

Simulación Ataque de Sistemas y Defensa con DevSecOps
I Cuatrimestre 2025

Araya Porras Diego, Facultad de Ingeniería informática
Arce Barquero Oscar Daniel, Facultad de Ingeniería informática
Ortega Fallas Jonnathan, Facultad de Ingeniería informática
Quesada Solano Dannel, Facultad de Ingeniería informática
Rodríguez Barboza Mario Josué, Facultad de Ingeniería informática
Leytón Chávez Lilliam Gissela, Facultad de Ingeniería informática
Universidad Politécnica Internacional

Organización de Archivos

Mata Guerrero Jose Javier

Abril de 2025

Contenido

Explicación del Tema Investigado.....	3
¿Qué es DevSecOps?.....	3
Componentes clave de DevSecOps	3
Integración continua	3
Entrega Continua	3
Seguridad Continua	4
Comunicación y Colaboración	4
Herramientas Utilizadas y su Propósito.....	4
Docker	4
DVWA	4
Trivy	4
GitHub Actions.....	5
Git/GitHub	5
Dificultades Encontradas en el Proyecto DevSecOps	5
Configuración Inicial de DVWA en Docker.....	5
Ejecución de SQL Injection.....	5
Integración de Trivy con GitHub Actions.....	6
Pasos Detallados de la Implementación	7
Pruebas.....	14
Referencias	17

Explicación del Tema Investigado

¿Qué es DevSecOps?

DevSecOps, que significa desarrollo, seguridad y operaciones, es un marco que integra la seguridad en todas las fases del ciclo de vida de desarrollo de software. Las organizaciones adoptan este enfoque para reducir el riesgo de publicar código con vulnerabilidades de seguridad. A través de la colaboración, la automatización y los procesos claros, los equipos comparten la responsabilidad de la seguridad, en lugar de dejarla al final cuando los problemas pueden ser mucho más difíciles y costosos de abordar. DevSecOps es un componente fundamental de una estrategia de seguridad multinube.

Es importante la implementación de DevSecOps ya que existen muchos métodos que los atacantes usan para obtener acceso a los datos y recursos de una organización, pero una táctica común es aprovechar las vulnerabilidades de software. Estos tipos de vulneraciones son costosas, consumen mucho tiempo y dependen de la gravedad, lo que afecta a la reputación de una empresa. El marco DevSecOps reduce el riesgo de implementar software con configuraciones incorrectas y otras vulnerabilidades que los actores malintencionados pueden aprovechar.

Componentes clave de DevSecOps

Integración continua

Con la integración continua, los desarrolladores confirman su código en un repositorio central varias veces al día. A continuación, el código se integra y prueba automáticamente. Este enfoque permite a los equipos detectar problemas de integración y errores al principio del proceso

Entrega Continua

Se basa en la integración continua para automatizar el proceso de mover código desde el entorno de compilación a un entorno de ensayo. Una vez en el ensayo, además de las pruebas unitarias, el software se prueba automáticamente para garantizar que la interfaz de usuario funciona

Seguridad Continua

La creación de seguridad en todo el ciclo de vida de desarrollo de software es un componente clave de DevSecOps. Esto incluye el modelado de amenazas al principio del proceso y las pruebas de seguridad automatizadas a lo largo de todo el ciclo de vida, empezando por los propios entornos de los desarrolladores

Comunicación y Colaboración

DevSecOps depende en gran medida de las personas y los equipos que trabajan en estrecha colaboración. La integración continua requiere que las personas colaboren para solucionar conflictos en el código, y los equipos necesitan comunicarse de forma eficaz para unificarse en torno a los mismos objetivos.

