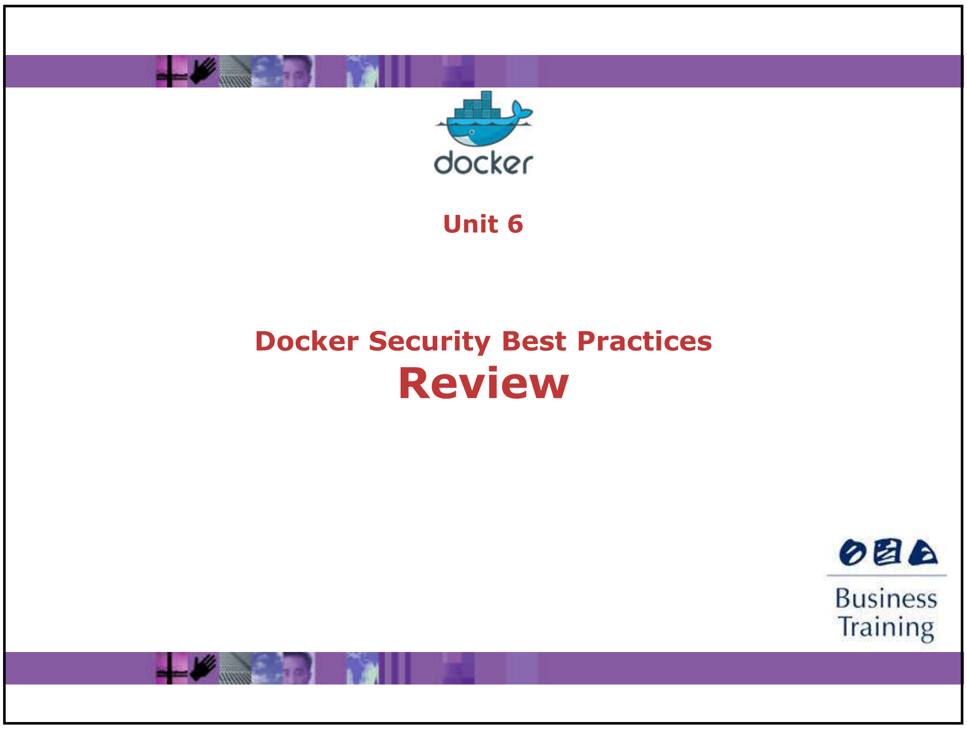


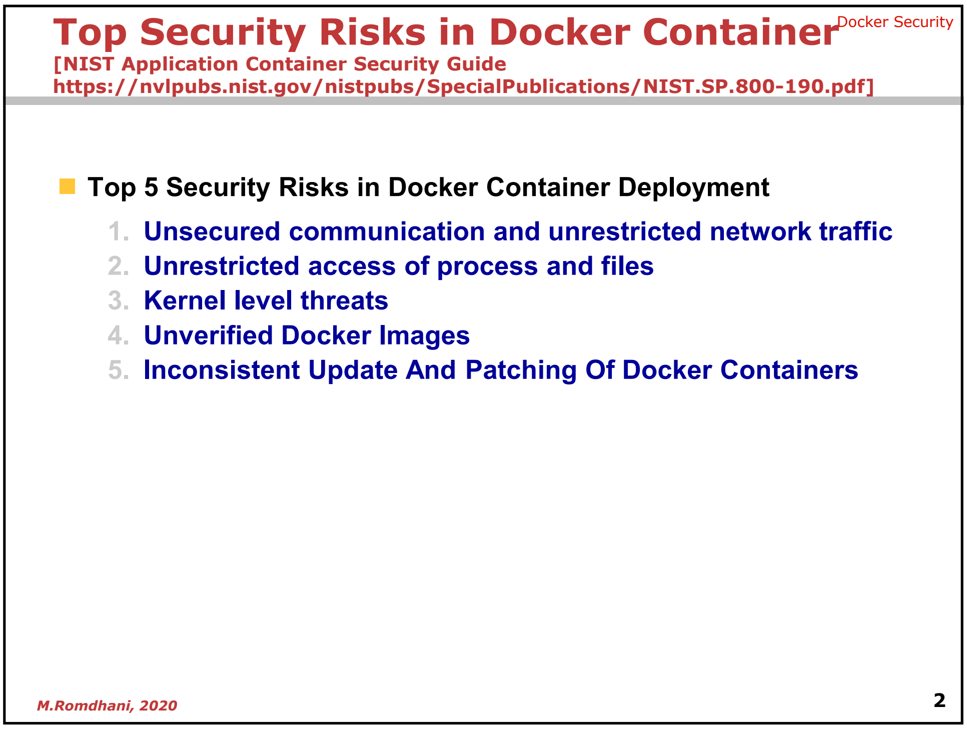
Unit 6

Docker Security Best Practices

Review

Business Training

1



Docker Security

Top Security Risks in Docker Container

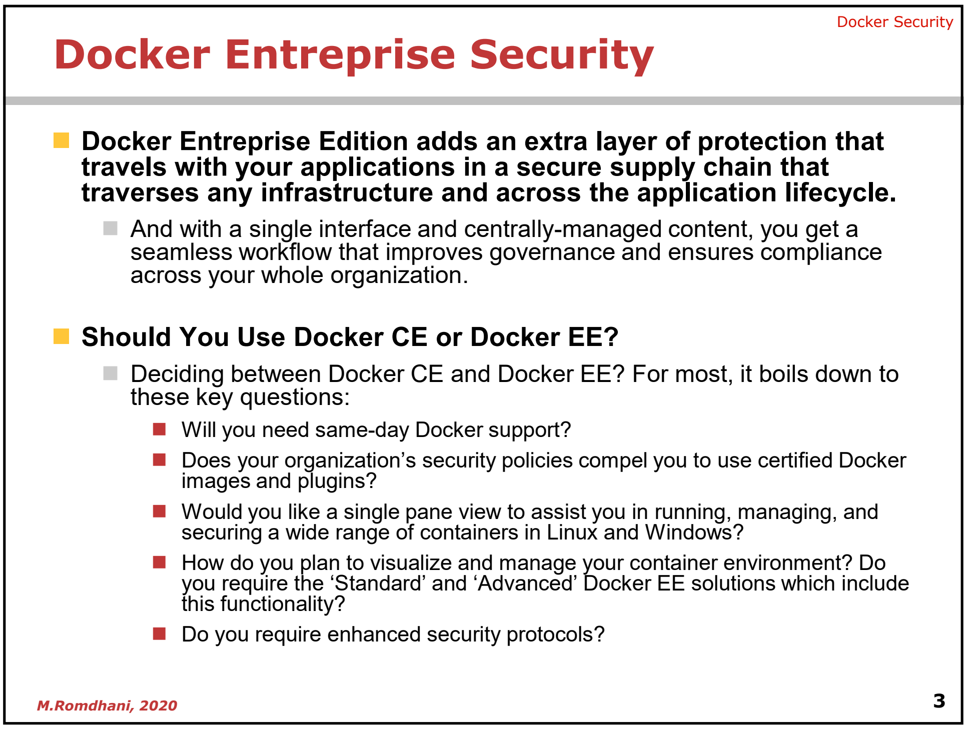
[NIST Application Container Security Guide
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf]

- Top 5 Security Risks in Docker Container Deployment
  1. Unsecured communication and unrestricted network traffic
  2. Unrestricted access of process and files
  3. Kernel level threats
  4. Unverified Docker Images
  5. Inconsistent Update And Patching Of Docker Containers

