104 – System reconfiguration

Team Information

Team Name : DogeCoin

Team Member : Dongbin Oh, Donghyun Kim, Donghyun Kim, Yeongwoong Kim

Email Address : dfc-dogecoin@naver.com

Instructions

Description Alice set up development environment for the development of the service module related with membership. Here is the virtual machine where Alice has built the development environment.

Target	Hash (HD5)
alice_Memory.dd	2CA3E96E49A429D9DD337698A72F607D
alice_virtualMachine.e01	4234cf31762f378a254b013e3354c77a
output.zip	A870C49835B94E8F6A59BCF8176DAF09

Questions

1	Identify the following information by analyzing the docker. ➢ Container ID & Name ➢ Containers' IP & Port ➢ Version of application service in container. ➢ Creation, start, and end time of container.	30
2	Among the services configured using docker, ➢ find an application that is vulnerable to the authentication. ➢ explain why you determined the app to be vulnerable.	30
3	What data could be leaked through the vulnerability? ➢ File Name ➢ Hash value (MD5)	40

Teams must:

- Develop and document the step-by-step approach used to solve this problem to allow another examiner to replicate team actions and results.
- Specify all tools used in deriving the conclusion(s).

Tools used:

Name:	Arsenal Image Mounter	Publisher:	Arsenal recon
Version:	2.0.010		
URL:	https://arsenalrecon.com/downloads/		

Name:	Winhex Portable	Publisher:	x-ways
Version:	19		
URL:	-		

Name:	Volatility	Publisher:	Volatility foundation
Version:	2.6		
URL:	https://github.com/volatilityfoundation/volatility		

Step-by-step methodology:

I. Docker 정보

1. 분석결과

[표 1] project_work_web_1 정보

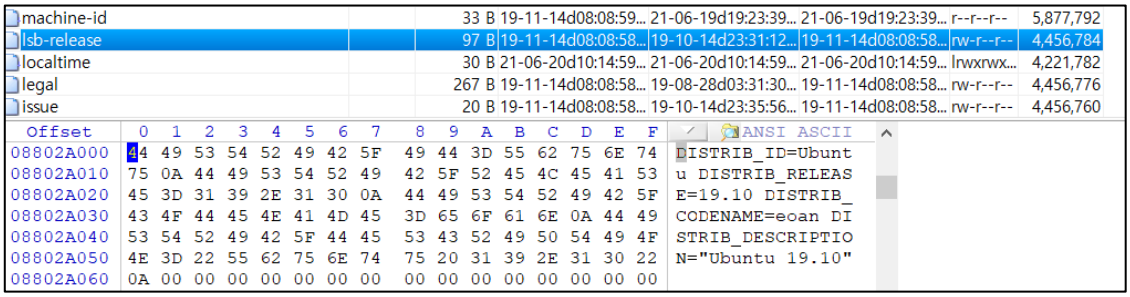
Container ID	45b102b51abe4f98af61bc56ad1eb413ae52d3cf0d2f4cd19b2d82f0295f0929
Name	project_work_web_1
IP & Port	172.18.0.3:5000
Version of application	Python 2.7.18
Creation Time (UTC)	2021-06-21T11:34:27
Start Time (UTC)	2021-06-21T11:48:56
End Time (UTC)	2021-06-22T12:02:17

[표 2] project_work_db_1 정보

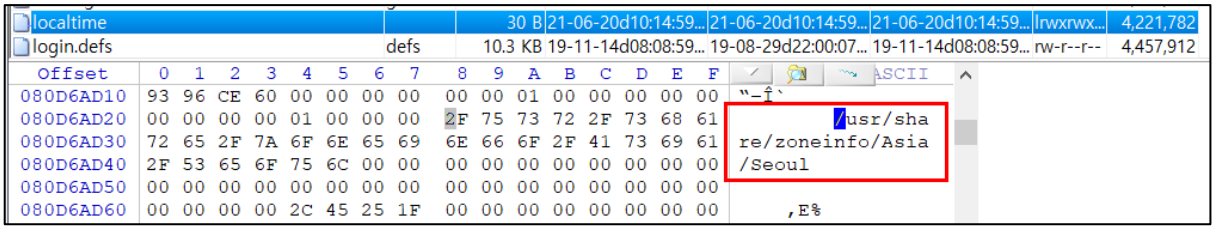
Container ID	4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c
Name	project_work_db_1
IP & Port	172.18.0.2:27017
Version of application	MongoDB 2.6.12
Creation Time (UTC)	2021-06-21T11:34:27
Start Time (UTC)	2021-06-21T11:48:55
End Time (UTC)	2021-06-22T12:02:17

2. 분석상세

alice_virtualMachine.e01 를 Arsenal Image Mounter 로 마운트 한 후 WinHex 로 분석하였다. [그림 1]과 같이, /etc/lsb-release 파일을 확인한 결과 제공된 이미지의 OS 는 Ubuntu 19.10 이다.