Herramientas Utilizadas y su Propósito

Docker

- Crear un entorno aislado y reproducible para desplegar aplicaciones vulnerables de forma segura
- Permitir la ejecución de DVWA sin afectar el sistema anfitrión
- Facilitar el escaneo de vulnerabilidades en imágenes dentro de contenedores

DVWA

- Aplicación web intencionalmente vulnerable para fines educativos o de hacking ético
- Demostrar vulnerabilidades reales (SQL Injection) en un entorno controlado
- Servir como banco de pruebas para técnicas de seguridad ofensivas y defensivas

Trivy

- Escáner de vulnerabilidades open-source para imágenes Docker
- Identificar paquetes vulnerables en las dependencias
- Detectar configuraciones inseguras en containers
- Integrarse en pipelines CI/CD para seguridad continua

GitHub Actions

- Automatizar el escaneo de seguridad en cada push al repositorio
- Garantizar que los chequeos de seguridad sean parte del flujo de desarrollo
- Proporcionar retroalimentación inmediata sobre vulnerabilidades

Git/GitHub

- Control de versiones para el código de infraestructura (Infrastructure as Code)
- Plataforma colaborativa para el proyecto
- Alojamiento del pipeline de seguridad (GitHub Actions)

Dificultades Encontradas en el Proyecto DevSecOps

Configuración Inicial de DVWA en Docker

Problema:

- La imagen vulnerable/web-dvwa requería configuración manual después del despliegue (creación/reseteo de base de datos)
- Problemas de conexión con la base de datos MySQL interna en algunos intentos iniciales

Solución:

- Documentar cuidadosamente los pasos de inicialización (login → reset DB → nuevo login)
- Verificar que el contenedor tuviera suficientes recursos asignados

Ejecución de SQL Injection

Problema:

- La inyección SQL básica (1' OR '1'=1) no funcionaba inicialmente en ciertas configuraciones de DVWA
- El nivel de seguridad de la aplicación afectaba la efectividad del ataque

Solución:

- Asegurarse de configurar DVWA en "low security mode"

- Probar diferentes payloads de SQL Injection hasta encontrar el efectivo

Integración de Trivy con GitHub Actions

Problema:

- La instalación manual de Trivy en el workflow era lenta y propensa a errores
- Los resultados del scan no se mostraban de forma clara en GitHub

Solución:

- Cambiar a la acción oficial de Trivy (aquasecurity/trivy-action)
- Configurar formato de salida más legible (format: 'table')

Pasos Detallados de la Implementación

1. Crear la web Vulnerable utilizando Docker desde PowerShell
docker run -d -p 8080:80 --name dvwa vulnerables/web-dvwa

```
Windows PowerShell
PS C:\Users\oscar> docker run -d -p 8080:80 --name dvwa vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
098cffd43466: Pull complete
b3d64a33242d: Pull complete
3e17c6eae66c: Pull complete
e9968e5981d2: Pull complete
6cff5f35147f: Pull complete
2cd72dba8257: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
07a103b8bf482f92727fd6f2c86391baa70d42f91b0fc5f0011748e6ead88757
PS C:\Users\oscar> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                NAMES
07a103b8bf48   vulnerables/web-dvwa  "/main.sh"              11 seconds ago Up 11 seconds  0.0.0.0:8080->80/tcp  dvwa
PS C:\Users\oscar>
```