M.Romdhani, 2020

2

2

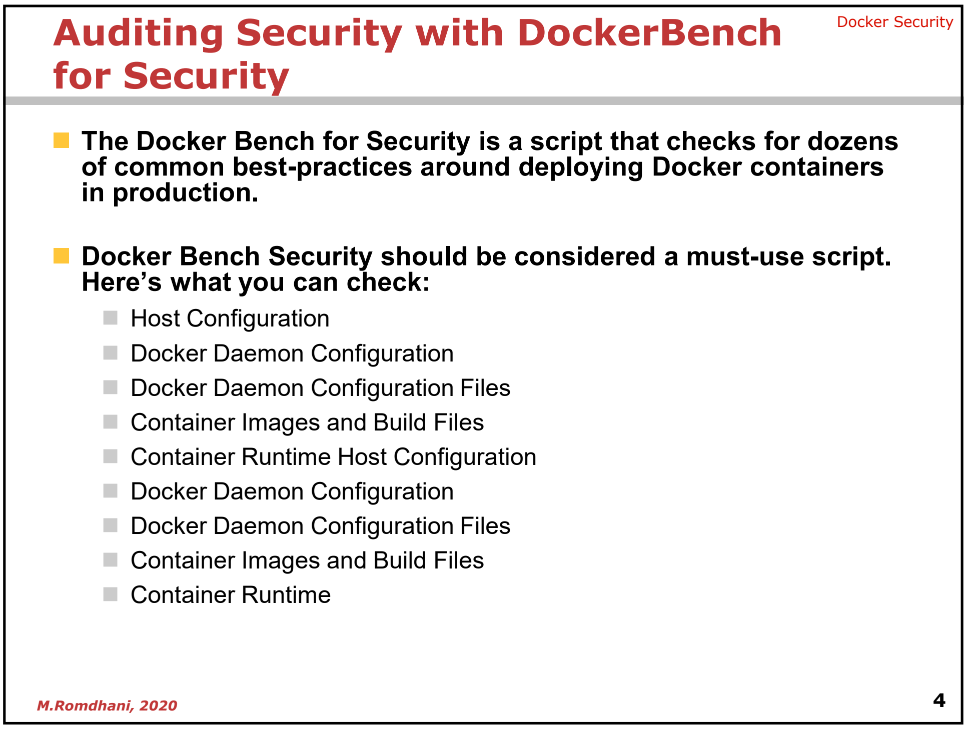


Docker Security

Docker Entreprise Security

- **Docker Entreprise Edition adds an extra layer of protection that travels with your applications in a secure supply chain that traverses any infrastructure and across the application lifecycle.**
  - And with a single interface and centrally-managed content, you get a seamless workflow that improves governance and ensures compliance across your whole organization.
- **Should You Use Docker CE or Docker EE?**
  - Deciding between Docker CE and Docker EE? For most, it boils down to these key questions:
    - Will you need same-day Docker support?
    - Does your organization's security policies compel you to use certified Docker images and plugins?
    - Would you like a single pane view to assist you in running, managing, and securing a wide range of containers in Linux and Windows?
    - How do you plan to visualize and manage your container environment? Do you require the 'Standard' and 'Advanced' Docker EE solutions which include this functionality?
    - Do you require enhanced security protocols?

