**A3 – Securing DevOps**

## **Docker**

##### **1. Use dive (https://github.com/wagoodman/dive) or any other method to find the file that was added and later removed from the image. Provide screenshots of the steps/results to complete the task.**

For analyzing the image, we used “dive” and “docker scout”. Based on the scan, we see that the file that gets added and removed later during the image build process is “gitlab\_pat” in the app/ directory.

Screenshots from DIVE:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Additional Screenshots from DOCKER SCOUT:

A screenshot of a computer

AI-generated content may be incorrect.

##### **2. Is it possible to get the contents of the deleted file? If yes, explain why and show the steps to do it with screenshots and elaborate how. If not, why not? Explain in detail how docker images work internally to support your answer.**

To get the file contents, using dive we found that the hash of the layers that are associated with this process:

* Add pat: 619733a14a2d7f311ccbfa4e16a562eb9f924b9684bf86f2c34324cc003ace87

To examine further, we create a tarball of each image layer using the following process:

docker save registry.gitlab.com/sl0wc0der/private-packages/assignment2:v1 -o image.tar

mkdir image-layers

tar -xf image.tar -C image-layers

Once extracted, we can see the following directory structure, and find the hash associated with the command that copies the token.

A screenshot of a computer

AI-generated content may be incorrect.

Extract the contents of this hash

A black and white text

AI-generated content may be incorrect.

This app directory contains the gitlab\_pat token file.

A screen shot of blue text

AI-generated content may be incorrect.

When a docker image is built, it is constructed in layers. i.e. Each command in the Dockerfile creates a new layer on top of the previous one for more efficiency by using caches to allow reuse of unmodified layers. Each layer contains only the difference (changes made by the command). So, when a file is removed during the build process, it is removed only in the upcoming layers, not the underlying ones.

##### **3. Assume that we want to be able to clone the repo inside the docker image with the restriction that we can only build the image with `docker build . -t <image\_name>:<version>`. How would you modify the Dockerfile to improve its security (not limited to the git clone operation)? Submit the final contents of the Dockerfile and explain the changes/choices made.**

We could follow the multi-stage docker build process inorder to improve the Dockerfile to restrict the sensitive operations in the build stage, also maintain the build process to use the command ` docker build . -t <image\_name>:<version>`:

A computer screen shot of a program

AI-generated content may be incorrect.

When we build the image using multi-stage build process, we use the token file only in the build\_stage, which is not included in the final image. Due to this, any sensitive operations such as copying tokens, cloning private repositories are isolated to the build\_stage and any files present in build\_stage layer will not be visible in the final image. For example, below is a screenshot of the build process of another sample image using a similar sensitive operation (adding/removing token). As we can see, dive cannot view the layers associated with the process. This makes it more secure.

An alternative approach is to pass the token as a build-time argument.

|  |  |
| --- | --- |
| Screenshot from DIVE: | Screenshot from DOCKER SCOUT: |
| A black screen with white text  AI-generated content may be incorrect. | A black rectangular object with a black stripe  AI-generated content may be incorrect. |

As we can see, after modifying the Dockerfile to create the image using the multi-stage build process, we can’t see any additional information about the sensitive tasks being performed.

## **Git**

##### **4. There's a git repo being cloned into the docker image. How can you use git commands to make sure that no secrets exist in the repo? Submit the steps taken to make sure the repo is safe from any secrets. You can also use external tools for this task if needed.**

To examine the git repo being cloned, we need find the hash of the layer performing git clone process using dive and extract its contents.



A screen shot of a computer

AI-generated content may be incorrect.

We can use the .git directory to examine the git commit log to fetch any secrets. For this step, we used different strategies to look for secrets: in both active files, and in the commit history.

A screen shot of a computer screen

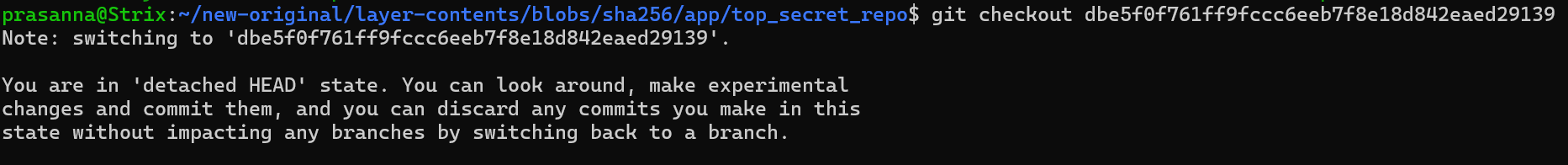
AI-generated content may be incorrect.

Based on the initial search we can see that “secrets.txt” file has been added & removed during the development process. We examine the commit history a bit deeper.