2. Dirigirse al navegador web e ingresar a localhost:8080



Username

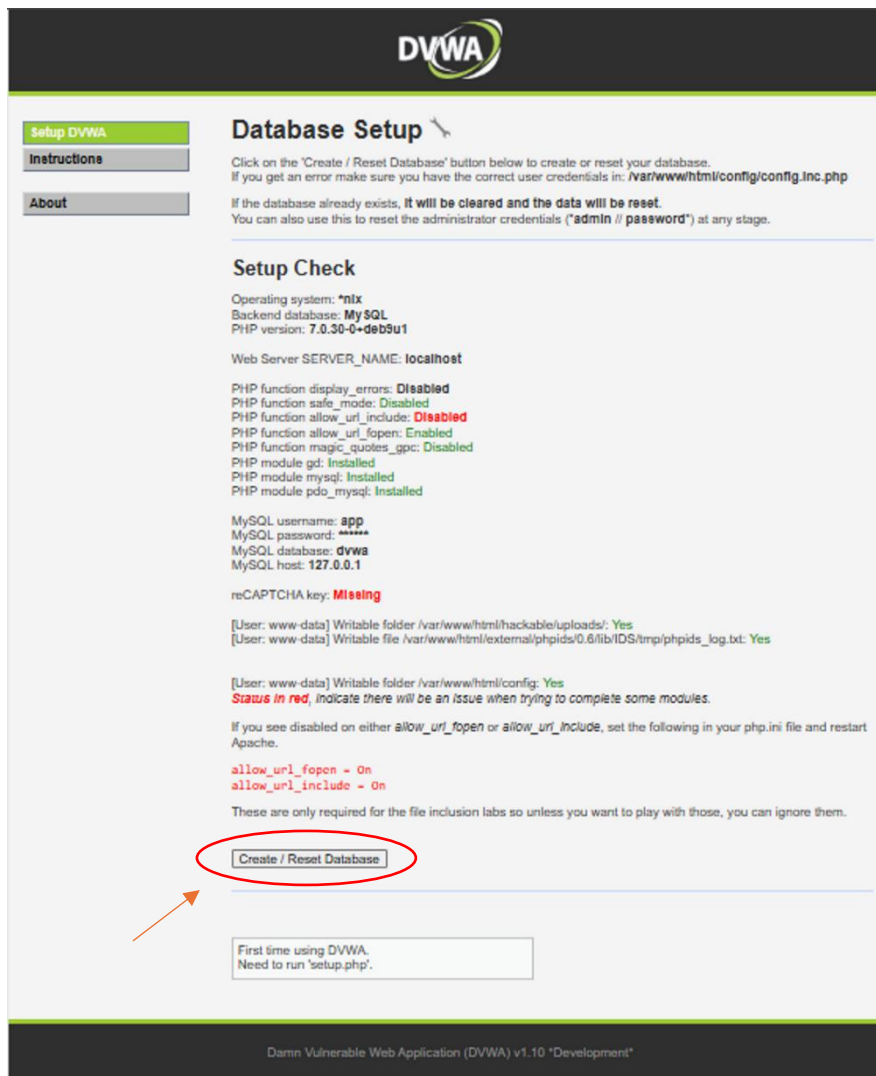
Password

You have logged out

[Damn Vulnerable Web Application \(DVWA\)](#)

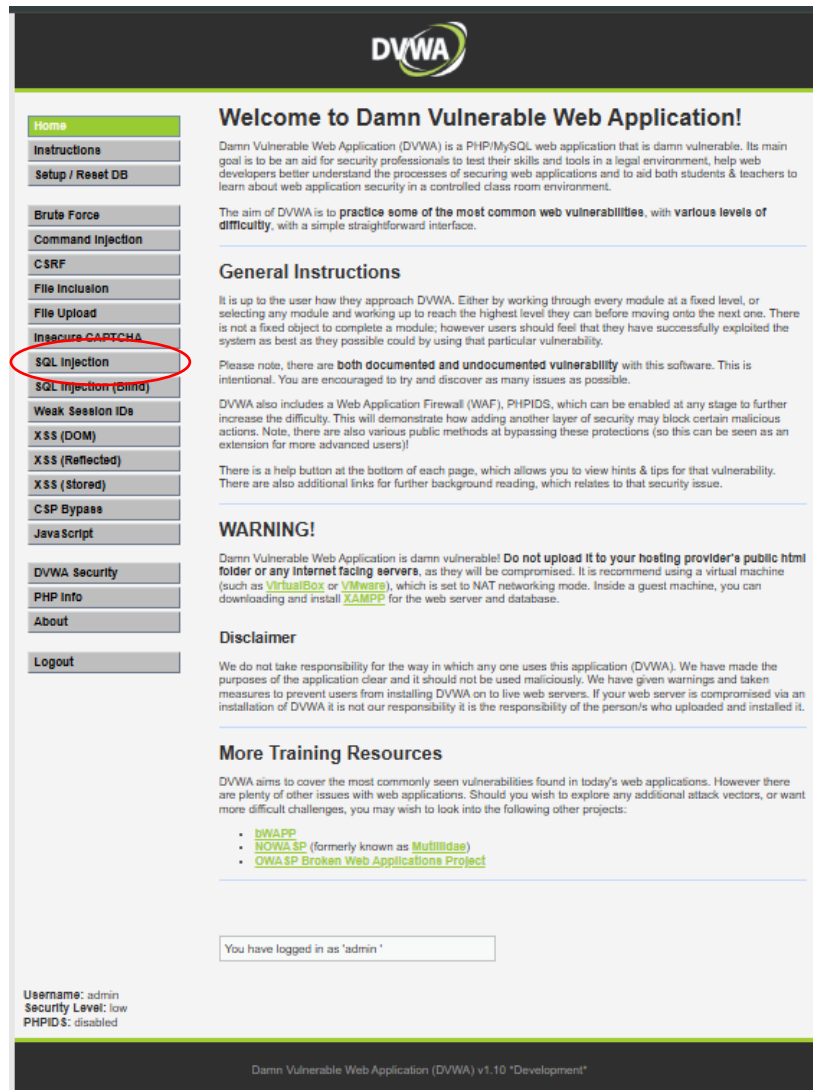
3. Ingresar a la pag de DVWA con las siguientes credenciales
Username: admin
Password: password