M.Romdhani, 2020 3

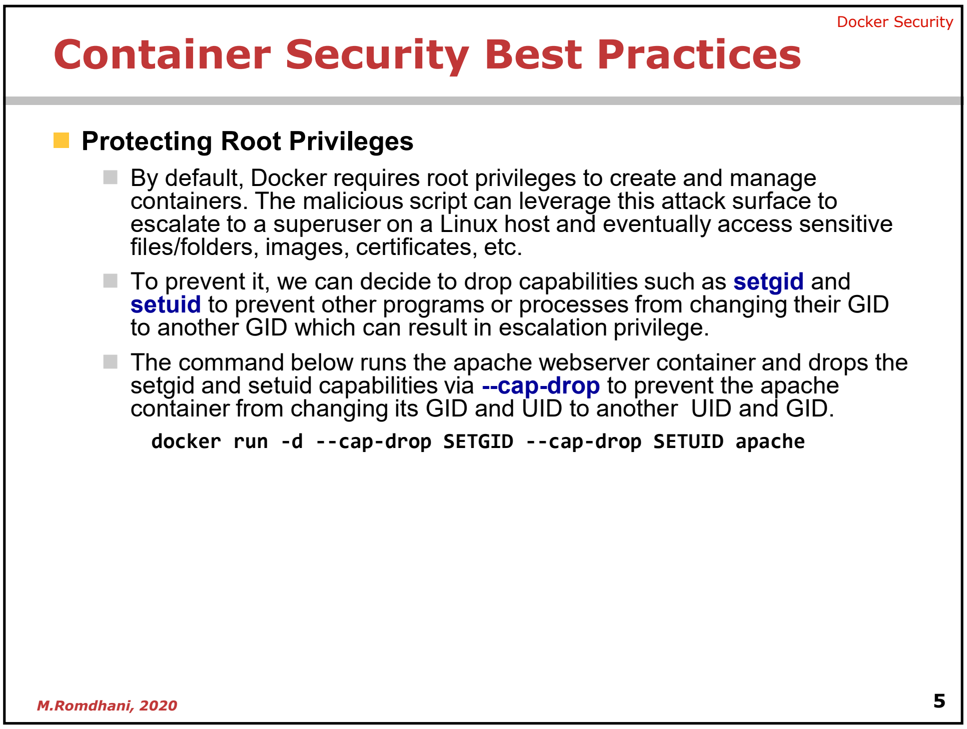


Docker Security

Auditing Security with DockerBench for Security

- **The Docker Bench for Security is a script that checks for dozens of common best-practices around deploying Docker containers in production.**
- **Docker Bench Security should be considered a must-use script. Here's what you can check:**
  - Host Configuration
  - Docker Daemon Configuration
  - Docker Daemon Configuration Files
  - Container Images and Build Files
  - Container Runtime Host Configuration
  - Docker Daemon Configuration
  - Docker Daemon Configuration Files
  - Container Images and Build Files
  - Container Runtime

M.Romdhani, 2020 4



Docker Security

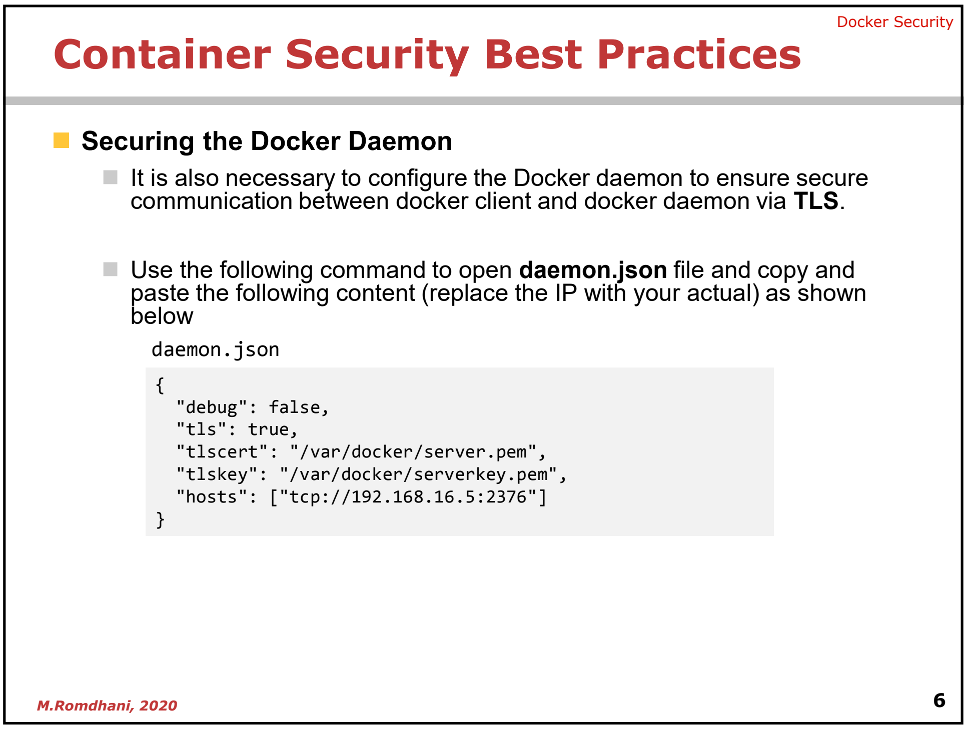
Container Security Best Practices

- **Protecting Root Privileges**
  - By default, Docker requires root privileges to create and manage containers. The malicious script can leverage this attack surface to escalate to a superuser on a Linux host and eventually access sensitive files/folders, images, certificates, etc.
  - To prevent it, we can decide to drop capabilities such as **setgid** and **setuid** to prevent other programs or processes from changing their GID to another GID which can result in escalation privilege.
  - The command below runs the apache webserver container and drops the setgid and setuid capabilities via **--cap-drop** to prevent the apache container from changing its GID and UID to another UID and GID.

```
docker run -d --cap-drop SETGID --cap-drop SETUID apache
```

