Identify vulnerabilities and security issues in container images

Scan Docker images using Amazon Elastic Container Registry (ECR)

How to Find and Fix Vulnerabilities in Your Container Image

Image Scanning: -

ECR provides integrated image scanning capabilities to help you identify vulnerabilities and security issues in your container images. It uses the Common Vulnerabilities and Exposures (CVE) database to detect known vulnerabilities and provides actionable insights to remediate them.

Pre-requirements: -

- ✓ To get started with AWS ECR,
- ✓ you must have AWS Account Install aws-cli in local system/server/VM/EC2.
- ✓ Create IAM user and give access to ECR roles.
- ✓ Create ECR repository Configure aws-cli with IAM Credentials.
- ✓ Create docker file for Build docker image for scanning image.

We will understand and achieve hands on lab.

- How to use private docker registry AWS ECR - Elastic Container Registry
- How to create repository on AWS ECR
- How to authenticate in AWS ECR
- How to push private image to AWS ECR
- How to pull private image from AWS ECR.

Scan Docker images using Amazon Elastic Container Registry (ECR), you can follow these step-by-step Instructions:

AWS CLI Installation:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install
```

```
aws --version
```

```
[ec2-user@ip-172-31-88-239 ~]$ aws --version
aws-cli/2.11.27 Python/3.11.3 Linux/6.1.29-47.49.amzn2023.x86_64 exe/x86_64.amzn.2023 prompt/off
[ec2-user@ip-172-31-88-239 ~]$
```

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Docker Installation:

sudo yum amazon-linux-extras install docker

sudo service docker start

(Log out and log back in again to pick up the new `docker` group permissions)

```
[ec2-user@ip-172-31-88-239 ~]$ sudo yum install docker
Last metadata expiration check: 1:32:32 ago on Mon Jun 12 07:51:14 2023.
Dependencies resolved.
================================================================================
 Package                                                Architecture
================================================================================
Installing:
 docker                                                 x86_64
Installing dependencies:
 containerd                                             x86_64
 iptables-libs                                          x86_64
 iptables-nft                                           x86_64
 libcgroup                                              x86_64
 libnetfilter_conntrack                                 x86_64
 libnfnetlink                                           x86_64
 libnftnl                                               x86_64
 pigz                                                   x86_64
 runc                                                   x86_64

Transaction Summary
================================================================================
Install  10 Packages