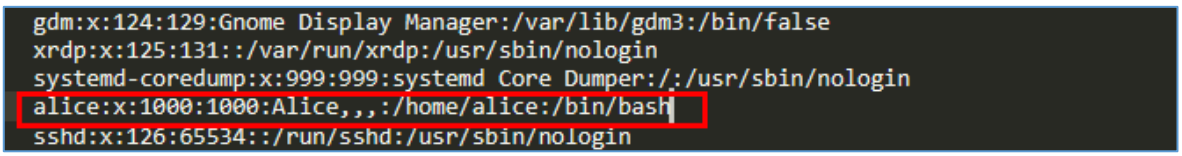


[그림 1] OS 버전 확인(/etc/lsb-release)



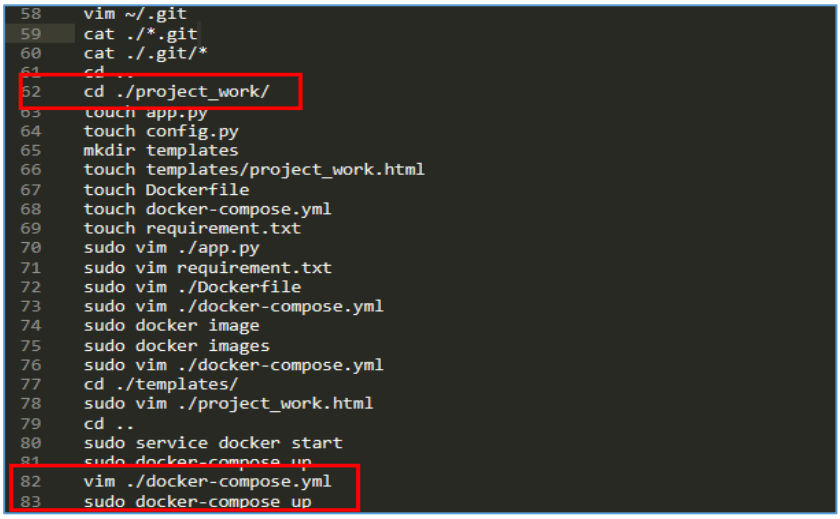
[그림 2] 타임존 확인(/etc/localtime), UTC+9

/home 디렉토리 하위에 alice 디렉토리가 존재하며, /etc/passwd 파일을 확인한 결과 alice 유저가 존재하며, bash 쉘을 기본적으로 사용함을 알 수 있었다.



[그림 3] /etc/passwd

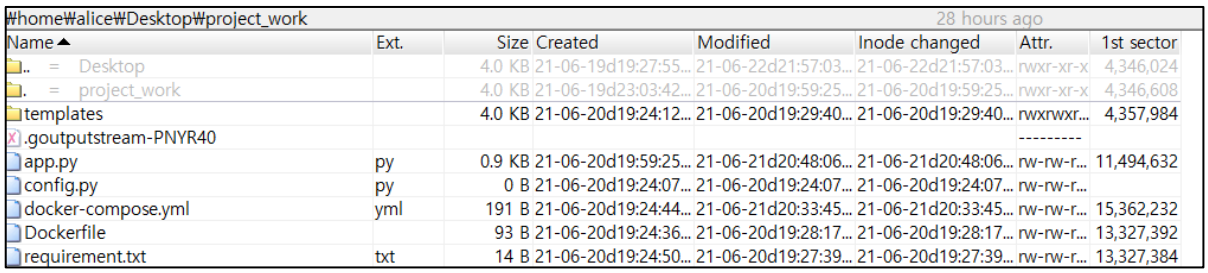
/home/alice/.bash_history 파일을 추출하여 bash 쉘 사용 기록을 분석했다.



[그림 4] .bash_history

사용자는 /home/alice/Desktop/project_work 디렉토리에서 주로 활동했다.

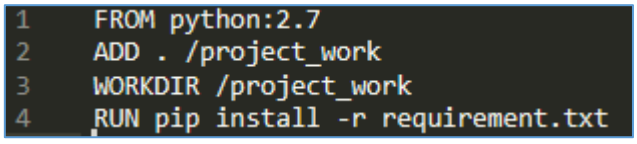
/home/alice/Desktop/project_work 디렉토리 하위에 docker-compose.yml 파일을 작성하고 docker-compose 로 이미지를 구성한 것으로 보인다.



#home¥alice¥Desktop¥project_work — 28 hours ago

Name	Ext.	Size	Created	Modified	Inode changed	Attr.	1st sector
.. = Desktop		4.0 KB	21-06-19d19:27:55...	21-06-22d21:57:03...	21-06-22d21:57:03...	rwxr-xr-x	4,346,024
. = project_work		4.0 KB	21-06-19d23:03:42...	21-06-20d19:59:25...	21-06-20d19:59:25...	rwxr-xr-x	4,346,608
templates		4.0 KB	21-06-20d19:24:12...	21-06-20d19:29:40...	21-06-20d19:29:40...	rwxrwxr...	4,357,984
.goutputstream-PNYR40						---------	
app.py	py	0.9 KB	21-06-20d19:59:25...	21-06-21d20:48:06...	21-06-21d20:48:06...	rw-rw-r...	11,494,632
config.py	py	0 B	21-06-20d19:24:07...	21-06-20d19:24:07...	21-06-20d19:24:07...	rw-rw-r...	
docker-compose.yml	yml	191 B	21-06-20d19:24:44...	21-06-21d20:33:45...	21-06-21d20:33:45...	rw-rw-r...	15,362,232
Dockerfile		93 B	21-06-20d19:24:36...	21-06-20d19:28:17...	21-06-20d19:28:17...	rw-rw-r...	13,327,392
requirement.txt	txt	14 B	21-06-20d19:24:50...	21-06-20d19:27:39...	21-06-20d19:27:39...	rw-rw-r...	13,327,384

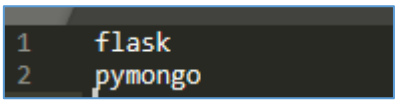
[그림 5] /home/alice/Desktop/project_work



```
1  FROM python:2.7
2  ADD . /project_work
3  WORKDIR /project_work
4  RUN pip install -r requirement.txt
```