M.Romdhani, 2020 **5**

5



Docker Security

Container Security Best Practices

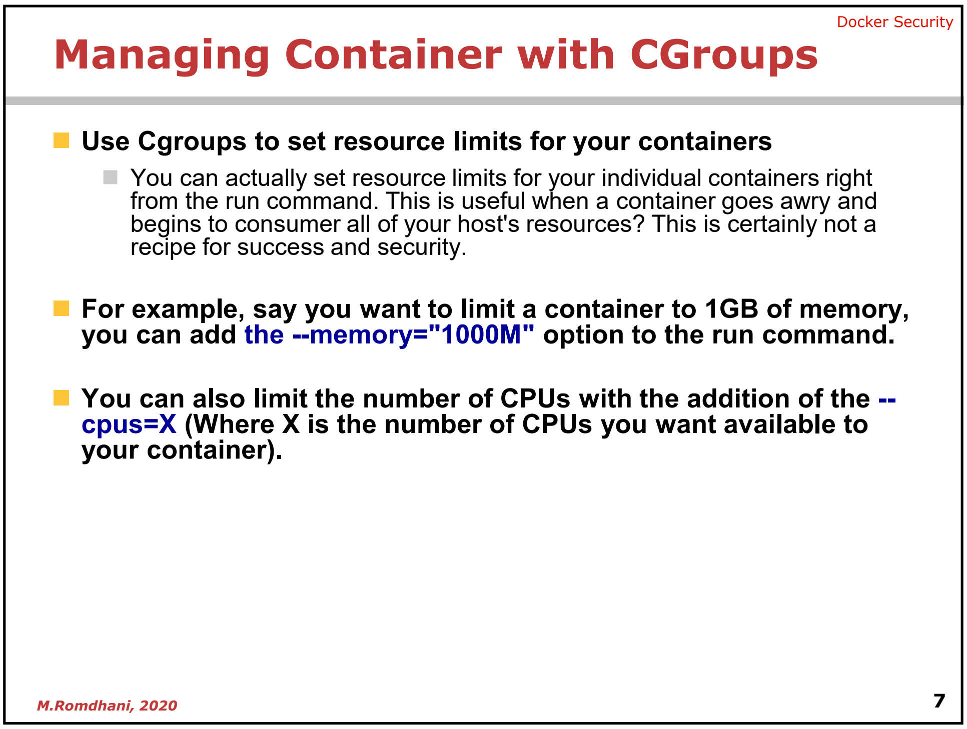
- **Securing the Docker Daemon**
  - It is also necessary to configure the Docker daemon to ensure secure communication between docker client and docker daemon via **TLS**.
  - Use the following command to open **daemon.json** file and copy and paste the following content (replace the IP with your actual) as shown below

daemon.json

```
{
  "debug": false,
  "tls": true,
  "tlscert": "/var/docker/server.pem",
  "tlskey": "/var/docker/serverkey.pem",
  "hosts": ["tcp://192.168.16.5:2376"]
}
```