Total download size: 77 M
Installed size: 300 M
Is this ok [y/N]:
```

```
[ec2-user@ip-172-31-88-239 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-88-239 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-88-239 ~]$ logout
```

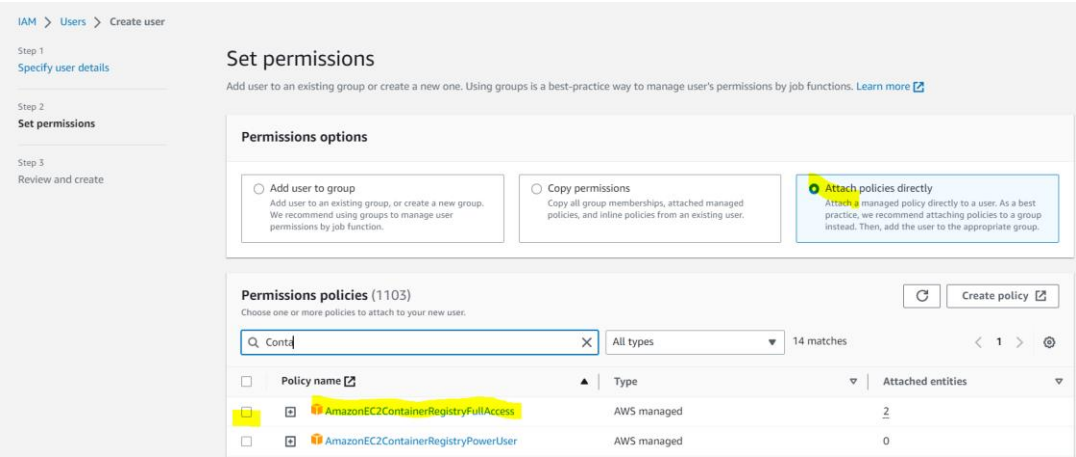
```
[ec2-user@ip-172-31-88-239 ~]$ docker --version
Docker version 20.10.23, build 7155243
```

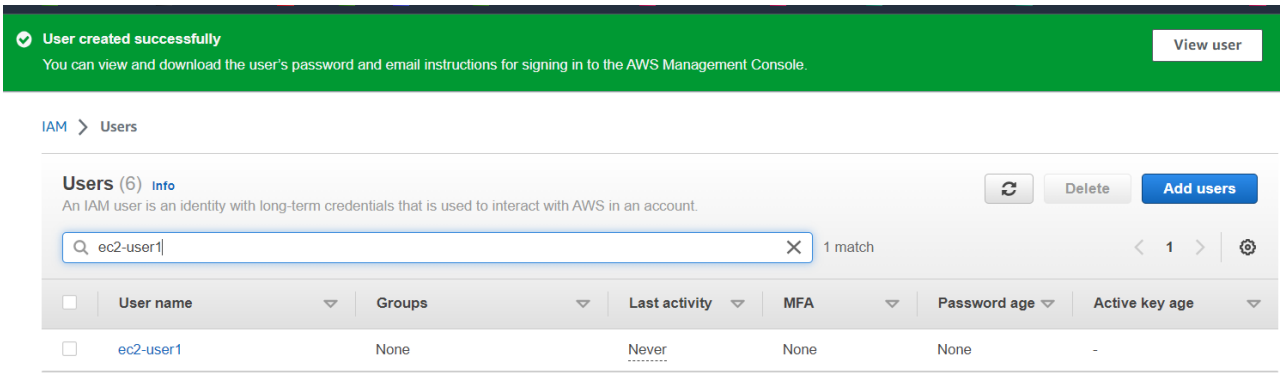
https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html

https://docs.docker.com/engine/install/

Create ECR repository Configure aws-cli with IAM Credentials

IAM Users > Create > user





Create access key, secret key and download save it.

Set up an ECR repository:

- Go to the Amazon ECR service in the AWS Management Console.
- Click on "Create repository" and provide a name for your repository.
- Configure the repository settings, such as access permissions, lifecycle policy, and encryption options.

Click on "Create repository" to complete the setup

Amazon ECR >Repositories >Create repository

Amazon ECR > Repositories > Create repository

Create repository

General settings

Visibility settings Info

Choose the visibility setting for the repository.

- (•) Private
  Access is managed by IAM and repository policy permissions.
- ( ) Public
  Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

557822060553.dkr.ecr.us-east-1.amazonaws.com/ ecr

3 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability Info

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

[ ] Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

Amazon ECR > Repositories

Private | Public

Private repositories (1)

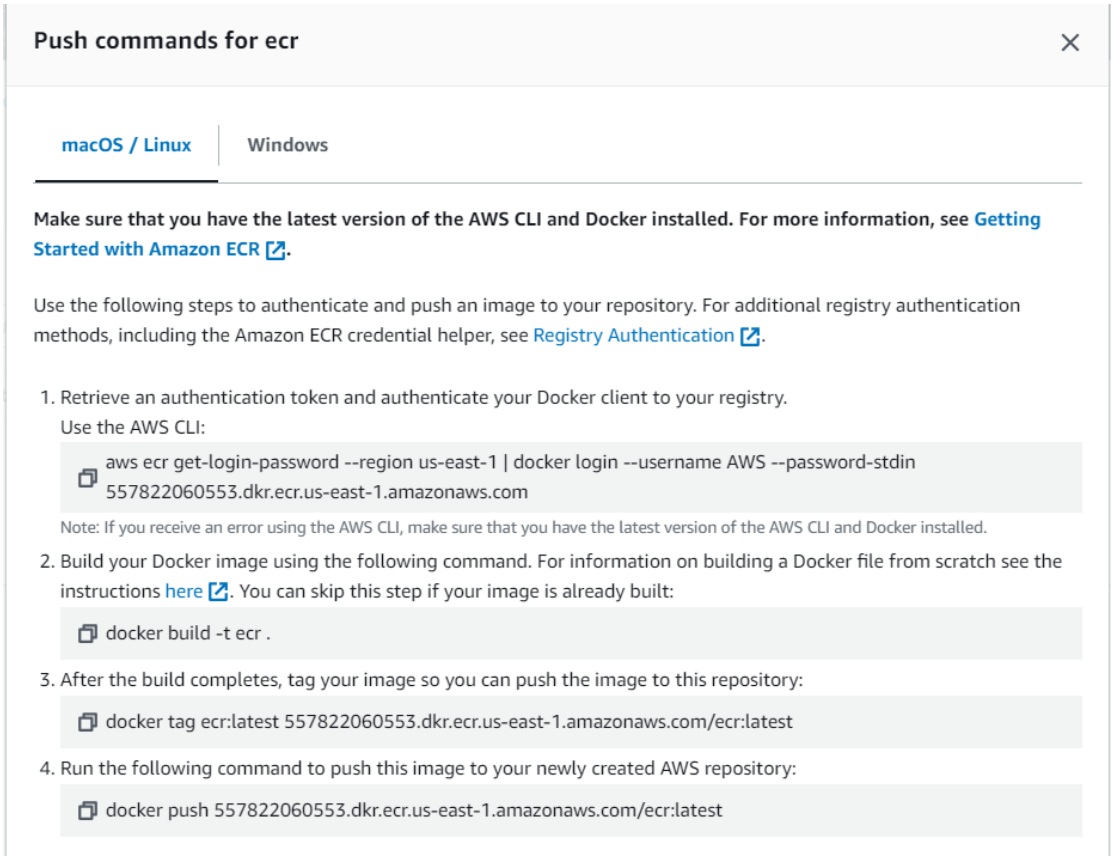
View push commands | Delete | Actions | Create repository

Find repositories

1

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
ecr	557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr	June 12, 2023, 15:27:13 (UTC+05.5)	Disabled	Manual	AES-256	Inactive

Click view commands for Push commands for Repositories



<u>Create docker file for Build docker image for scanning image: -</u>

Create a Dockerfile

touch Dockerfile

nano Docekerfile

Edit the `Dockerfile` you just created and add the following content.

