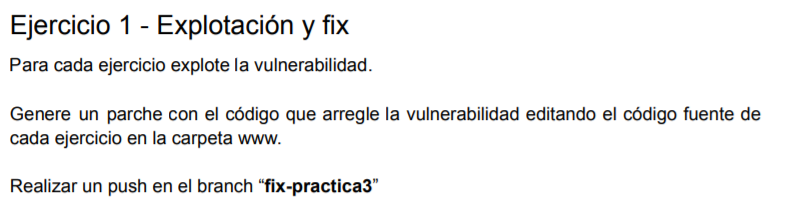
**DSA - Práctica 3**

Alejo Alfredo Santi y Fermín Moreno

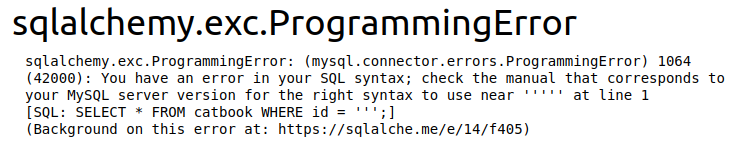


## Reto 1

Vulnerabilidad explotada: Inyección SQL Union Operator

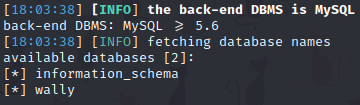
La página es vulnerable a ataques SQL, lo comprobamos utilizando el caracter “‘“ en la consulta SQL.



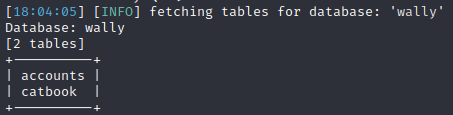


En el error que devuelve la página determinamos que la página hace una consulta con el motor MySQL de la forma SELECT \* FROM catbook WHERE id = ‘’, y deja entre las comillas lo que nosotros ingresamos en el input.

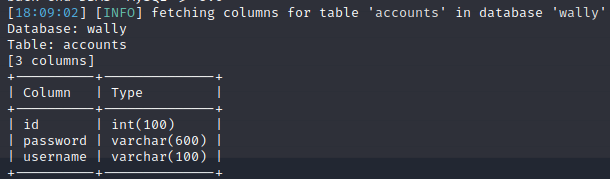
Por lo tanto, el próximo paso es conocer la estructura de la base de datos, lo cual pudimos obtenerla con SQLMap:



Listamos las tablas de la db wally:



Listamos las columnas de la tabla account:



Y ahora agregamos a la consulta una unión para obtener los datos de la tabla accounts:

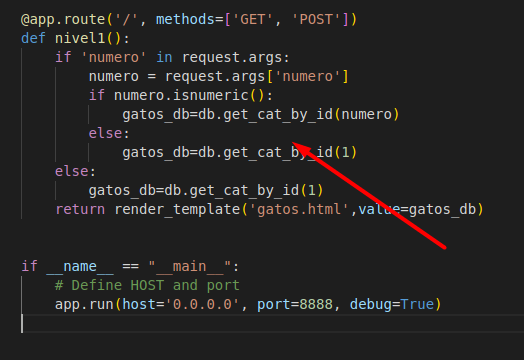
1' UNION ALL SELECT id, username, password, null, "a" from accounts where id= '1=1



Como resultado obtenemos un listado en donde el segundo elemento tiene los datos de la tabla “accounts”.