M.Romdhani, 2020 **6**

6



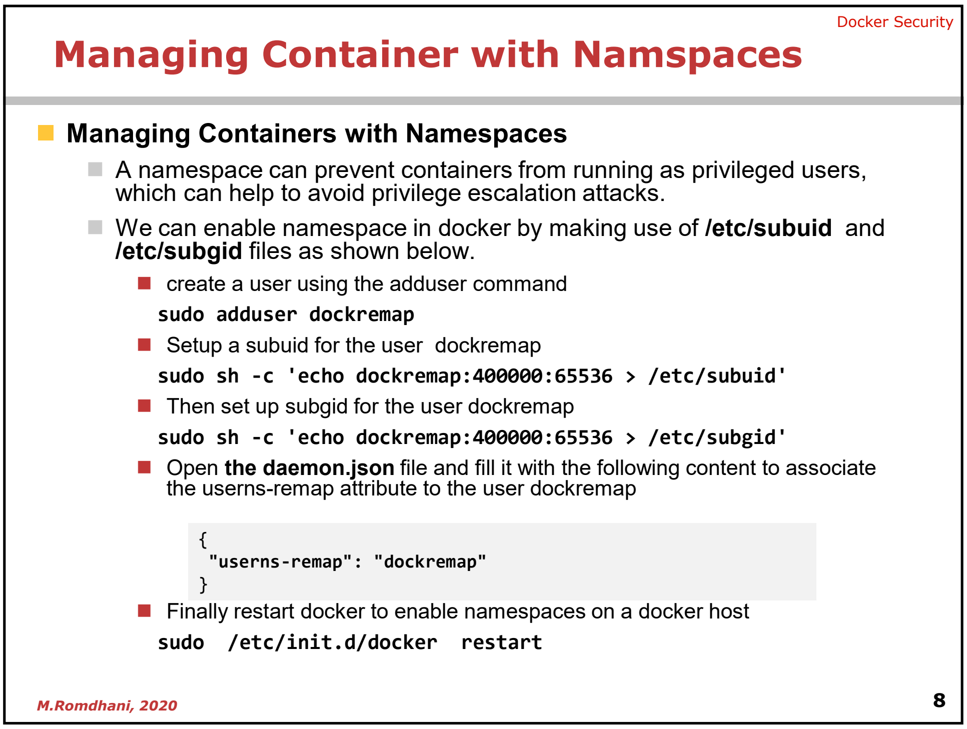
Docker Security

Managing Container with CGroups

- **Use Cgroups to set resource limits for your containers**
  - You can actually set resource limits for your individual containers right from the run command. This is useful when a container goes awry and begins to consume all of your host's resources? This is certainly not a recipe for success and security.
- **For example, say you want to limit a container to 1GB of memory, you can add the --memory="1000M" option to the run command.**
- **You can also limit the number of CPUs with the addition of the --cpus=X (Where X is the number of CPUs you want available to your container).**

M.Romdhani, 2020 **7**

7



Docker Security

Managing Container with Namspaces

- **Managing Containers with Namespaces**
  - A namespace can prevent containers from running as privileged users, which can help to avoid privilege escalation attacks.
  - We can enable namespace in docker by making use of **/etc/subuid** and **/etc/subgid** files as shown below.
    - create a user using the adduser command

```
sudo adduser dockremap
```

    - Setup a subuid for the user dockremap

```
sudo sh -c 'echo dockremap:400000:65536 > /etc/subuid'
```

    - Then set up subgid for the user dockremap

```
sudo sh -c 'echo dockremap:400000:65536 > /etc/subgid'
```

    - Open **the daemon.json** file and fill it with the following content to associate the userns-remap attribute to the user dockremap

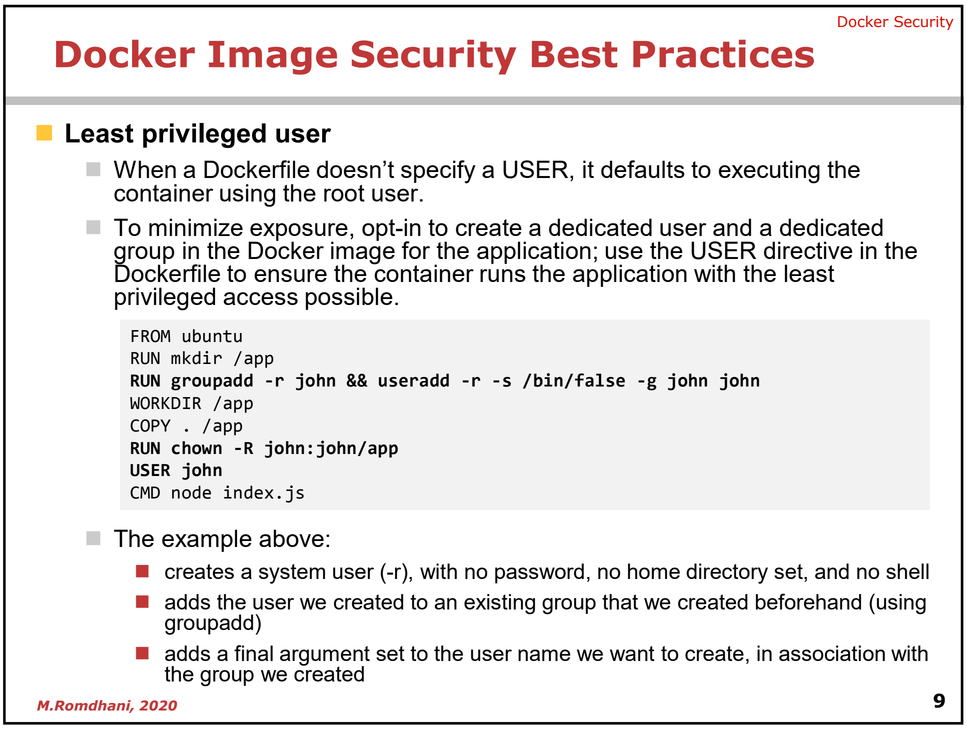
```
{
 "userns-remap": "dockremap"
}
```