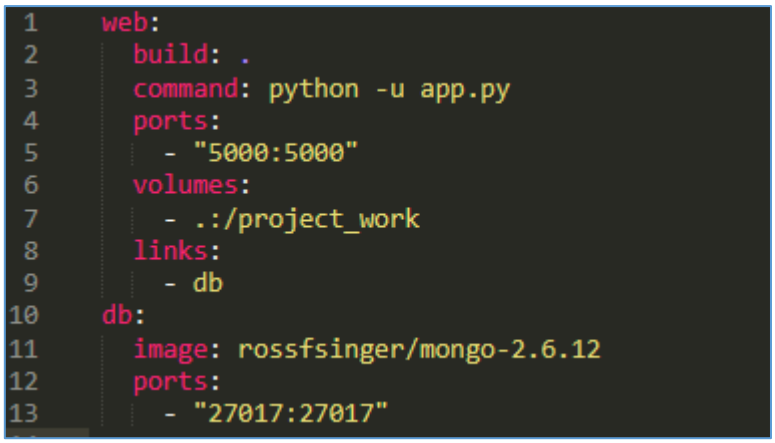
[그림 6] Dockerfile

DockerFile 분석 결과 사용자가 사용한 도커 이미지는 python 이미지이며, requirement.txt 에 명시된 파이썬 패키지(flask, pymongo)를 설치한다.



```
1  flask
2  pymongo
```

[그림 7] requirement.txt



```
1  web:
2    build: .
3    command: python -u app.py
4    ports:
5      - "5000:5000"
6    volumes:
7      - .:/project_work
8    links:
9      - db
10 db:
11   image: rossfsinger/mongo-2.6.12
12   ports:
13     - "27017:27017"
```

[그림 8] docker-compose.yml

Docker-compose.yml 에 따르면, docker-compose 시 해당 컨테이너는 python -u app.py 명령어를 실행하고, 포트는 5000 번을 사용한다. 또한 데이터베이스(db)를 연결하는데, db 이미지는 rossfsinger/mongo-2.6.12 도커 이미지를 사용하며 포트는 27017 포트를 사용한다.

```python
#-*- coding:utf-8 -*_
from flask import Flask, redirect, url_for, request, render_template
from pymongo import MongoClient
import os

app = Flask(__name__)

print os.environ['DB_PORT_27017_TCP_ADDR']
client = MongoClient(os.environ['DB_PORT_27017_TCP_ADDR'], 27017)
db = client.workspace

@app.route('/')
def todo():
    _items = db.user_info.find()
    items = [item for item in _items]
    return render_template('project_work.html', items=items)

@app.route('/new',methods=['POST'])
def new():
    item_doc = {
        #'name': request.form['name'],
        #'description': request.form['description']

        'id' : request.form['id'],
        'pw' : request.form['pw'],
        'name' : request.form['name'],
        'email' : request.form['email'],
        'person_code' : request.form['person_code'],
        'age' : request.form['age'],
        'gender' : request.form['gender']

    }
    db.workspace.insert_one(item_doc)
    return redirect(url_for('todo'))

if __name__ == "__main__":
        app.run(host='0.0.0.0', debug=True)
```

[그림 9] app.py

app.py 는 Python Web 프레임워크인 Flask 를 사용하며 Pymongo 로 MongoDB 데이터베이스를 사용하고 있다. 사용자로부터 id, pw, name, email, person_code, age, gender 를 입력받아 등록하는 문제에서 제시한 멤버십 관련 어플리케이션임을 알 수 있다.

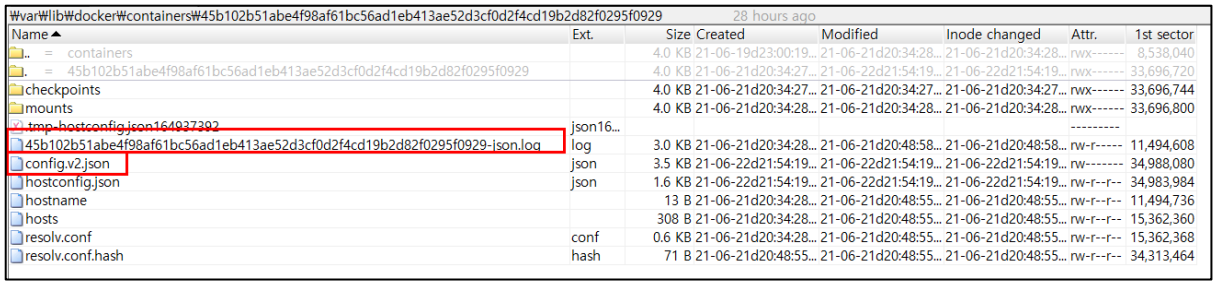
컨테이너 로그는 /var/lib/docker/containers/{Container ID} 디렉토리에 저장된다. 세개의 컨테이너 디렉토리를 확인할 수 있었다.

₩var₩lib₩docker₩containers 28 hours ago

Name	Ext.	Size	Created	Modified	Inode changed	Attr.	1st sector
.. = docker		4.0 KB	21-06-19d23:00:19...	21-06-22d21:54:19...	21-06-22d21:54:19...	rwx--x-...	8,538,008
. = containers		4.0 KB	21-06-19d23:00:19...	21-06-21d20:34:28...	21-06-21d20:34:28...	rwx------	8,538,040
3af211ff9aaccb7edadf332b6112233a15d9...		4.0 KB	21-06-21d20:32:39...	21-06-22d21:54:19...	21-06-22d21:54:19...	rwx------	33,809,408
45b102b51abe4f98af61bc56ad1eb413ae5...		4.0 KB	21-06-21d20:34:27...	21-06-22d21:54:19...	21-06-22d21:54:19...	rwx------	33,696,720
4db256d69288f47297a3b89290e7c6a03b8...		4.0 KB	21-06-21d20:34:27...	21-06-22d21:54:19...	21-06-22d21:54:19...	rwx------	33,809,680

[그림 10] /var/lib/docker/containers

디렉토리 하위에는 컨테이너 각 설정사항이 기록되는 config.v2.json 파일과 컨테이너의 실행 로그인 {containerID}-json.log 가 위치한다.