```
[ec2-user@ip-172-31-88-239 ~]$ touch Dockerfile
[ec2-user@ip-172-31-88-239 ~]$ nano Dockerfile
[ec2-user@ip-172-31-88-239 ~]$
```

```
FROM public.ecr.aws/docker/library/ubuntu:18.04

# Install dependencies
RUN apt-get update && \
 apt-get -y install apache2

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && \
 echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
 echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80
^G Help          ^O Write Out     ^W Where Is      ^K Cut
CMD /root/run_apache.shd File     ^\ Replace       ^U Paste
ext
```

FROM public.ecr.aws/docker/library/ubuntu:18.04

Install dependencies

RUN apt-get update && \

 apt-get -y install apache2

Install apache and write hello world message

RUN echo 'Hello World!' > /var/www/html/index.html

Configure apache

RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && \

 echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \

 echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \

 echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \

 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh

docker build -t ecr .

```
[ec2-user@ip-172-31-88-239 ~]$ ls
Dockerfile  aws  awscliv2.zip
[ec2-user@ip-172-31-88-239 ~]$ docker build -t ecr .
Sending build context to Docker daemon  260.1MB
Step 1/6 : FROM public.ecr.aws/docker/library/ubuntu:18.04
18.04: Pulling from docker/library/ubuntu
7c457f213c76: Pull complete
Digest: sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Status: Downloaded newer image for public.ecr.aws/docker/library/ubuntu:18.04
 ---> f9a80a55f492
Step 2/6 : RUN apt-get update &&  apt-get -y install apache2
 ---> Running in 7a30b113e6d1
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [83.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3771 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2403 kB]
```

docker tag ecr:latest 557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr:latest

