SAMUEL CLEMOS GOMARA
IVÁN ESPINOSA MILLA

# TRABAJO FINAL SSI

# SSI - Trabajo Final

## Modalidad 2: Portado de Máquina VulnHub a Docker

Samuel Clemos Gómara e Iván Espinosa Milla

**Máquina seleccionada:**

- **Nombre:** The Planets: Mercury
- **Sistema Operativo:** Linux
- **Enlace VulnHub:** https://www.vulnhub.com/entry/the-planets-mercury,544/
- **Modalidad:** Modalidad 2 - Trabajo Complejo: Portado de Máquina VulnHub

## Objetivo

Este proyecto consiste en el portado completo de la máquina “The Planets: Mercury” de VulnHub a un entorno Docker, utilizando como base el Docker proporcionado por el profesor (Debian base). El objetivo es crear un entorno completamente funcional que replique las vulnerabilidades y el comportamiento de la máquina original, desplegable desde cero en el OVA limpio proporcionado por el profesor.

## Arquitectura del Sistema

El proyecto está compuesto por **tres contenedores Docker** orquestados mediante docker-compose:

### 1. Contenedor de Base de Datos (db)

- **Base:** Debian (Docker base del profesor)
- **Servicio:** MariaDB
- **Puerto:** 3306
- **Función:** Almacena las credenciales de usuarios y datos de “Mercury facts”

### 2. Contenedor de Servidor Web (server)

- **Base:** Debian (Docker base del profesor)
- **Framework:** Django con Python 3
- **Puerto:** 8080
- **Función:** Servidor web con aplicación Django vulnerable a SQL Injection

### 3. Contenedor SSH (ssh)

- **Base:** Debian (Docker base del profesor)
- **Servicio:** OpenSSH Server
- **Puerto:** 22

- **Función:** Acceso SSH con usuarios configurados y variables de entorno definidas que permiten una vulnerabilidad para ejecutar una bash con permisos de root.

### 4. Contenedor Pivote (pivot)

- **Base:** Debian (Docker base del profesor)
- **Servicio:** OpenSSH Server
- **Puerto:** 22
- **Función:** Pivotaje desde fuera hacia dentro de la red de los contenedores.

# Documentación de Portado

## Proceso Completo de Migración

#### Fase 1: Ruptura de la máquina original y obtención de ficheros clave

1. **Ruptura de la máquina original**

   Siguiendo los pasos de un tutorial, rompimos la máquina y sacamos los ficheros necesarios desde el propio ssh que estaba dentro del alcance del reto.

2. **Extracción de archivos relevantes:**

- Aplicación web Django completa desde /home/webmaster/mercury_proj/
- Scripts personalizados (/usr/bin/check_syslog.sh)
- Archivos de notas con credenciales (notes.txt)

#### Fase 2: Construcción del Contenedor de Base de Datos(MariaDB)

Se ha seguido el siguiente proceso para crear el docker que simula la base de datos:

1. Crear un docker basado en debian (el dado por el profesor).
2. Instalar mariadb-server.
3. Configurar la base de datos(insertar en la configuración que escuche en 0.0.0.0).
4. Crear la base de datos “mercury”.
5. Crear las tablas “facts” y “users”.
6. Insertar las filas, extraídas usando sqlmap en el proceso de ruptura de la máquina, en cada una de las tablas.
7. Insertar una flag como easter egg en la tabla “facts” que no está en la máquina original.

Todo el proceso se ha hecho “a mano”(no se ha cogido la configuración de la máquina, ya que no se ha visto necesario por la baja complejidad del proceso) para conseguir que la

base de datos funcione de forma correcta y simule la base de datos de la máquina que se está portando.

Todo el proceso está repartido entre los ficheros “dockerfile”, “prepare.sh”, “execute.sh”, “initMariaDB.sh” e “init.sql”:

- “dockerfile”: Determina la imagen del contenedor(debian), copia los ficheros necesarios del host al docker y ejecuta los scripts “prepare.sh” y “execute.sh”.
- “prepare.sh”: Instala “mariadb-server” en el docker y configura la base de datos para escuchar en “0.0.0.0”.
- “execute.sh”: Levanta el servicio “mariadb-server”, ejecuta “initMariaDB.sh” y mantiene levantado el docker.
- “initMariaDB.sh”: Inicializa la base de datos utilizando el fichero “init.sql”
- “init.sql”:
    - Crea la base de datos mercury
    - Crea las tablas users y facts
    - Inserta las credenciales originales de la máquina VulnHub

**Fase 3: Construcción del Contenedor Web**

Al romper la máquina original, los ficheros python que ejecutan el servidor web estaban disponibles, así que se decidió cogerlos directamente desde el ssh de la máquina original. Y se siguió el siguiente proceso:

1. Crear un docker basado en debian (el dado por el profesor).
2. Instalar dependencias, entre ellas Python y Django (algunas de las dependencias se descubrieron a través de errores que iban apareciendo en la ejecución del código original).
3. Se cambió la configuración del servidor python para que la conexión a la base de datos apuntara al docker creado anteriormente a través de la network creada (en el docker-compose).
4. Ejecución del servidor de python

Para ello han hecho falta los siguiente ficheros: proyecto de python original, “prepare.sh”, “execute.sh” y “dockerfile”:

- “dockerfile”: Determina la imagen del contenedor, copia el proyecto de python del host al docker, ejecuta “prepare.sh” y “execute.sh”.
- “prepare.sh”: Instala las dependencias necesarias en el docker.
- “execute.sh”: Ejecuta el servidor web y mantiene el docker levantado.