¥var¥lib¥docker¥containers¥45b102b51abe4f98af61bc56ad1eb413ae52d3cf0d2f4cd19b2d82f0295f0929 28 hours ago

Name	Ext.	Size	Created	Modified	Inode changed	Attr.	1st sector
.. = containers		4.0 KB	21-06-19d23:00:19...	21-06-21d20:34:28...	21-06-21d20:34:28...	rwx------	8,538,040
. = 45b102b51abe4f98af61bc56ad1eb413ae52d3cf0d2f4cd19b2d82f0295f0929		4.0 KB	21-06-21d20:34:27...	21-06-22d21:54:19...	21-06-22d21:54:19...	rwx------	33,696,720
checkpoints		4.0 KB	21-06-21d20:34:27...	21-06-21d20:34:27...	21-06-21d20:34:27...	rwx------	33,696,744
mounts		4.0 KB	21-06-21d20:34:28...	21-06-21d20:34:28...	21-06-21d20:34:28...	rwx------	33,696,800
.tmp-hostconfig.json164937392	json16...					--------	
45b102b51abe4f98af61bc56ad1eb413ae52d3cf0d2f4cd19b2d82f0295f0929-json.log	log	3.0 KB	21-06-21d20:34:28...	21-06-21d20:48:58...	21-06-21d20:48:58...	rw-r-----	11,494,608
config.v2.json	json	3.5 KB	21-06-22d21:54:19...	21-06-22d21:54:19...	21-06-22d21:54:19...	rw-------	34,988,080
hostconfig.json	json	1.6 KB	21-06-22d21:54:19...	21-06-22d21:54:19...	21-06-22d21:54:19...	rw-r--r--	34,983,984
hostname		13 B	21-06-21d20:34:28...	21-06-21d20:48:55...	21-06-21d20:48:55...	rw-r--r--	11,494,736
hosts		308 B	21-06-21d20:34:28...	21-06-21d20:48:55...	21-06-21d20:48:55...	rw-r--r--	15,362,360
resolv.conf	conf	0.6 KB	21-06-21d20:34:28...	21-06-21d20:48:55...	21-06-21d20:48:55...	rw-r--r--	15,362,368
resolv.conf.hash	hash	71 B	21-06-21d20:48:55...	21-06-21d20:48:55...	21-06-21d20:48:55...	rw-r--r--	34,313,464

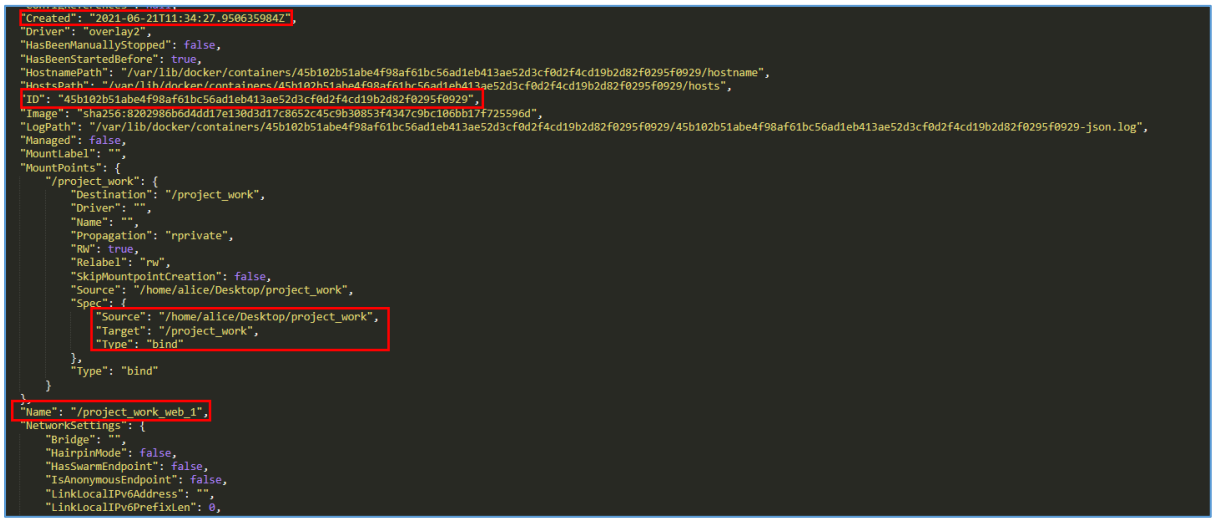
[그림 11] 컨테이너 설정 관련 파일

아래는 멤버십 어플리케이션 컨테이너인 container id 45b102b51abe4f98af61bc56ad1eb413ae52d3cf0d2f4cd19b2d82f0295f0929 의 config.v2.log 의 일부다.


```
"AppArmorProfile": "docker-default",
"Args": [
    "-u",
    "app.py"
],
"Config": {
    "AttachStderr": false,
    "AttachStdin": false,
    "AttachStdout": false,
    "Cmd": [
        "python",
        "-u",
        "app.py"
    ],
    "Domainname": "",
    "Entrypoint": null,
    "Env": [
        "PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
        "LANG=C.UTF-8",
        "PYTHONIOENCODING=UTF-8",
        "GPG_KEY=C01E1CAD5EA2C4F0B8E3571504C367C218ADD4FF",
        "PYTHON_VERSION=2.7.18",
        "PYTHON_PIP_VERSION=20.0.2",
        "PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/d59197a3c169cef378a22428a3fa99d33e080a5d/get-pip.py",
        "PYTHON_GET_PIP_SHA256=421ac1d44c0cf9730a088e337867d974b91bdce4ea2636099275071878cc189e"
    ],
    "ExposedPorts": {
        "5000/tcp": {}
    },
    "Hostname": "45b102b51abe",
    "Image": "project_work_web",
    "Labels": {
```


