Writeup by: linkedin.com/in/guy-eldad/

Docker Forensics

You are a skilled digital forensics investigator facing a challenging task. An intriguing Docker image awaits your analysis, with multiple layers concealing crucial information and clues. Your goal is to uncover these hidden layers, examine their contents, and solve the mystery by piecing together the puzzle.
Put your skills to the test, as each layer holds a new surprise and valuable insights.
Can you unveil the secrets lurking beneath the surface of this enigmatic Docker image?

Image name: **mmox/what-is-0xl4ugh**

Question 1: How many layers are in this docker container that you pull into your device?

First, we need to inspect the docker image.
We can use the inspect command to get low-level configuration information about various Docker objects like images, containers, volumes, networks, nodes, etc.

The command we will use is “docker inspect image name"
In our scenario the image is mmox/what-is-0xl4ugh

By using this command, we can see the Layers that was requested from the Q1:

```
"Type": "layers",
"Layers": [
    "sha256:8cbe4b54fa88d8fc0198ea0cc3a5432aea41573e6a0ee26eca8c79f9fbfa40e3",
    "sha256:2656be2aa5fac8a343862df7e21ec539a566703932c43e3c7d1ad118116b5045",
    "sha256:9a6e8fbbcab1a21c0c3e537623067533ab46b3d868e3512f2e6cd072cf328535",
    "sha256:d682838149b05ce191482d9ce83b77402962c3c756643012d4a75eac0671ce99",
    "sha256:91692ec0375d041999e209bd4e61cfa1498373895517477cf4207a14873f7d84",
    "sha256:d54990acf4bf523e54f01d5739e26871ca29756d84ffc0d8ab4a1ea658203c2b",
    "sha256:5e33913e7a80c68ff4db44bdbd1005074c0cfc33a410555bf8f62b34f4f34a44",
    "sha256:bc67315bbd3edcf08cf90780b8e45def6bb8f86ab5f6dd7167fa5c374c6f2b82",
    "sha256:141ab80e91a29919b1649b326786949caef6dc911ffc64dcef2d42fc4932f26d",
    "sha256:1c7bc8e4260f2da1c388177406c45b6310d82a8a2a506d8625c6cf818cb54fc5",
    "sha256:e7487bb4bb0fcdeff35ac407114df33dd80fb448ad443907da80d383fb7fb13d",
    "sha256:f9878015f10b6ee3205210eed3ea2f47e76a69d12cc68e3eb4f231e98d1e6c5b",
    "sha256:c0c53cc6687cfd3116af6f4b8bca2d061c0114bdfa5096a9f88058080fabe685",
    "sha256:827b50e8b7c6e94660239e0919a6f7c0218cb2b7cd86098462561f5576aafb21",
    "sha256:b0934e6625238b23d9e3762d6cff2a4723147d290e4fe6a00bc5db0a75c44c7f",
    "sha256:37140add3262ad8312000eac1f25fdc28e42723415947b6f5383bae68746d8a3",
    "sha256:31c257542eda40a332f623ec8920b78cce78452625fe4f8c2f8aae85eac60bf8",
    "sha256:3de8b31e158a7a4a40c2b0e1fb4d8ff32e652f1053dae4a9749ab9a9a83e4b64",
    "sha256:6801aa12f352cd8aa0c38a95d7d8a19788caea3d83dbc11c149e61e9dc8e353e",
    "sha256:97092da5da4c89b2cdc9b07b019b43b6dc800edc927dd6b56542dd377be2101f",
    "sha256:f47cc1a54ff9639bb22a26ef297c07cf185e39c438e1abd75c4e2b27e2805c64",
    "sha256:ee8e1c967bbe3dd47c4a7ae328814fac741ddc5b4f68729371d634d0e93f27b8",
    "sha256:4a3b630d2fca30d3fa2c1c6d1fe8e94ec08015aea49f001717e4bc1c7b2ab406"
```


Answer: 23

Question 2: What is the web server that the image used?

We can locate the web server using the same command we used in the previous question. "inspect."


```
  "PHP_INI_DIR=/usr/local/etc/php",
  "APACHE_CONFDIR=/etc/apache2",
  "APACHE_ENVVARS=/etc/apache2/envvars"
  "PHP_CFLAGS=-fstack-protector-strong
```


Answer: apache2

Question 3: What is the complete GitHub URL that a Docker container uses to fetch a website?

For this question, we need to review the history of activity performed on the docker image. We can use the command “docker history”. The docker history command provides a timeline of an image, revealing the commands used to create each layer.

The history command has various plugins, and the one we will use is "--no-trunc".
The full command is "docker history --no-trunc image name"
The docker history --no-trunc command is used to display the detailed history of an image. It provides information about each layer within the image and details about the creation of each layer.