- Una vez adentro de DVWA en la parte inferior seleccionar “Create/Reset Database”



- Esperar unos segundos a que la pag nos redirija nuevamente al login de la misma, y volvemos a ingresar con las mismas credenciales que se utilizaron anteriormente

6. Una vez dentro nuevamente, se despliega un menú al lado izquierdo de la pag de DVWA, se selecciona la opción que dice “SQL Injection”



DVWA

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficulty**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module, however users should feel that they have successfully exploited the system as best as they possibly could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommended using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- [BWAPP](#)
- [NOWASP](#) (formerly known as [Mutillidae](#))
- [OWASP Broken Web Applications Project](#)

You have logged in as 'admin'

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 "Development"

7. En la barra de búsqueda Ingresar el siguiente comando: 1' OR '1'=1
y click en submit (esto confunde a la parte de SQL de la pag haciendo que filtre el ID
de varios usuarios a la vez)

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Username: admin
Security Level: low
PHPIDS: disabled

Vulnerability: SQL Injection

User ID:

ID: 1' OR '1'=1
First name: admin
Surname: admin

ID: 1' OR '1'=1
First name: Gordon
Surname: Brown

ID: 1' OR '1'=1
First name: Hack
Surname: Me

ID: 1' OR '1'=1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'=1
First name: Bob
Surname: Smith

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_injection
- <http://bobby-tables.com/>

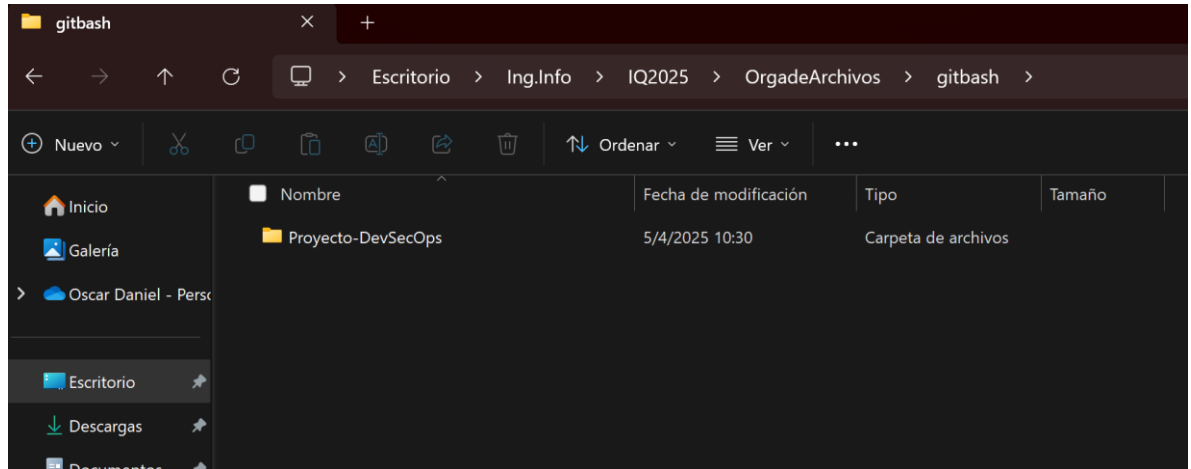
Damn Vulnerable Web Application (DVWA) v1.10 "Development"

8. Ejecutar Trivy usando Docker desde PowerShell (esto despliega un reporte de vulnerabilidades de la pag DVWA)
docker run --rm aquasec/trivy image --severity CRITICAL vulnerables/web-dvwa