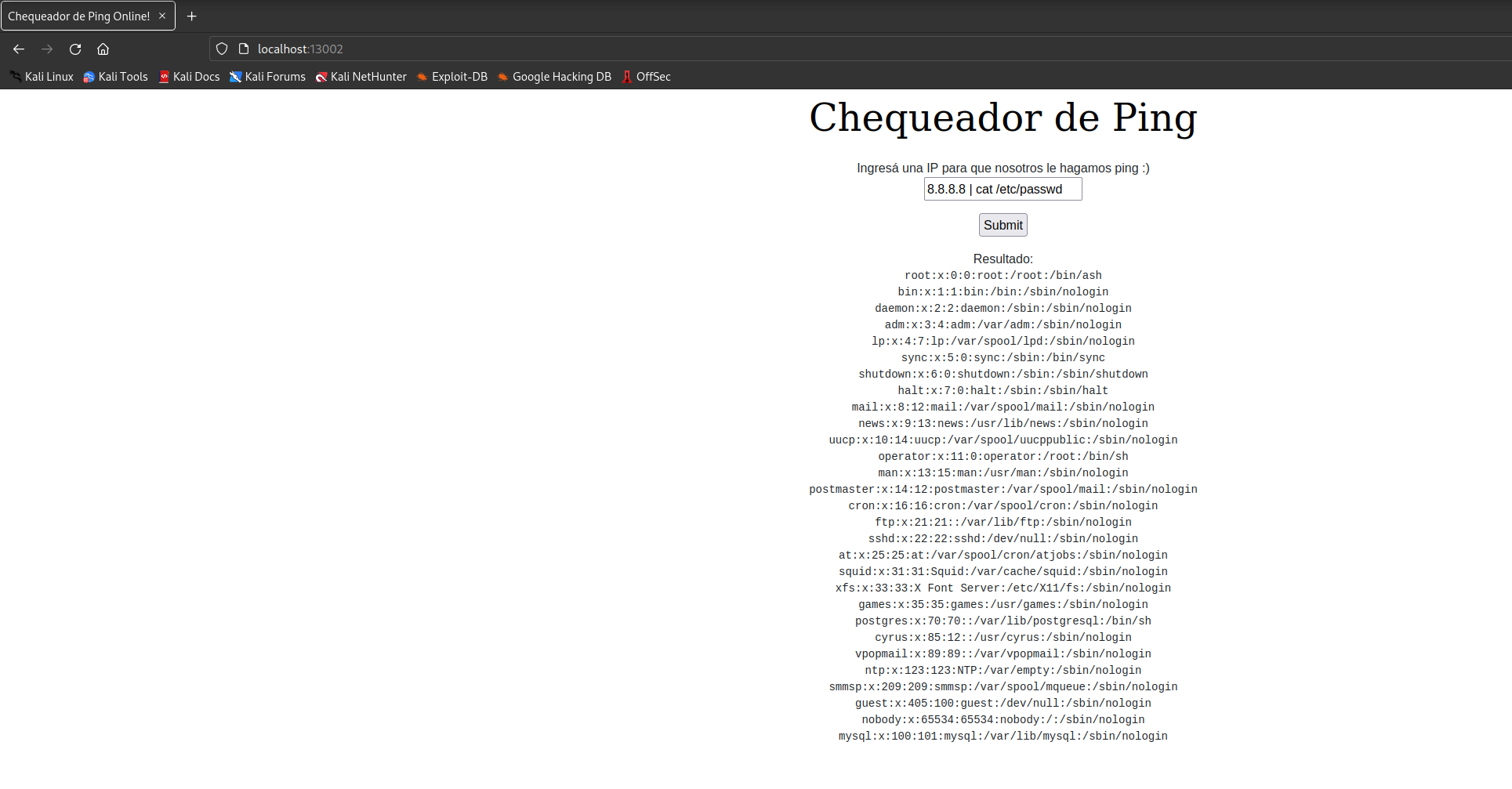
### Solución

Para solucionar el problema, debemos sanear el contenido del input. En este caso utilizamos, dentro del código, la función isnumeric() de Python. De esta manera, si el input es un número, se realiza la consulta SQL deseada, sin dar la opción al usuario de inyectar otra cosa.



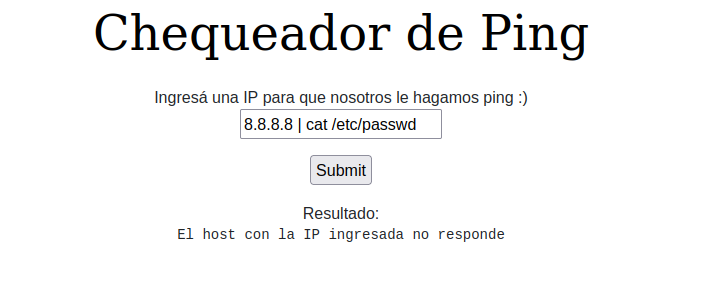
## Reto 2

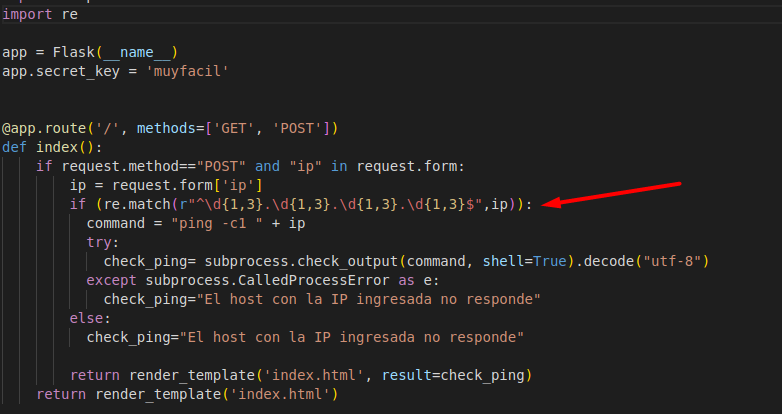
En esta página se nos permite ingresar la ip a donde el servidor va a realizar un ping, por lo que se nos ocurrió que el servidor simplemente pegaba la ip en el bash luego del comando ping, lo que generaría una vulnerabilidad, ya que se le puede agregar un comando usando un pipe y utilizar el bash. En este caso listamos los usuarios del sistema:



### Solución

En este caso, para sanear el input, utilizamos una expresión regular que chequee que el input tenga la forma (1 a 3 dígitos).(1 a 3 dígitos).(1 a 3 dígitos).(1 a 3 dígitos). De esta manera, si el input no tiene una IPv4 el chequeador de ping va a responder que el host no responde.

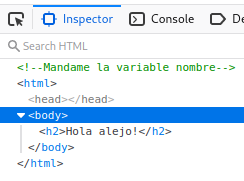




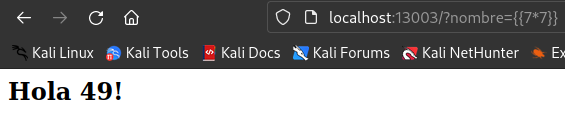
## 

## Reto 3

Esta página nos permite pasar por parámetro un texto en el pedido get, el cual da como respuesta un html con el parámetro escrito.



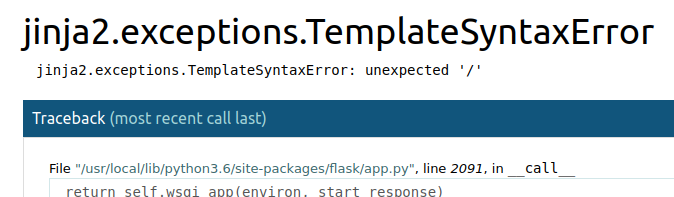
Por lo tanto nos dió la pista de que está utilizando un template en donde pega lo que llega como parámetro, por lo que probamos pasando por parámetro el valor {{7\*7}}:



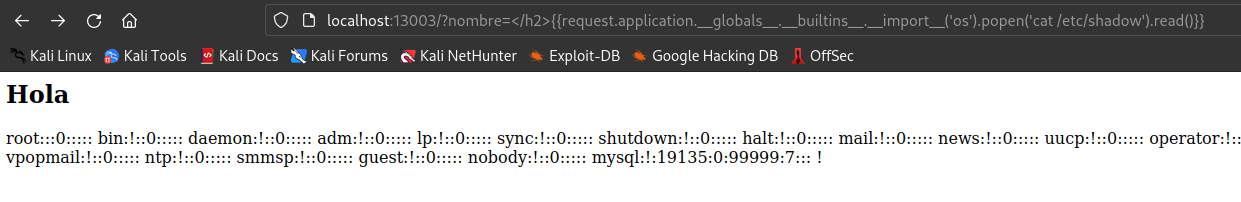
Comprobamos que efectivamente se hace la cuenta, lo que significa que al parámetro lo está procesando un motor de plantillas sin hacer ningún tipo de comprobación, esto representa una vulnerabilidad, a continuación explicamos cómo la explotamos.

Tratamos de generar un error en el motor de plantillas para saber cual se estaba utilizando

Probamos con [http://localhost:13003/?nombre={{/](http://localhost:13003/?nombre=%7B%7B/)}}



Obtuvimos un error que nos hace saber que se está utilizando jinja2, por lo que probamos el siguiente comando de jinja para ejecutar:



El comando es equivalente al siguiente código Python:

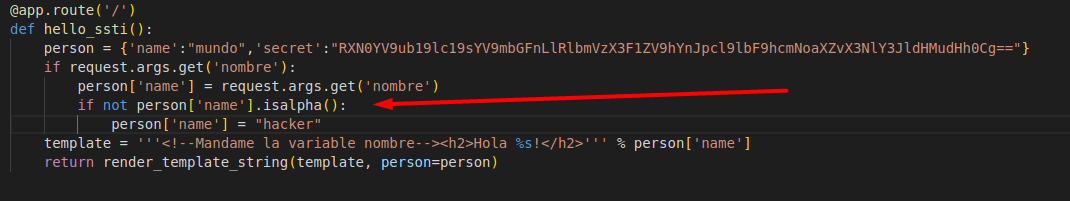
import ‘os’

os.popen(“cat /etc/shadow”)

Por lo que obtenemos un listado de los usuarios.

### Solución

Para solucionarlo saneamos el input utilizando la función isalpha() de Python.



De esta manera, si lo que ingresa el usuario son solo letras, se utiliza en el template, sino lo que ingresa se descarta para evitar inyecciones.