[그림 12] 멤버십 어플리케이션 config.v2.log 파일

python -u app.py 명령어로 실행됨과 Port 는 TCP 5000 번 포트를 사용함을 확인할 수 있다.



[그림 13] 멤버십 어플리케이션 config.v2.log 파일

시간 관련 기록은 UTC 타임존으로 기록된다. 생성시간(CreatedAt)은 2021-06-21 11:34:27 이다. Container ID(ID)는 45b102b51abe4f98af61bc56ad1eb413ae52d3cf0d2f4cd19b2d82f0295f0929 이며 이름은 project_work_web_1(Name)이다. home/alice/Desktop/project_work (Source) 디렉토리를 /project_work (Target)에 bind(Type)로 마운트했다.

[그림 14] 멤버십 어플리케이션 config.v2.log 파일

2021-06-21 11:48:56 에 시작되었으며(StartedAt) 2021-06-22 12:02:17 에 종료되었다(FinishedAt).

아래는 멤버십 어플리케이션과 연동되는 mongodb 컨테이너인 container ID 4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c 의 config.v2.log 의 일부이다.

```
"AppArmorProfile": "docker-default",
"Args": [
    "mongod"
],
"Config": {
    "AttachStderr": false,
    "AttachStdin": false,
    "AttachStdout": false,
    "Cmd": [
        "mongod"
    ],
    "Domainname": "",
    "Entrypoint": [
        "/entrypoint.sh"
    ],
    "Env": [
        "affinity:container==8f1f8b874cc4459da373479c5b7f48052d5cb8093d40aac3e83d7b09e94c8ec5",
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
        "GOSU_VERSION=1.7",
        "MONGO_VERSION=2.6.12"
    ],
    "ExposedPorts": {
        "27017/tcp": {}
    },
    "Hostname": "4db256d69288",
    "Image": "rossfsinger/mongo-2.6.12",
    "Labels": {
        "com.docker.compose.config-hash": "ea6a3823979adc692abff8bd5e1532cedd7c4560a40d98a1e6641ebeb27aa1d0",
        "com.docker.compose.container-number": "1",
        "com.docker.compose.oneoff": "False",
        "com.docker.compose.project": "project_work",
        "com.docker.compose.service": "db",
        "com.docker.compose.version": "1.21.0"
    },
```


[그림 15] Database 컨테이너 config.v2.json (이미지 정보)

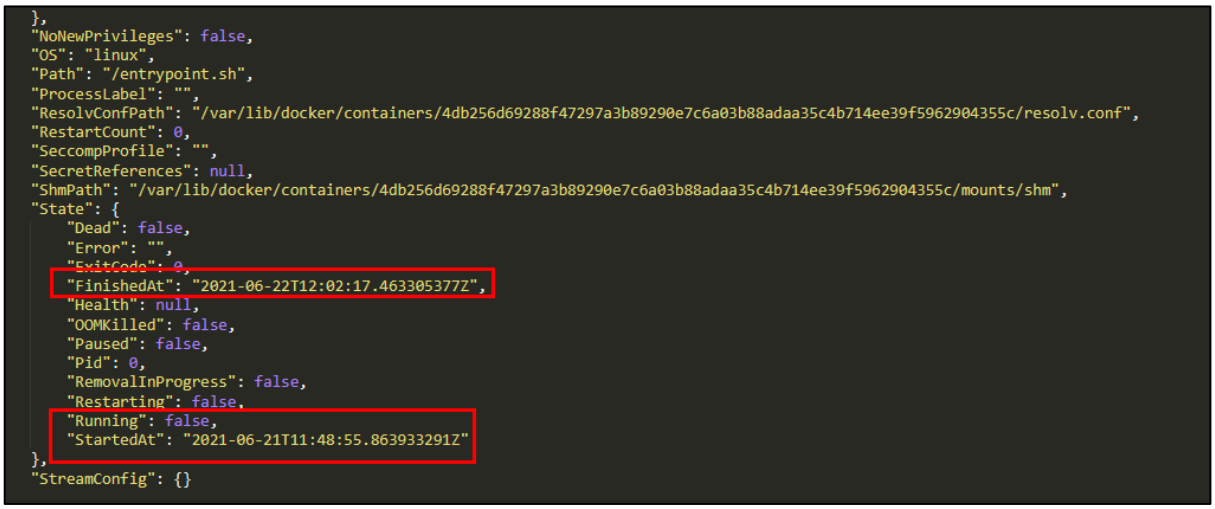

```
"Created": "2021-06-21T11:34:27.5517151522",
"Driver": "overlay2",
"HasBeenManuallyStopped": false,
"HasBeenStartedBefore": true,
"HostnamePath": "/var/lib/docker/containers/4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c/hostname",
"HostsPath": "/var/lib/docker/containers/4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c/hosts",
"ID": "4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c",
"Image": "sha256:f24b314db44911e120cccf90939d84a9fab1af36cda82cb3eaca2be688da8bbe",
"LogPath": "/var/lib/docker/containers/4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c/4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c-json.log",
"Managed": false,
"MountLabel": "",
"MountPoints": {
    "/data/configdb": {
        "Destination": "/data/configdb",
        "Driver": "local",
        "Name": "    0dde440563cbbf12096cf169819b12a350fe60cb2bb47a8477c8af8d83ff57e6",
        "RW": true,
        "Relabel": "rw",
        "SkipMountpointCreation": false,
        "Source": "/var/lib/docker/volumes/0dde440563cbbf12096cf169819b12a350fe60cb2bb47a8477c8af8d83ff57e6/_data",
        "Spec": {
            "Source": "0dde440563cbbf12096cf169819b12a350fe60cb2bb47a8477c8af8d83ff57e6",
            "Target": "/data/configdb",
            "Type": "volume"
        },
        "Type": "volume"
    },
    "/data/db": {
        "Destination": "/data/db",
        "Driver": "local",
        "Name": "fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4",
        "RW": true,
        "Relabel": "rw",
        "SkipMountpointCreation": false,
        "Source": "/var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data",
        "Spec": {
            "Source": "fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4",
            "Target": "/data/db",
            "Type": "volume"
        },
        "Type": "volume"
    }
},
"Name": "/project_work_db_1",
"NetworkSettings": {
```