9. Crear repositorio en GitHub con el nombre Proyecto-DevSecOps y clonarlo de forma local en la dirección que se desee

git clone <https://github.com/tuusuario/Proyecto-DevSecOps.git>

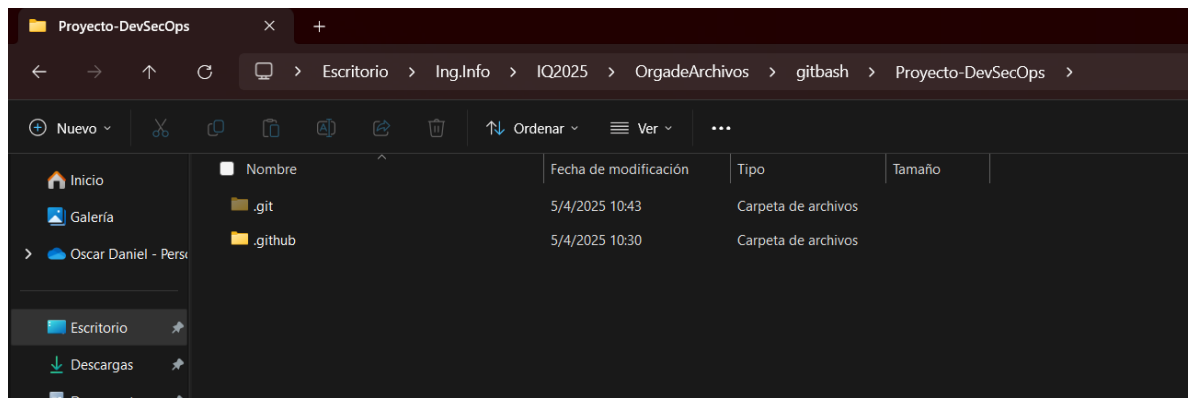
cd Proyecto-DevSecOps



10. Dentro de ese repositorio que acaba de ser clonado, se crean los siguientes archivos

mkdir -p .github/workflows

touch .github/workflows/scan.yml



11. Una vez que se tengan esos 2 archivos dentro del repositorio local, se ingresa a `github>workflows>scan` y se abre el editor de texto, y se pega este código

