Indice principal

Contenidos

Documentación respaldatoria

Objetivo

La presente documentación corresponde al proyecto desarrollado denominado “trabajo”.

Dicho proyecto permite ejecutar un script denominado “ejercicio.php”, el cual obtiene (dadas un conjunto de direcciones ip cargadas previamente en un archivo denominado “fichero.txt”, ubicado en el mismo path que el programa principal) las versiones de el/los protocolo/s SSL/TLS disponibles (SSLv2, SSLv3, TLS1.0, TLS1.1, TLS1.2, TLS1.3) correspondientes a las direcciones cargadas en el archivo. A su vez, al mismo tiempo que el script scanea los puertos, va cargando los resultados de este en la base de datos.

Sistema Operativo

Dicho proyecto se ha desarrollado en lenguaje multiplataforma (php), pudiendo ser ejecutado tanto sobre sistema operativo Windows como Linux. Esta producción se ha distribuido utilizando tecnología de contenedores (Docker) ejecutando el proyecto sobre Linux distribución Debian versión 10. Para que el mismo pueda ser ejecutado correctamente, se recomiendo utilizar el usuario root.

Código

El código del proyecto principal , en su versión 1.0, tal como se encuentra publicado en el repositorio Github en su version 1.0 al día de la fecha, será comentado y explicado en el anexo de la presente documentación.

Base de datos

El SGBD (Sistema de Gestion de Base de Datos) utilizada para el presente proyecto es Mysql. El mismo, al igual que el lenguaje creador es uno de los SGBD mas reconocidos por su robustez y su uso en multiplataformas.

Para poder realizar el proyecto, se optó por crear una base de datos denominada “SSLDB” con las demás características elegidas por defecto.

Para dicha base de datos, se creó una tabla denominada IPS, el cual contiene los siguientes campos:

- Indice (Clave primaria) , autoincremental, no nulo
- Ip:
    - Descripción: Campo que almacena las direcciones ip scaneadas
    - Tipo de dato: varchar
    - Longitud: 45
    - Característica adicional: no nulo
- Puerto:
    - Descripción: Campo que almacena el valor del puerto scaneado
    - Tipo de dato: Int
    - Característica adicional: no nulo
- SSLv2:

    - Descripción: Campo que almacena un valor 1 si la versión hallada del protocolo corresponde a SSLv2, caso contrario, almacena un 0 (cero)
    - Tipo de dato: tinyint
    - Característica adicional: nulo por defecto
- SSLv3:
    - Descripción: Campo que almacena un valor 1 si la versión hallada del protocolo corresponde a SSLv3, caso contrario, almacena un 0 (cero)
    - Tipo de dato: tinyint
    - Característica adicional: nulo por defecto
- TLSv1.0:
    - Descripción: Campo que almacena un valor 1 si la versión hallada del protocolo corresponde a TLSv1.0, caso contrario, almacena un 0 (cero)
    - Tipo de dato: tinyint
    - Característica adicional: nulo por defecto
- TLSv1.1:
    - Descripción: Campo que almacena un valor 1 si la versión hallada del protocolo corresponde a TLSv1.1, caso contrario, almacena un 0 (cero)
    - Tipo de dato: tinyint
    - Característica adicional: nulo por defecto
- TLSv1.2:
    - Descripción: Campo que almacena un valor 1 si la versión hallada del protocolo corresponde a TLSv1.2, caso contrario, almacena un 0 (cero)
    - Tipo de dato: tinyint
    - Característica adicional: nulo por defecto
- TLSv1.3:
    - Descripción: Campo que almacena un valor 1 si la versión hallada del protocolo corresponde a SSLv2, caso contrario, almacena un 0 (cero)
    - Tipo de dato: tinyint
    - Característica adicional: nulo por defecto
- Fecha:
    - Descripción: Campo que almacena el valor de la fecha en formato UNIX. Este formato de valor se caracteriza por ser el conteo de los días transcurridos desde el 1-1-1970, por ello el tipo de dato es bigint y no date.
    - Tipo de dato: bigint
    - Característica adicional: nulo por defecto

En caso de querer utilizar el proyecto generando la base de datos por separado (fuera del uso de Docker), se deberá importar el archivo denominado **ssldb.sql**, utilizando como ejemplo la siguiente sintaxis desde línea de comando: {mysql -u **usuario** -p**password** < ssldb.sql}

Repositorio

El presente proyecto ha sido subido a un repositorio publico denominado GITHUB, el cual ha sido utilizado como sistema de control de versiones. Dado que dicho repositorio es publico, se puede acceder a el desde cualquier navegador tipeando la siguiente url: https://github.com/ccaraccio/trabajo . Desde dicha url se puede descargar el proyecto completo, pudiendo ver y modificar su código ,si así se lo desea.

Contenedor

El presente proyecto ha sido desarrollado utilizando tecnología de contenedores para poder realizar una distribución rápida, sencilla y desatendida. Para esto se utilizó “Docker” (www.docker.com).