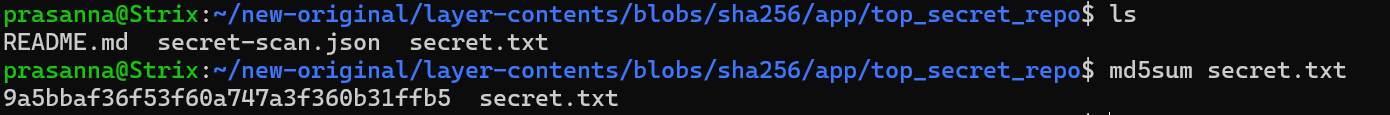
A screenshot of a computer

AI-generated content may be incorrect.

Based on the content of “secrets.txt”, it says that we need to submit the md5sum of the file. To do this, we need to retrieve the file by reverting to the commit that added the file.



md5sum of the file:



##### **5. If you find any secret contents/files, also provide the commands to fix the existing git repo to eliminate the secrets.**

To remove the trace of secrets.txt from the commit history, we need to rewrite the commit history using `git filter-branch --force --index-filter 'rm -f secrets.txt' -- --all` command and force push the update history.

A computer screen with white text

AI-generated content may be incorrect.

To avoid pushing secrets, many git providers come with secret push protection features. As a security best practice, we can use environmental variables, key management systems, etc. to store secrets that are needed during the development. Additionally, we can use .gitignore config file to avoid committing any sensitive files in our repository.

## **Python/Pip**

##### **6. Identify the security issues with the requirements.txt file and explain the issues you find in detail.**

A screen shot of a computer

AI-generated content may be incorrect.

* The requirements.txt file reveals the url to a private registry. Although this URL alone may not be exploited, this could still reveal the inner structure of the repository.
* The packages are mentioned without specifying whether it is hosted in PyPi or the private registry.
* Also, the versions for the packages are not pinned to a specific version.

##### **7. Provide proof-of-concept exploits for all the issues identified along with screenshots of the exploits. You are not required to perform the exploit inside a Docker container.**

* By revealing the URL of the private registry, the attackers could target the registry to perform brute-force attacks to gain access.
* If the package is not found in PyPi, and it will be fetched from the private registry. If the registry is compromised, an attacker could cause installation of a malicious package. Also, an attacker(insider) could create a package with similar name, and with “latest” tag in the same registry. When pip tries to fetch the package, it would give preference to the package with the “latest” tag, and it might lead to malicious code execution.
* Due to absence of version pinning, we could face compatibility issues and crash the application when there is an incompatible version upgrade.

###### Proof of concept:

**Step-1:** In this, we will use our devpi server as a replacement to the official PyPi registry. For this, we need to install and run the devpi server in our machine:

pip install devpi-server

devpi-init

A screen shot of a computer

AI-generated content may be incorrect.

**Step-2**: In another terminal, setup the devpi-server index and login

devpi use http://localhost:3141

devpi login root --password=

devpi index -c myindex bases=root/pypi volatile=False

devpi use root/myindex

A screenshot of a computer program

AI-generated content may be incorrect.

**Step-3**: Setup a malicious package into our server. For this, setup the following directory contents.

A screenshot of a computer program

AI-generated content may be incorrect.

Here, \_\_init\_\_.py is just an empty file to indicate that it is a package. The setup.py contents are as follows:

# setup.py

from setuptools import setup

import os

os.system("echo 'Malicious Code Executed!' > /tmp/malicious.txt")

setup(

name="much\_needed\_python\_package",

version="0.0.1",

description="A malicious package to demonstrate dependency confusion",

packages=["much\_needed\_python\_package"],

)

**Step-4**: Build and upload the package

A screen shot of a computer code

AI-generated content may be incorrect.

A computer screen with white text

AI-generated content may be incorrect.

**Step-5**: Configure pip to use our devpi-server instead of the official PyPi registry using the following command

pip config set global.index-url http://localhost:3141/root/myindex/+simple/

**Result:**

For this, we will use the same requirements.txt file given. In this, only the requests is a valid package, and it is fetched from the private registry, while the other one does not exist in private registry. So, pip will try to fetch it from the devpi server and executes the malicious code that writes a file.

A screenshot of a computer program

AI-generated content may be incorrect.

##### **8. Fix the requirements files and the Dockerfile if required by your fix along with an explanation of the fixes.**

A screen shot of a computer

AI-generated content may be incorrect.

* Added --index-url to specify the defaul search source as the Official PyPi.
  + It ensures that the packages are fetched from a trusted, centralized source.
  + It mitigates the risk of dependency confusion, where a package might be inadvertently pulled from an untrusted or compromised repository.
* Mentioned the specific versions of the packages that need to be installed.
  + Version pinning ensures consistent package versions across environments, aiding debugging and stability. It enhances security by avoiding unverified updates and prevents unexpected issues from newer versions.