    - Finally restart docker to enable namespaces on a docker host

```
sudo  /etc/init.d/docker  restart
```

M.Romdhani, 2020 **8**

8

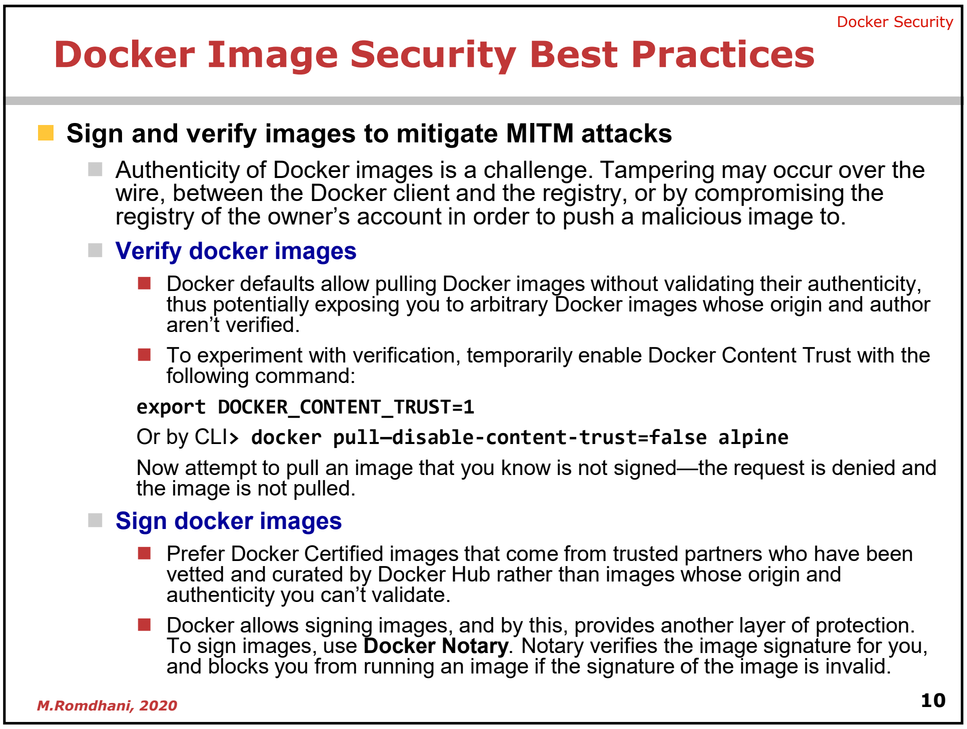


Docker Image Security Best Practices

- **Least privileged user**
  - When a Dockerfile doesn't specify a USER, it defaults to executing the container using the root user.
  - To minimize exposure, opt-in to create a dedicated user and a dedicated group in the Docker image for the application; use the USER directive in the Dockerfile to ensure the container runs the application with the least privileged access possible.

```
FROM ubuntu
RUN mkdir /app
RUN groupadd -r john && useradd -r -s /bin/false -g john john
WORKDIR /app
COPY . /app
RUN chown -R john:john/app
USER john
CMD node index.js
```