[그림 16] Database 컨테이너 config.v2.json (마운트 정보)

Rossfsinger/mongo-2.6.12 이미지를 사용하고, 2021-06-21 11:34:27 에 생성되었다.

ID 는 4db256d69288f47297a3b89290e7c6a03b88adaa35c4b714ee39f5962904355c 이다.

Container 이름은 project_work_db_1 이다. volume 타입으로 로컬 디렉토리 /var/lib/docker/volumes/0dde440563cbbf12096cf169819b12a350fe60cb2bb47a8477c8af8d83ff57e6/_data 가 컨테이너의 /data/configdb 으로 마운트 되며, /var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data 가 /data/db 로 마운트 된다.



[그림 17] Database 컨테이너 config.v2.json (시작 및 종료 정보)

2021-06-21 11:48:55 에 실행되어 2021-06-22 12:02:17 에 종료되었다. 멤버십 어플리케이션 컨테이너와 동시에 실행/종료되었음을 알 수 있다.

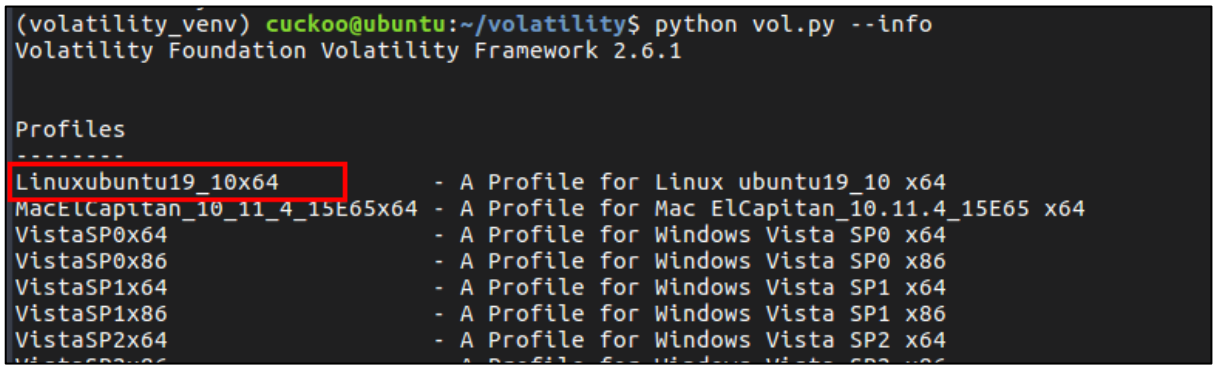
네트워크 연결 정보를 파악하기 위하여 메모리 이미지를 분석하였다. 메모리 분석 도구인 Volatility 를 사용하기 위해서는 OS 프로필이 필요한데, 현재 Volatility 는 Ubuntu 19.10 버전의 프로필을 제공하고 있지 않아 직접 생성해야 한다.

프로필 생성을 위해서는 커널 데이터 구조에 대한 명세인 module.dwarf 과 커널 심볼 정보를 담고 있는 System.map 파일이 필요하다. System.map 파일은 대부분 /boot/ 디렉토리에서 찾을 수 있다. module.dwarf 파일과 System.map 파일을 모두 획득하면 이것을 압축해야 한다. 문제에서 제공된 output.zip 파일을 이용하여 프로필을 제작했다.

[표 3] 압축에 사용한 명령어

zip ubuntu19_10.zip boot/System.map-5.3.0-64-generic module.dwarf

생성된 ubuntu19_10.zip 파일을 volatility/volatility/plugins/overlays/linux 디렉토리에 저장하면 프로필 생성은 완료된다. 다음 그림에서 확인할 수 있다.

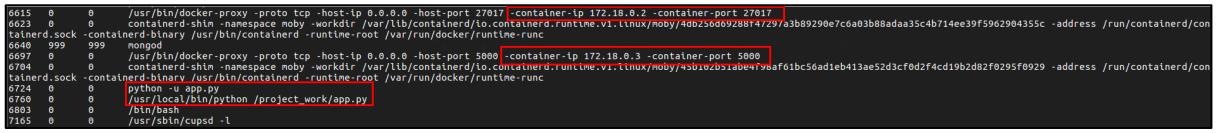


[그림 18] 설치된 프로파일 확인

아래 명령어를 사용하여 시스템 상 실행되고 있던 프로세스 목록과 실행 커맨드를 파악하였다.