**Modificaciones en el proyecto del servidor web:**

```
# Configuración de base de datos adaptada para Docker
 DATABASES = {
 'default': {
 'ENGINE': 'django.db.backends.mysql',
```

```
'NAME': 'mercury',
'USER': 'root',
'PASSWORD': 'server',
'HOST': 'db', #Antes localhost o 127.0.0.1
'PORT': '3306',
}
}
```

No ha sido necesario hacer nada dentro del proyecto(a parte de lo anteriormente mencionado) ya que ha sido copiado entero de la máquina original(includo configuraciones).

**Fase 4: Construcción del Contenedor SSH**

Esta parte del proyecto ha sido la más costosa, debido a que se ha hecho “a mano” intentando utilizar el mínimo de pasos y sin copiar las configuraciones de la máquina original, debido a que no sabemos cuales son todos los ficheros que realmente necesitamos para hacerlo. Se ha seguido el siguiente proceso:

1. Crear un docker basado en debian (el dado por el profesor).
2. Copia de los ficheros específicos de la máquina original(“notes.txt” y “check_syslog.sh”)
3. Instalación de las dependencias
4. Creación de usuarios(iguales a los de la máquina original).
5. Configuración de permisos de root y ejecución de ficheros y carpetas para todos los usuarios (especialmente para linuxmaster, que es el usuario con el que se ejecuta todo el proceso de escalada de privilegios).
6. Inserción correcta de las flags de user y root en las carpetas correspondientes.
7. Para simular el syslog, se ha copiado un fichero hecho con logs aleatorios. Se intentó hacer con los logs del docker, pero al ser más difícil de lo esperado y al no tener ninguna relevancia en el reto, se ha decidido falsearlos.
8. Ejecución del servidor ssh.

Para ello se han utilizado los siguientes ficheros: “dockerfile”, “prepare.sh”, “execute.sh” y "start.sh”:

- “dockerfile”: Determina la imagen del contenedor, copia los diferentes ficheros en sus respectivas carpetas dentro del docker y ejecuta “prepare.sh” y “execute.sh”
- “prepare.sh”: Instala las dependencias, crea los usuarios de la máquina y los configura(especialmente, linuxmaster).
- “execute.sh”: Ejecuta el servidor ssh y lo mantiene levantado.
- “start.sh”: Crea de forma aleatoria las flags cada vez que se ejecuta el docker.

**Fase 5: Construcción del contenedor pivote**

Se ha creado un docker que se comporta como una máquina sin ningún tipo de funcionalidad más que servir de pivote entre la red interna de los dockers y el host, este docker se puede utilizar para construir túneles SSH entre el host y los dockers de las red interna. Para crearlo se ha seguido el proceso:

1. Crear un docker basado en debian (el dado por el profesor)
2. Instalación de las dependencias necesarias
3. Creación del usuario con credenciales ssi:ssi
4. Ejecución del servidor ssh

Para ello se han utilizado los siguientes ficheros: “dockerfile”, “prepare.sh” y “execute.sh”:

- “dockerfile”: Determina la imagen del contenedor y ejecuta “prepare.sh” y “execute.sh”.
- “prepare.sh”: Instala las dependencias necesarias y crea el usuario ssi:ssi.
- “execute.sh”: Ejecuta el servidor ssh y lo mentiene levantado

**Fase 6: Integración con Docker Compose**

Los 4 dockers se integran utilizando docker-compose, que permite ejecutarlos y eliminarlos todos sin necesidad de ir uno por uno.

## Componentes Modificados u Omitidos

Componentes Modificados:

1. Logs del sistema:

- Original: /var/log/syslog con entradas reales del sistema
- Modificado: Archivo syslog copiado directamente desde la máquina original
- Justificación: Complejidad excesiva para una parte irrelevante del reto

# Despliegue del Proyecto

**1. Clonar o descargar el proyecto**

git clone https://github.com/espinosa153952/SSI_TrabajoFinal
cd SSI_TrabajoFinal

**2. Instalar todas las dependencias necesarias para ejecutar docker, sqlmap...**

make init

**3. Construir los Dockers**

make main

**4. Abrir los túneles SSH (password: “ssi”)**

Terminal 1: make pivotingServer

Terminal 2: make pivotingSSH

**5. Borrar los dockers, imágenes y volúmenes**

make clean