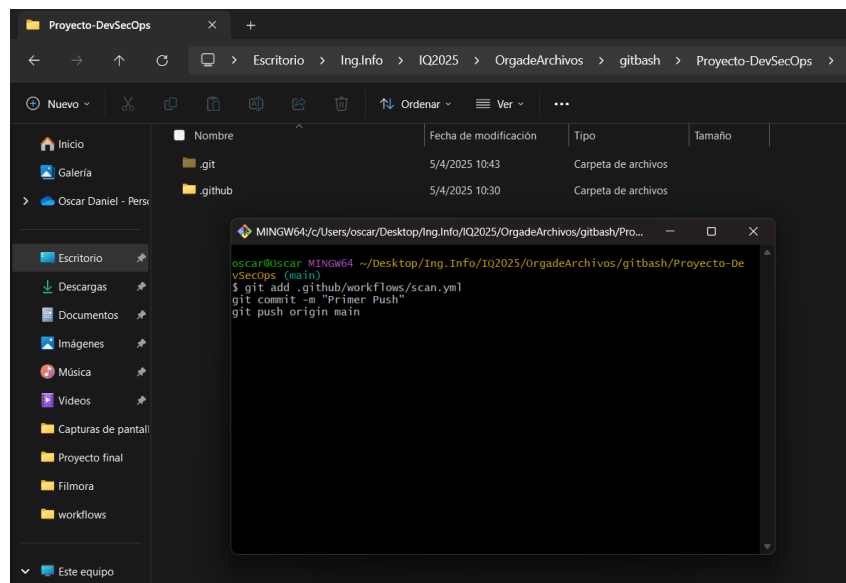
```
name: Docker Image Security Scan
on: [push] # Se ejecuta al hacer git push

jobs:
  scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

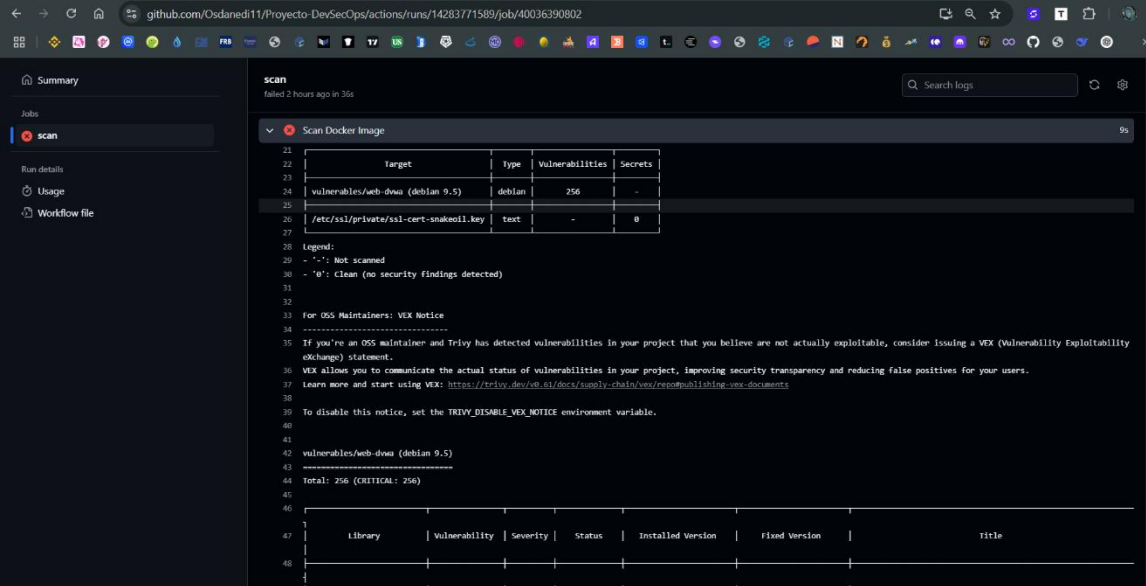
      - name: Install Trivy
        run: |
          sudo apt-get update
          sudo apt-get install -y wget apt-transport-https gnupg lsb-release
          wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
          echo "deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a
          /etc/apt/sources.list.d/trivy.list
          sudo apt-get update
          sudo apt-get install -y trivy

      - name: Scan Docker Image
        run: |
          trivy image --severity CRITICAL --exit-code 1 vulnerables/web-dwva
```

12. Se guardan los cambios, nos dirigimos a la misma ruta del paso 10, y realiza un push
- ```
git add .github/workflows/scan.yml
git commit -m "Primer Push"
git push origin main
```



13. Una vez hecho todo esto, vamos a nuestro repositorio de GitHub, lo refrescamos, y nos dirigimos a actions>workflow en ejecución o completado (hay que esperar a que termine), scan>scan docker image y nos muestra el reporte



The screenshot shows a GitHub Actions workflow run for the 'scan' job. The workflow is titled 'Scan Docker Image' and has a duration of 9s. The main content is a Trivy scan report for the image 'vulnerables/web-dmaw (debian 9.5)'. The report includes a table of findings and a summary of the scan results.

| Target                                 | Type   | Vulnerabilities | Secrets |
|----------------------------------------|--------|-----------------|---------|
| vulnerables/web-dmaw (debian 9.5)      | debian | 256             | -       |
| /etc/ssl/private/ssl-cert-snakeoil.key | text   | -               | 0       |

Legend:  
- '-': Not scanned  
- '0': Clean (no security findings detected)

For OSS Maintainers: VEX Notice  
-----  
If you're an OSS maintainer and Trivy has detected vulnerabilities in your project that you believe are not actually exploitable, consider issuing a VEX (Vulnerability Exchange) statement.  
VEX allows you to communicate the actual status of vulnerabilities in your project, improving security transparency and reducing false positives for your users.  
Learn more and start using VEX: <https://trivy.dev/v0.61/docs/supply-chain/vex/republishing-vex-documents>  
To disable this notice, set the TRIVY\_DISABLE\_VEX\_NOTICE environment variable.