```
 ---> 9cfe33c245c4
Successfully built 9cfe33c245c4
Successfully tagged ecr:latest
[ec2-user@ip-172-31-88-239 ~]$ docker images
REPOSITORY                              TAG       IMAGE ID       CREATED              SIZE
ecr                                     latest    9cfe33c245c4   About a minute ago   205MB
public.ecr.aws/docker/library/ubuntu    18.04     f9a80a55f492   13 days ago          63.2MB
[ec2-user@ip-172-31-88-239 ~]$ docker tag ecr:latest 557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr:latest
[ec2-user@ip-172-31-88-239 ~]$ docker images
REPOSITORY                                          TAG       IMAGE ID       CREATED          SIZE
557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr    latest    9cfe33c245c4   3 minutes ago    205MB
ecr                                                 latest    9cfe33c245c4   3 minutes ago    205MB
public.ecr.aws/docker/library/ubuntu                18.04     f9a80a55f492   13 days ago      63.2MB
[ec2-user@ip-172-31-88-239 ~]$
```

https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html

Before Pushing Container, images configure aws cli.

aws configure: -

```
[ec2-user@ip-172-31-88-239 ~]$ aws configure
AWS Access Key ID [None]: AKIAYDYGJFAE7RJBD6EP
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
[ec2-user@ip-172-31-88-239 ~]$
```

Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CL

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 557822060553.dkr.ecr.us-east-1.amazonaws.com

```
[ec2-user@ip-172-31-88-239 ~]$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-
stdin 557822060553.dkr.ecr.us-eaws ecr get-login-password --region us-east-1 | docker login --username AWS --password-
stdin 557822060553.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-88-239 ~]$
```

Run the following command to push this image to your newly created AWS repository

