Documentación Paso a Paso: Detección de Ataques SQLi con Machine Learning, ModSecurity y Flask

Creador: Osmel Pillot Leyva 👑
Fecha: 01/04/2025 🗓️

Este documento explica cómo implementar un sistema de detección de inyecciones SQL (SQLi) utilizando:

- **Debian 12** como sistema operativo.
- **Apache + ModSecurity** como WAF (Web Application Firewall).
- **Flask + scikit-learn** para el modelo de Machine Learning.
- **DVWA (Damn Vulnerable Web App)** para pruebas.

---

1. Instalación del Entorno Base

1.1 Instalar Debian 12 y Actualizar

```
sudo apt update && sudo apt upgrade -y
```

1.2 Instalar Apache y PHP (para DVWA)

```
sudo apt install apache2 php php-mysql libapache2-mod-php -y
```

1.3 Instalar ModSecurity para Apache

```
sudo apt install libapache2-mod-security2 -y
sudo cp /etc/modsecurity/modsecurity.conf-recommended
/etc/modsecurity/modsecurity.conf
sudo sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/'
/etc/modsecurity/modsecurity.conf
sudo systemctl restart apache2
```

---

2. Configurar DVWA para Pruebas

2.1 Descargar e Instalar DVWA

```
cd /var/www/html
sudo git clone https://github.com/digininja/DVWA.git
sudo chown -R www-data:www-data DVWA/
sudo cp DVWA/config/config.inc.php.dist DVWA/config/config.inc.php
```

- Editar `config.inc.php` y configurar la base de datos:

```
$_DVWA['db_user'] = 'dvwa';
$_DVWA['db_password'] = 'p@ssw0rd';
$_DVWA['db_database'] = 'dvwa';
```

- Configurar permisos:

```
sudo mysql -e "CREATE DATABASE dvwa; CREATE USER 'dvwa'@'localhost' IDENTIFIED
BY 'p@ssw0rd'; GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost'; FLUSH
PRIVILEGES;"
```

- Acceder a DVWA en:

```
http://localhost/DVWA/setup.php
```

- Hacer clic en **"Create / Reset Database"** para inicializar DVWA.
- Credenciales **user**: admin , **password**: password

---

3. Entorno Python para el Modelo de ML

3.1 Instalar Python y Virtualenv

```
sudo apt install python3 python3-pip python3-venv -y
mkdir /opt/sql_detector && cd /opt/sql_detector
python3 -m venv venv
source venv/bin/activate
```

3.2 Instalar Dependencias (Flask, scikit-learn, joblib)

```
pip install flask scikit-learn joblib
```

3.3 Guardar el Modelo Entrenado

```
mkdir /var/www/sql_detector
```

- Guardar el modelo y vectorizador en la carpeta **sql_detector**

---

4. Aplicación Flask para Detección de SQLi

4.1 Crear `app.py`

```python
from flask import Flask, request, jsonify
import joblib
import json

app = Flask(__name__)

model = joblib.load('modelo_entrenado.pkl')
vectorizer = joblib.load('vectorizador.pkl')


def extraer_y_preprocesar(solicitud):
    if not solicitud or not isinstance(solicitud, dict):
        return []
    parametros = solicitud["parameters"]
    valores_filtrados = [v for k, v in parametros.items() if k not in ['ARGS:Submit','ARGS:Token']]
    #print(valores_filtrados)
    return valores_filtrados

@app.route('/predecir', methods=['POST'])
def predecir():
    solicitud = request.get_json()
    print(solicitud)
    data = extraer_y_preprocesar(solicitud)
    print(data)
    # Preprocesar y vectorizar la solicitud
    texto = ' '.join(data).lower()
    X_new = vectorizer.transform([texto])
    print(X_new)
    #texto = ' '.join(data).lower()

    # Realizar la predicción
    prediccion = model.predict(X_new)[0]

    # Devolver la predicción
    return 'Maliciosa' if prediccion == 1 else 'Legítima'
    #return "Hola"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

- Para correr el servidor , ejecutar:

```
python3 app.py
```

---

5. Integración con ModSecurity

5.1 Configurar ModSecurity para Redirigir a Flask

- Editar `/etc/modsecurity/modsecurity.conf` y agregarle reglas propias:

```
SecRuleEngine On
Include /etc/modsecurity/rules/sqli_rules.conf
```

5.2 Regla para Bloquear SQLi Detectados

```
nano /etc/modsecurity/rules/sqli_rules.conf
```

```
SecRule ARGS "@rx .*"\
        "id:1000,\
        phase:2,\
        t:none,\
        deny,\
        status:403,\
        msg:'Ataque SQLi detectado por ML',\
        chain"
        SecRuleScript "/etc/modsecurity/lua/script.lua"
```

5.3 Crear Script en Lua para redireccionar a Flask