vulnerables/web-dmaw (debian 9.5)  
-----  
Total: 256 (CRITICAL: 256)

| Library   | Vulnerability  | Severity | Status      | Installed Version | Fixed Version | Title                                                      |
|-----------|----------------|----------|-------------|-------------------|---------------|------------------------------------------------------------|
| libssl1.1 | CVE-2023-38408 | CRITICAL | Not scanned | 1.1.1f-1          | 1.1.1f-1      | libssl1.1: Denial of Service (DoS) via crafted certificate |

## Pruebas

En nuestro archivo scan que es donde se hacen las pruebas cuando queramos hacer un push se pueden agregar los siguientes códigos (uno a la vez):

✓256

2

```
name: Docker Image Security Scan
on: [push]

jobs:
 scan:
 runs-on: ubuntu-latest
 steps:
 - name: Checkout code
 uses: actions/checkout@v4

 - name: Scan and Generate Report
 run: |
 docker run --rm aquasec/trivy image \
 --severity CRITICAL \
 --exit-code 1 \
 --format table \
 vulnerables/web-dvwa | tee trivy-report.txt
 echo "Vulnerabilidades encontradas (workflow configurado para fallar)"

 - name: Upload Report
 if: always()
 uses: actions/upload-artifact@v4 # ¡Versión corregida!
 with:
 name: trivy-results
 path: trivy-report.txt
```

(este cuenta con un workflow exitoso, pero solo tiene la capacidad de mostrar 256 vulnerabilidades porque solo muestra las de severidad CRITICAL)

Fuerza fallo en el Workflow (situacion real)

```
- name: Scan Docker Image
run: |
 docker run --rm aquasec/trivy image \
 --severity HIGH,CRITICAL \
 --exit-code 1 \ # Fuerza fallo si hay vulnerabilidades
 vulnerables/web-dvwa
```

X  
No Scan

(Este no cuenta con scan, solamente es un workflow fallido, simplemente es para probar que el trivy que se instaló ya es compatible a bloquear o mostrar fallidos los workflows)

```
name: Docker Image Security Scan
on: [push]

jobs:
 scan:
 runs-on: ubuntu-latest
 steps:
 - name: Checkout code
 uses: actions/checkout@v4

 - name: Scan Docker Image
 run: |
 docker run --rm aquasec/trivy image --severity
 CRITICAL,HIGH vulnerables/web-dvwa
```

✓  
805

(Este muestra un workflow exitoso, ya que no tiene la línea de código “--exit-code 1”, y muestra 805 vulnerabilidades porque además de mostrar las de severidad CRITICAL, también toma en cuenta las de severidad HIGH )

```
name: Docker Image Security Scan
on: [push]

jobs:
 scan:
 runs-on: ubuntu-latest
 steps:
 - name: Checkout code
 uses: actions/checkout@v4

 - name: Scan Docker Image
 run: |
 docker run --rm aquasec/trivy image \
 --severity CRITICAL,HIGH \
 --exit-code 1 \
 --format table \
 vulnerables/web-dywa | tee trivy-report.txt
 echo "Vulnerabilidades detectadas (workflow configurado para fallar)"
 exit 1 # Fuerza el fallo

 - name: Upload Report
 if: always()
 uses: actions/upload-artifact@v4
 with:
 name: trivy-results
 path: trivy-report.txt
```

X  
805

(Muestra un workflow fallido, cuenta con “--exit-code 2”, pero además muestra el scan y con 805 vulnerabilidades porque toma en cuenta las de severidad CRITICAL y HIGH)



## Referencias

- Moudabbes, M. K. C. (s/f). DevSecOps: Automatización Empresarial. Softtek.com. Recuperado el 5 de abril de 2025, de <https://blog.softtek.com/es/por-qu%C3%A9-las-empresas-deber%C3%ADan-elegir-devsecops-para-automatizar-sus-procesos>
- ¿Qué es DevSecOps? (s/f-a). Redhat.com. Recuperado el 5 de abril de 2025, de <https://www.redhat.com/es/topics/devops/what-is-devsecops>
- ¿Qué es DevSecOps? (s/f-b). Microsoft.com. Recuperado el 5 de abril de 2025, de <https://www.microsoft.com/es-mx/security/business/security-101/what-is-devsecops>