```
[ec2-user@ip-172-31-88-239 ~]$ docker images
REPOSITORY                                          TAG       IMAGE ID       CREATED          SIZE
557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr    latest    9cfe33c245c4   10 minutes ago   205MB
ecr                                                 latest    9cfe33c245c4   10 minutes ago   205MB
public.ecr.aws/docker/library/ubuntu                18.04     f9a80a55f492   13 days ago      63.2MB
[ec2-user@ip-172-31-88-239 ~]$ docker push 557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr:latest
The push refers to repository [557822060553.dkr.ecr.us-east-1.amazonaws.com/ecr]
71c961713b95: Pushed
7bf3a3cad6c3: Pushed
725a918b4124: Pushed
548a79621a42: Pushed
latest: digest: sha256:bd723399a453822c7494042914578752f3ba81a33fd5b96a1838c6da5a80d20c size: 1155
[ec2-user@ip-172-31-88-239 ~]$
```

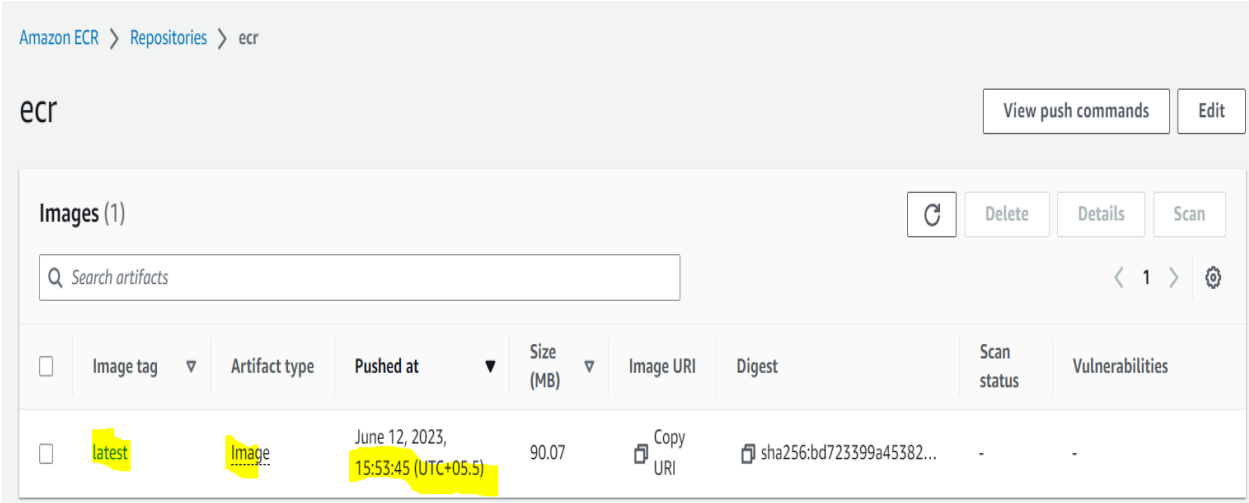


Image scan started successfully

Amazon ECR > Repositories > ecr

ecr

View push commands | Edit

Images (1)

Delete | Details | Scan

Search artifacts

1

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
latest	Image	June 12, 2023, 15:53:45 (UTC+05.5)	90.07	Copy URI	sha256:bd723399a45382...	In progress	-

latest	Image	June 12, 2023, 15:53:45 (UTC+05.5)	90.07	Copy URI	sha256:bd723399a45382...	Complete	5 Medium + 11 others (details)

Amazon ECR > Repositories > ecr > sha256:bd723399a453822c7494042914578752f3ba81a33fd5b96a1838c6da5a80d20c

Overview

Critical	High	Medium	Low	Informational	Undefined
0	0	5	7	4	0

Vulnerabilities (16)

1

Name	Package	Severity	Description
CVE-2019-17567	apache2:2.4.29-1ubuntu4.27	MEDIUM	Apache HTTP Server versions 2.4.6 to 2.4.46 mod_proxy_wstunnel configured on an URL that is not necessarily Upgraded by the origin server was tunneling the whole connection regardless, thus allowing for subsequent requests on the same connection to pass through with no HTTP validation, authentication or authorization possibly configured.
CVE-2020-13844	gcc-8:8.4.0-1ubuntu1~18.04	MEDIUM	Arm Armv8-A core implementations utilizing speculative execution past unconditional changes in control flow may allow unauthorized disclosure of information to an attacker with local user access via

Vulnerabilities (16)

1

Name	Package	Severity	Description
CVE-2019-17567	apache2:2.4.29-1ubuntu4.27	MEDIUM	Apache HTTP Server versions 2.4.6 to 2.4.46 mod_proxy_wstunnel configured on an URL that is not necessarily Upgraded by the origin server was tunneling the whole connection regardless, thus allowing for subsequent requests on the same connection to pass through with no HTTP validation, authentication or authorization possibly configured.
CVE-2020-13844	gcc-8:8.4.0-1ubuntu1~18.04	MEDIUM	Arm Armv8-A core implementations utilizing speculative execution past unconditional changes in control flow may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis, aka "straight-line speculation."
CVE-2020-11080	nghttp2:1.30.0-1ubuntu1	MEDIUM	In nghttp2 before version 1.41.0, the overly large HTTP/2 SETTINGS frame payload causes denial of service. The proof of concept attack involves a malicious client constructing a SETTINGS frame with a length of 14,400 bytes (2400 individual settings entries) over and over again. The attack causes the CPU to spike at 100%. nghttp2 v1.41.0 fixes this vulnerability. There is a workaround to this vulnerability. Implement nghttp2_on_frame_recv_callback callback, and if received frame is SETTINGS frame and the number of settings entries are large (e.g., > 32), then drop the connection.
CVE-2019-9511	nghttp2:1.30.0-1ubuntu1	MEDIUM	Some HTTP/2 implementations are vulnerable to window size manipulation and stream prioritization manipulation, potentially leading to a denial of service. The attacker requests a large amount of data from a specified resource over multiple streams. They manipulate window size and stream priority to force the server to queue the data in 1-byte chunks. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both.
CVE-2019-9513	nghttp2:1.30.0-1ubuntu1	MEDIUM	Some HTTP/2 implementations are vulnerable to resource loops, potentially leading to a denial of service. The attacker creates multiple request streams and continually shuffles the priority of the streams in a way that causes substantial churn to the priority tree. This can consume excess CPU.

The End

Thanks for Everyone

Manikanta Suru