Now we will search for the GitHub URL.

```
                                                                                71.3kB    buildkit.dockerf
ile.v0
<missing>                                                                   11 months ago   RUN /bin/sh -c git cl
one https://github.com/0xMM0X/BIG-GHAZY.git /var/www/html # buildkit
```

Answer: https://github.com/0xMM0X/BIG-GHAZY.git

Question 4: What is the database username and password? Format: User:password

To obtain the database username and password, we'll need to execute the Docker image to access it. For this purpose, we'll utilize the following command:
"docker run -it mmox/what-is-0xl4ugh bash"
This command is used to run a Docker container interactively with a Bash shell.

By using the ls command, we can see several files, one of which is db.php.
db.php is commonly named for files that handle database connections or configurations. Therefore, it's plausible that db.php could contain a username and password for a database connection.

We will first use cat on this file to check if the database contains the username and password we are looking for.

```
root@ip-172-31-21-159:~# docker run -it mmox/what-is-0xl4ugh bash
root@0a6dc18ff04c:/var/www/html# ls
css     fonts   index.php  login.php   notes      profile.php  scss  vendor
db.php  images  js         logout.php  notes.php  reg.php      up
root@0a6dc18ff04c:/var/www/html# cat db.php
<?php
$servername = "127.0.0.1";
$username = "Ghazy";
$password = "0xL4ugh_F0R_EV3R!!";
$dbname = "login";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}

?>


root@0a6dc18ff04c:/var/www/html#
```

Here we can see the username and password.

```
$servername = "127.0.0.1";
$username = "Ghazy";
$password = "0xL4ugh_F0R_EV3R!!";
$dbname = "login";
```

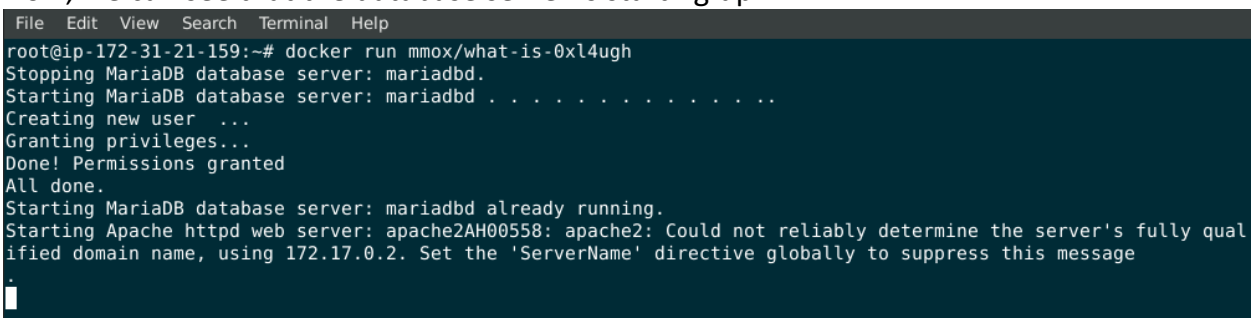
Answer: Ghazy:0xL4ugh_F0R_EV3R!!

Question 5: What is the secret that gets removed?

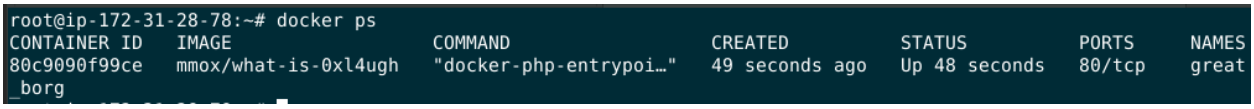
To answer this question, we need to gain access to the image container to inspects its files.

We will use the “docker run image name” command. This command is one of the fundamental commands in Docker used to create and start a Docker container from a Docker image.

Now, we can see that the database server is starting up.



We open a new tab and execute the “docker ps” command to list all running Docker containers.



Now, we need to use the command “docker diff container number” to inspect changes made to the filesystem of a container. This command compares the filesystem changes between the container's current state and its initial state when it was created from an image.

The output will display changes categorized with letters:

A: Added files or directories

C: Changed files or directories

D: Deleted files or directories

```
root@ip-172-31-21-159:~# docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS     NAMES
976733fe4aba   mmox/what-is-0xl4ugh   "docker-php-entrypoi…"   4 minutes ago   Up 4 minutes   80/tcp    happy_z
hukovsky
root@ip-172-31-21-159:~# docker diff 976733fe4aba
C /var
C /var/lib
C /var/lib/mysql
C /var/lib/mysql/aria_log.00000001
A /var/lib/mysql/ibtmp1
A /var/lib/mysql/login
A /var/lib/mysql/login/db.opt
A /var/lib/mysql/login/users.frm
A /var/lib/mysql/login/users.ibd
C /var/lib/mysql/mysql
C /var/lib/mysql/mysql/global_priv.MAI
C /var/lib/mysql/ib_logfile0
A /var/lib/mysql/multi-master.info
C /run
A /run/mysqld
A /run/mysqld/mysqld.pid
A /run/mysqld/mysqld.sock
C /run/apache2
A /run/apache2/apache2.pid
D /SUPER_SECRET.txt
root@ip-172-31-21-159:~#
```

The question was “**What is the secret that gets removed?**”

Here, we observe a filename **‘/SUPER_SECRET.TXT’** marked with the letter **‘D’,** indicating that this file was removed.