```lua
function main()
    local request_data = {
        method = m.getvar("REQUEST_METHOD") or "NO_METHOD",
        ip = m.getvar("REMOTE_ADDR") or "NO_IP"
    }

    local all_args = m.getvars("ARGS", {"none"})
    local params = {}
    for _, arg in ipairs(all_args) do
        params[arg["name"]] = arg["value"]
    end

    local payload = {
        request= request_data,
        parameters = params
    }

    local json = require("dkjson")
    local json_payload = json.encode(payload)

    local flask_url = "http://localhost:5000/predecir"
    local http = require("socket.http")
    local ltn12 = require("ltn12")

    local response_body = {}
    local res, code, response_headers = http.request{
        url= flask_url,
        method= "POST",
        headers = {
            ["Content-Type"] = "application/json",
            ["Content-Length"] = #json_payload
        },
        source = ltn12.source.string(json_payload),
        sink = ltn12.sink.table(response_body)
    }

    m.log(1, "Enviando JSON a Flask" .. json_payload)
    if code == 200 then
        m.log(1, "Datos enviados correctamente a Flask")
        local response_text = table.concat(response_body)
        m.log(1, "Respuesta de Flask: " .. response_text)
        if response_text == "Maliciosa" then
            m.log(1,"Bloqueando solicitud....")

            m.setvar("tx.outbound_anomaly_score", "100")
            m.setvar("tx.anomaly_score","100")
            m.setvar("tx.error_message","Acceso denegado por politica de seguridad")
            m.setvar("tx.status","403")
            return true
        else
            m.log("Permitiendo Solicitud")
            return false
        end
    else
        m.log(1, "Error al enviar datos a Flask. Codigo: " .. tostring(code))
    end

    return false
end
```

5.4 Reiniciar Apache

```
sudo systemctl restart apache2
```

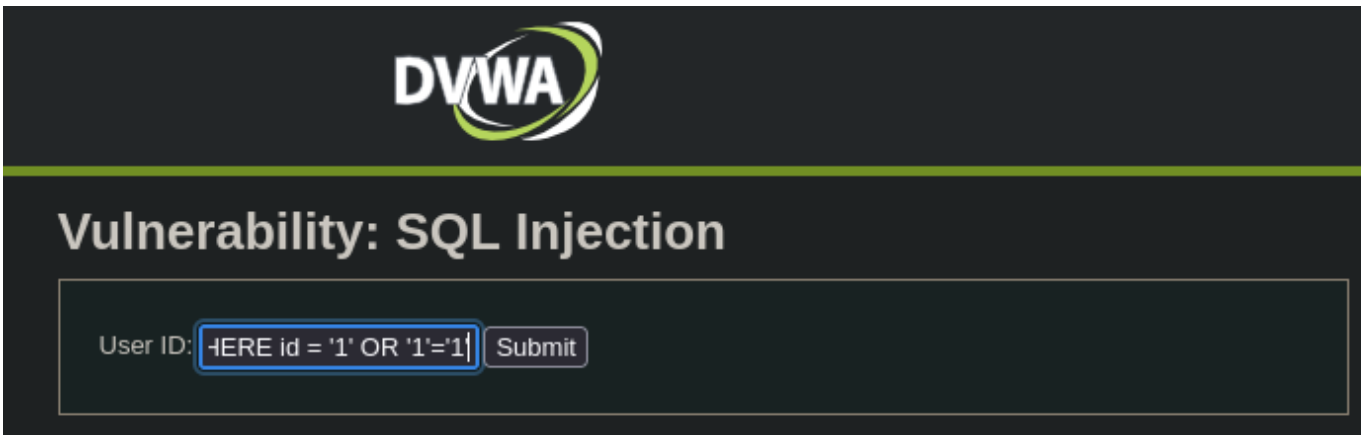
---

6. Pruebas Finales

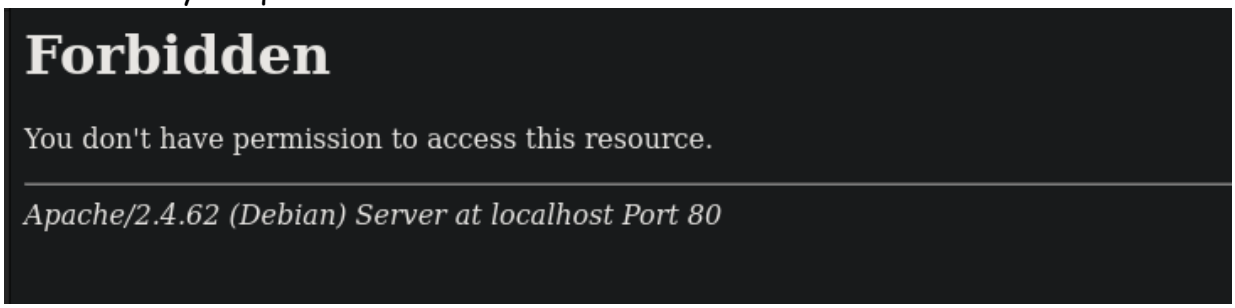
6.1 Realizar Ataques SQLi en DVWA

- Ir a **DVWA > SQL Injection** e intentar:

> **Código de ejemplo**
> SELECT * FROM dvwa WHERE id = '1' OR '1'='1'



- **Resultado esperado:**
  - ModSecurity bloqueará la solicitud.



6.2 Ver Logs de ModSecurity

```
tail -f /var/log/apache2/modsec_audit.log
```

```
Message: Enviando JSON a Flask{"request":{"method":"GET","ip":"127.0.0.1"},"parameters":{"ARGS:user_toke
n":"eda82d82f391de48faff17aa94d57353","ARGS:Submit":"Submit","ARGS:id":"SELECT * FROM dvwa  WHERE id = '
1' OR '1'='1'"}}
Message: Datos enviados correctamente a Flask
Message: Respuesta de Flask: Maliciosa
Message: Bloqueando solicitud....
```