[표 4] 프로세스 목록과 실행 명령어 조회에 사용한 명령어

python vol.py -f ./alice_Memory.dd --profile=Linuxubuntu19_10x64 linux_psaux



[그림 19] 실행 명령어 조회

결과에 따르면 python -u app.py 로 앱이 실행되었으며, 실행하는 동시에 db container 와 web container 가 실행되었다. 또한 container 의 ip/port 는 db 는 172.18.0.2:27017, web 은 172.18.0.3:5000 이었다.

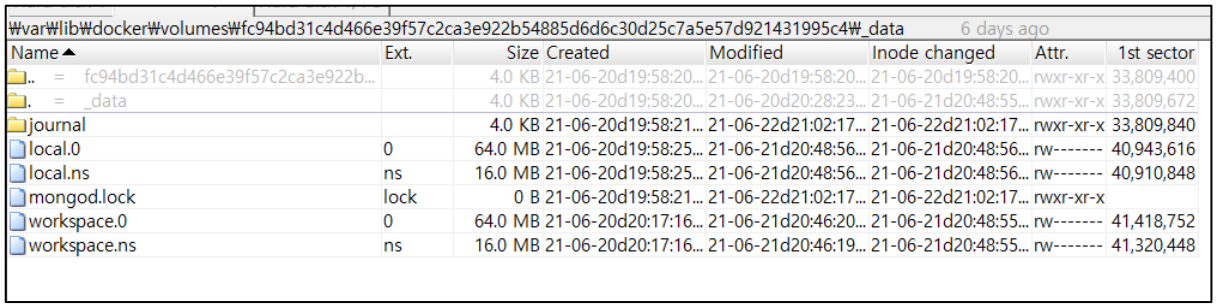
II. 인증에 취약한 SERVICE

상술했듯 container 는 python 2.7 버전에서 flask 와 pymongo 패키지, mongodb 2.6.12 버전을 사용하고 있다. 이 중 인증에 취약한 어플리케이션은 mongodb 2.6.12 로 판단된다. Mongodb 는 3.2 이하 버전에서 관리용 포트(27017)의 인증 취약점이 존재한다. 최초 서비스 구동시 인증 과정이 없으며 id 나 비밀번호 없이 관리용 포트로 db 에 접속이 가능하다. /etc/mongod.conf 를 수정해야 인증이 활성화된다. [1]

DB container 의 /data/db 폴더가 /var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data 에 마운트 된 것을 확인했었다.
/var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c

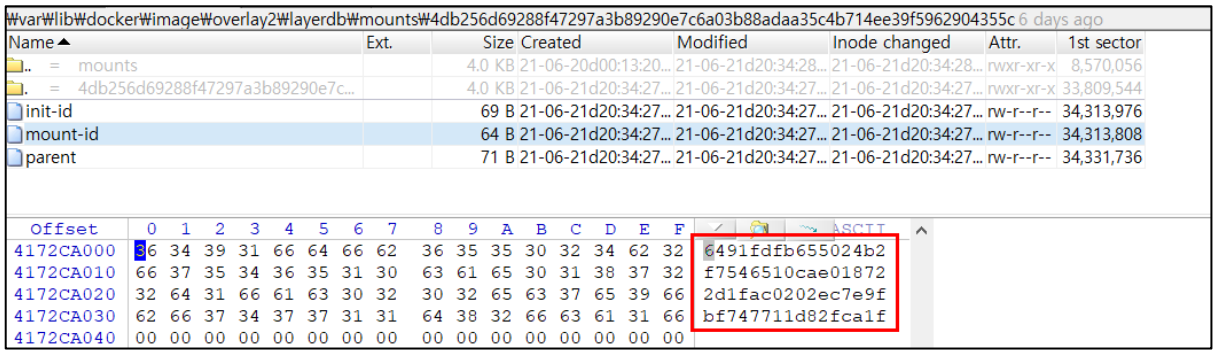
[1] https://www.krcert.or.kr/data/secNoticeView.do?bulletin_writing_sequence=25003

4/_data 경로를 분석하면 container 의 /data/db 디렉토리 내용을 알 수 있는데 admin db 가 생성되지 않았음을 알 수 있다.



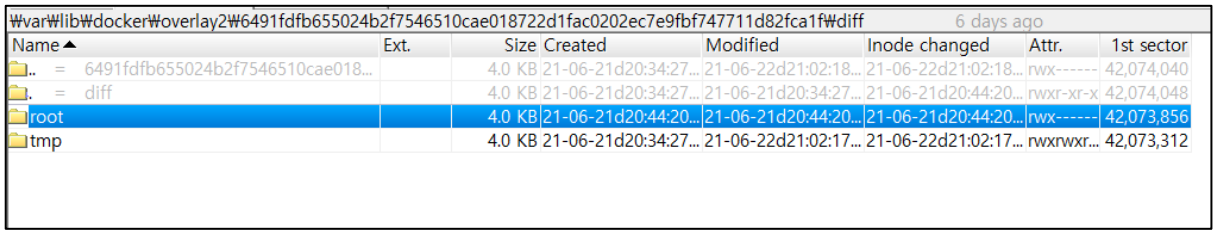
[그림 20] /var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data

db 이미지의 mount id 는 6491fdfb655024b2f7546510cae018722d1fac0202ec7e9fbf747711d82fca1f 이다.