To retrieve the files and perform further investigation, we first need to create a new directory to save the files. Then, we will use the command “docker save image name -o docker.tar” to save the Docker image to a .tar file, allowing us to extract the files from the image.

```
root@ip-172-31-28-78:~# ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos  letsdefend.png  snap  ssl
root@ip-172-31-28-78:~# mkdir docker
root@ip-172-31-28-78:~# cd docker
root@ip-172-31-28-78:~/docker# docker save mmox/what-is-0xl4ugh -o docker.tar
root@ip-172-31-28-78:~/docker# ls
docker.tar
root@ip-172-31-28-78:~/docker#
```

After successfully obtaining the docker.tar file, we will use the command “tar -xf docker.tar” to decompress the image and access the blobs, enabling us to search for the files.

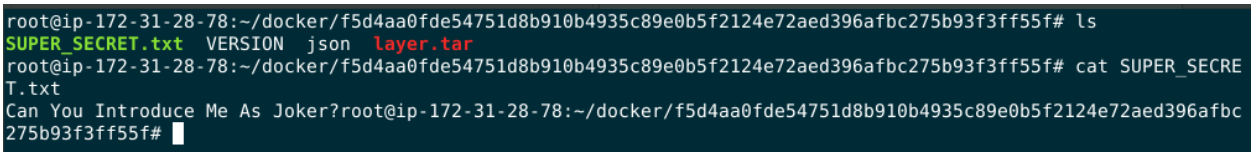
```
root@ip-172-31-21-159:~/docker# tar -xf docker.tar
root@ip-172-31-21-159:~/docker# ls
342a3a5dee0787f273bb91d1026a6af84f79c46bccaf9a3dee6edce10e593b7b
3a3240dfc8de2e827d9eabff067a6923a0be7c5cf5b2d22120aec1e6fe93a7f4
4367250f698446b5dd26468f28f7ca5ef70ba8005540593522fe31b447132b5f
49a1424a42ac4bcf81955a1e0ec47b97fd83091acf428d2a83e61b143316c4f7
4b6e67b2194a2cbb2837abbf5182d6a51d8091197f79a272cf28a1a4990b0d61
58f56435545ebb29538bc6a231b1924bc71a1b0a4a7998db1dcf139aa9cf7435
6398ee0434bc0e2fc0fc25a3ed9c878f0307db6d79307bb096461a5a7a4670b3
6865f854f5481615883ba3924c45008b995d5edc89d802c89019195f98b53296
691d5db0290fe48d674f88165373d102e1c0f72ca4af993e3aa73aad7bb55787
69fc127bb375b7b2645fffd6b43dc2e1a2b035b801bbbd6986c371fce90557cc
6d020808f2c3259c35ebe745cce9d3ded01b0bada9fa85c7ec62e2f8a0b7ce0a
6da2ffd88998964e41fd368190e2cc13b462274acd15cbf775a7e01acfb51126
747d69338298d82f27056f58420be3155ad5c0190ab478e39f4fb148e2bbbb37
8defb693b0034fb1e6d504e811ddaf2fbf47c52be78bef10e22326bc72395043.json
928864ddb741509f21d116b81e4cd3d19aa77326498bb37e6cd58dba7df10791
a0e37ea911977760442b9df0b0482cbfd9c30d76a71ea197ed6ae065ebaa63f0
b815b14e3aaea5a93d95e9c4d79f40d64df0de8c6d6c03de9b28c2976d095abf
cb78428fe89a87caf81f7bcdba914f52495792b5d9bbb969b943473ecdce523f
d3cc15798e93d5bb7b27ed19961d3571835c715793c40a53acb502cec1b42beb
docker.tar
ec2d6ca9130b5d8e24d5c1645f5ff46cb70c56427779f54bf81bff3850361711
f5d4aa0fde54751d8b910b4935c89e0b5f2124e72aed396afbc275b93f3ff55f
fb5839c2038937f1ecef964372997a465b5deed8678ea0e5f2c0564c8ea900f4
fdf30caf9284f384f52c9439894931d69befbb5a6a4f5f3156b4a2d90106af00
fe215d19dded49c4cc94b29db5a9124bd769f69e9dbb92b2b061ab5fc19b08d1
manifest.json
repositories
root@ip-172-31-21-159:~/docker#
```

Now, we can view all the files and access the directories, we can begin our investigation by examining the files within the folders.
Nearly every directory contains the file “layer.tar”, so ensure you use the command “tar -xf layer.tar” to inspect all the files.

After investigating several directories, we found the file ‘SUPER_SECRET.txt’ in the directory “f5d4aa0fde54751d8b910b4935c89e0b5f2124e72aed396afbc275b93fff55f” after extracting the layer.tar file.

By using the cat command on the file we can see the answer:

“**Can You Introduce Me As Joker?”**



Writeup by: linkedin.com/in/guy-eldad/