  - The example above:
    - creates a system user (-r), with no password, no home directory set, and no shell
    - adds the user we created to an existing group that we created beforehand (using groupadd)
    - adds a final argument set to the user name we want to create, in association with the group we created



Docker Image Security Best Practices

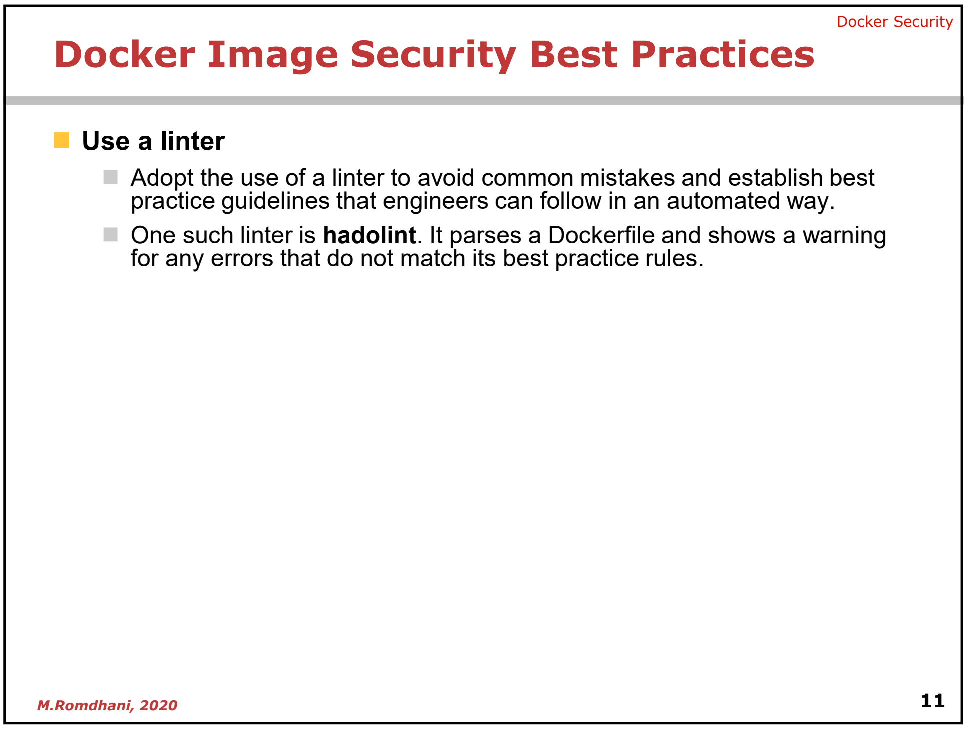
- **Sign and verify images to mitigate MITM attacks**
  - Authenticity of Docker images is a challenge. Tampering may occur over the wire, between the Docker client and the registry, or by compromising the registry of the owner's account in order to push a malicious image to.
  - **Verify docker images**
    - Docker defaults allow pulling Docker images without validating their authenticity, thus potentially exposing you to arbitrary Docker images whose origin and author aren't verified.
    - To experiment with verification, temporarily enable Docker Content Trust with the following command:

`export DOCKER_CONTENT_TRUST=1`

Or by CLI> `docker pull–disable-content-trust=false alpine`

Now attempt to pull an image that you know is not signed—the request is denied and the image is not pulled.

  - **Sign docker images**
    - Prefer Docker Certified images that come from trusted partners who have been vetted and curated by Docker Hub rather than images whose origin and authenticity you can't validate.
    - Docker allows signing images, and by this, provides another layer of protection. To sign images, use **Docker Notary**. Notary verifies the image signature for you, and blocks you from running an image if the signature of the image is invalid.



Docker Security

Docker Image Security Best Practices

- **Use a linter**
  - Adopt the use of a linter to avoid common mistakes and establish best practice guidelines that engineers can follow in an automated way.
  - One such linter is **hadolint**. It parses a Dockerfile and shows a warning for any errors that do not match its best practice rules.

M.Romdhani, 2020

11

11