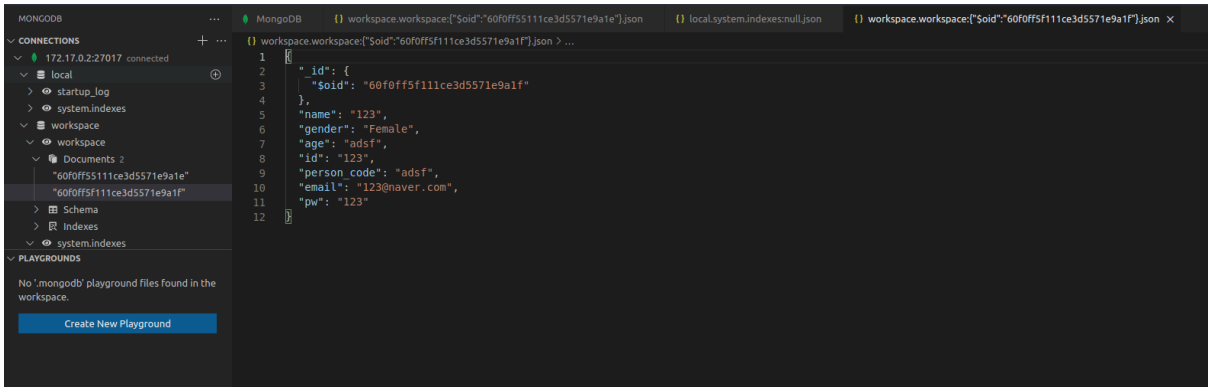
[그림 21] mount id

해당 폴더 디렉토리 조사 결과 mongodb.conf 관련 변경 흔적은 없었음을 알 수 있었다.



[그림 22] mongodb.conf 가 포함된 root 의 Modified Date

실제로 같은 이미지로 구축한 docker container 의 database 에 27017 포트로 아무런 인증없이 접속이 가능한 것을 확인하였다.



[그림 23] 환경 재구성을 통해 확인한 DB 접속

III. 유출 가능성이 있는 데이터

유출 가능한 데이터는 local 데이터베이스와 workspace 데이터베이스다. Workspace 데이터베이스가 유저들의 개인정보 데이터를 저장하고 있어 실질적인 유출 가치가 있는 데이터베이스라고 할 수 있겠다.

/var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data 에서 확인되는 2 개의 데이터베이스 정보는 다음 표와 같다.

[표 5] 데이터베이스 정보 - 1

파일명	/var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data /local.0
MD5	DF25ACEDFAEB2854769050AB38A1A396
비고	DB 메타정보

[표 6] 데이터 베이스 정보 - 2

파일명	/var/lib/docker/volumes/fc94bd31c4d466e39f57c2ca3e922b54885d6d6c30d25c7a5e57d921431995c4/_data /workspace.0
MD5	3465AD3FCFA5C03F4EAE246E168FC8DF
비고	유저 개인정보

local.0 workspace.0

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00566C80   2E 63 6F 6D 00 00 00 00 00 00 00 00 00 00 00 00  .com............
00566C90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00566CA0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00566CB0   00 01 00 00 00 20 37 00 B0 6D 56 00 B0 6B 56 00  ..... 7.°mV.°kV.
00566CC0   C5 00 00 00 07 5F 69 64 00 60 D0 7C 0C 01 AC 95  Å...._id.`Ð|..¬•
00566CD0   02 61 E0 56 97 02 69 64 00 0B 00 00 00 71 6A 79  .aàV—.id.....qjy
00566CE0   6F 75 34 37 31 31 30 00 02 70 77 00 21 00 00 00  ou47110..pw.!...
00566CF0   37 63 36 61 35 31 33 37 38 65 64 39 63 32 33 65  7c6a51378ed9c23e
00566D00   63 39 36 62 34 31 66 64 64 30 30 65 65 39 38 34  c96b41fdd00ee984
00566D10   00 02 6E 61 6D 65 00 0D 00 00 00 57 61 72 72 65  ..name.....Warre
00566D20   6E 20 42 72 6F 77 6E 00 02 73 65 78 00 05 00 00  n Brown..sex....
00566D30   00 6D 61 6C 65 00 02 61 67 65 00 03 00 00 00 32  .male..age.....2
00566D40   38 00 02 70 65 72 73 6F 6E 5F 63 6F 64 65 00 0F  8..person_code..
00566D50   00 00 00 38 32 38 38 32 31 2D 31 37 34 36 32 37  ...828821-174627
00566D60   34 00 02 65 6D 61 69 6C 00 17 00 00 00 68 63 66  4..email.....hcf
00566D70   64 66 67 6A 68 67 6C 40 68 6F 74 6D 61 69 6C 2E  dfgjhgl@hotmail.
00566D80   63 6F 6D 00 00 00 00 00 00 00 00 00 00 00 00 00  com.............
00566D90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00566DA0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00566DB0   00 01 00 00 00 20 37 00 B0 6E 56 00 B0 6C 56 00  ..... 7.°nV.°lV.
00566DC0   C7 00 00 00 07 5F 69 64 00 60 D0 7C 0C 01 AC 95  Ç...._id.`Ð|..¬•
00566DD0   02 61 E0 5A 88 02 69 64 00 0B 00 00 00 6F 73 73  .aàZˆ.id.....oss
00566DE0   6A 71 32 34 30 37 31 00 02 70 77 00 21 00 00 00  jq24071..pw.!...
00566DF0   37 33 30 38 38 34 62 33 62 35 63 39 65 34 63 31  730884b3b5c9e4c1
```

[그림 24] workspace.0 에서 확인되는 개인정보