Para poder realizar el proyecto, se procedió a crear un archivo denominado **Dockerfile,** como se detalla a continuación:

1. Se creó una carpeta denominada ”dockerweb” (en mi caso) dentro de mi carpeta personal “c:\Users\carlos\dockerweb”
2. Una vez creada la carpeta, se procedió a crear un archivo denominado Dockerfile (sin extensión) .Este archivo contiene las instrucciones necesarias para automatizar el despliegue de un contenedor como ser: Sistemas Operativo a utilizar, software de base de datos a descargar e instalar, librerías de php, entre otros. A continuación se describen las instrucciones guardadas en el Dockerfile:
    A. from debian:latest //Indica que se va a descargar la ultima versión de Debian
    B. MAINTAINER CARLOS carloscaraccio@gmail.com //Indica quien es responsable de mantener el codigo
    C. RUN apt-get update //Ejecuta una actualización del repositorio
    D. RUN apt-get -y install wget lsb-release gnupg //Instala las librerias wget, lsb-release y gnupg
    E. RUN apt-get -y install php //Instala el interprete de lenguaje PHP
    F. RUN wget https://dev.mysql.com/get/mysql-apt-config_0.8.16-1_all.deb //Descarga el archivo.DEB indicado para actualizar el archivo sources.list , donde se obtienen los respositorios. En este caso puntual indica el repositorio para descargar Mysql
    G. RUN wget https://dev.mysql.com/get/Downloads/MySQL-8.0/mysql-community-server-core_8.0.23-1debian10_amd64.deb //Descarga el archivo deb indicado para permitir la instalación del core del mysql-community-server
    H. RUN apt-get install libaio1 libmecab2 libnuma1 // Permite descarga e instalar librerías requeridas.
    I. RUN dpkg -i mysql-apt-config_0.8.16-1_all.deb // Instala las lineas de APT para poder instalar mysql
    J. RUN dpkg -i mysql-community-server-core_8.0.23-1debian10_amd64.deb //Instala el core del motor mysql-community-server
    K. RUN apt-get update //Ejecuta una nueva actualización de repositorio
    L. expose 3306 //Abre el Puerto tcp 3306 para permitir la conexioni a mysql

3. Finalizada la edición del archivo, se procede a guardar los cambios, salir de la edición y luego ejecutar el comando : **Docker build -t nombre path.**
   En mi caso sería: Docker build -t web c:\users\carlos\dockerweb
4. Una vez finalizado el proceso anterior, el Docker ha sido creado, pudiendo consultar la imagen tipeando el comando: **Docker images**

Anexo

En dicha sección se copia el código php , explicando detalladamente cada una de las líneas que lo componen :

```
<?php
//Variables de conexión a la base de datos
$usuario='fpuser';
$clave='xxxxx';
$db='ssldb';
$host='localhost';
$link=mysqli_connect($host,$usuario,$clave,$db);
if (!$link)
{
        echo "Error, no se pudo conectar a la Base de Datos";
        exit;
}
// Abrir archivo “fichero.txt” para lectura
$fp=fopen("./fichero.txt","r");
//Leer archivo hasta el final
while(!feof($fp)){
        //Defino las variables
        $SSLv2=0;$SSLv3=0;$TLSv10=0;$TLSv11=0;$TLSv12=0;$TLSv13=0;
        //$ip: variable donde almaceno la ip que obtengo del archivo
        $ip=fgets($fp);
        if (strlen($ip)==0)
                exit;
        echo "\nLa ip a escanear es: ".$ip;
        //Ejecuto el archivo script.sh, pasandole como parametro la ip que saco del fichero.txt
        $comando="./script.sh ". $ip;
        echo "\nEl comando a ejecutar es: ".$comando;
        $retval=shell_exec($comando);
        //Separo los protocolos devueltos
        $protocolos=explode("|",$retval);
```

```
foreach ($protocolos as $key=>$value){
        //El primer pipe no contiene valor
        if ($key>0){
                echo "[".$key."]"." valor: ".trim($value);
                $valor=trim($value);
                echo "Valor vale: ".$valor."\n";
                switch($valor)
                {
                        case 'SSLv2:':
                                $SSLv2=1;
                                break;
                        case  'SSLv3:':
                                $SSLv3=1;
                                break;
                        case 'TLSv1.0:':
                                $TLSv10=1;
                                break;
                        case 'TLSv1.1:':
                                $TLSv11=1;
                                break;
                        case 'TLSv1.2:':
                                $TLSv12=1;
                                break;
                        case 'TLSv1.3:':
                                $TLSv13=1;
                                break;
                }
        }
}
echo "Insertando datos en la DB...\n";
$ip=trim($ip);
```

```
        $sql="Insert into ips
values(null,'$ip',443,$SSLv2,$SSLv3,$TLSv10,$TLSv11,$TLSv12,$TLSv13,unix_timestamp())";
        echo "Comando ejecutado:\n "."\t".$sql."\n";
        $resultado=mysqli_query($link,$sql);
        echo "resultado del query: ".$resultado."\n";
}
fclose($fp); //Se cierra el archivo abierto para lectura

?>
```

Autor:

Carlos Alejandro Caraccio