etc/apparmor.d/root.apparmor.myscript.sh /etc/apparmor.d/disable/ & apparmor_parser -R /etc/apparmor.d/root.apparmor.myscript.sh`: Disable apparmor profile
- `apparmor_parser -q <<EOF`: Create an apparmor profile and loads it directly to the kernel
- `systemctl reload apparmor`
- You can unload the profile using `apparmor_parser` by specifying the `-R` option. This removes the profile from the in-kernel policy.

  Explanation of a profile

```
# Last Modified: Wed Jun 26 07:52:24 2024
abi <abi/3.0>,

include <tunables/global>
```

```
/root/apparmor/myscript.sh {
  include <abstractions/base>
  include <abstractions/bash>
  include <abstractions/consoles>
  include <abstractions/user-tmp>

  /root/apparmor/myscript.sh r,
  /usr/bin/bash ix,
  /usr/bin/rm mrix,
  /usr/bin/touch mrix,

}
```

Breakdown of the Profile

Header Information

```
# Last Modified: Wed Jun 26 07:52:24 2024
abi <abi/3.0>,
include <tunables/global>
```

 # Last Modified: Indicates the date and time when the profile was last modified.

abi `<abi/3.0>`: Specifies the application binary interface (ABI) version. This ensures compatibility with the version of AppArmor's ABI.

include `<tunables/global>`: Includes global tunables, which define variables used across all AppArmor profiles. Typically, these include system-specific paths or settings.

Profile Definition for the Script

```
/root/apparmor/myscript.sh {
```

/root/apparmor/myscript.sh: Specifies the profile applies to the script at this path.

Include Abstractions

```
include <abstractions/base>
include <abstractions/bash>
include <abstractions/consoles>
include <abstractions/user-tmp>
```

These lines include common policy abstractions, which are reusable sets of rules:

`<abstractions/base>`: General security policies for many basic system functions.

`<abstractions/bash>`: Policies specific to the Bash shell, which might include common commands and interactions.

`<abstractions/consoles>`: Rules for interactions with console devices.

`<abstractions/user-tmp>`: Rules for using temporary files in user space. File and Capability Permissions

```
/root/apparmor/myscript.sh r,
/usr/bin/bash ix,
/usr/bin/rm mrix,
/usr/bin/touch mrix,
```

These lines specify what actions are allowed for the script and other programs it might interact with:

/root/apparmor/myscript.sh r,: The script can be read (r) by itself. /usr/bin/bash ix,: The script can execute (x) Bash with inheritance (i), meaning that the permissions of the script apply to the subprocesses started by Bash.

/usr/bin/rm mrix,: The script can execute (x) the rm command with permissions to read (r), write (w), and modify (m) files.

/usr/bin/touch mrix,: The script can execute (x) the touch command with permissions to read (r), write (w), and modify (m) files.

NOTE: m Permission in AppArmor

The m stands for "memory map" and includes capabilities related to memory management and manipulation, such as:

Logging

- `journalctl -e -u kube-apiserver|etcd|kubelet`: See events in `journalctl` filtering by process and starting from the latest ones

Notes

- `kubeadm` creates certificates for `kube-apiserver` valid for 10.96.0.1 and internal IP addresses by default