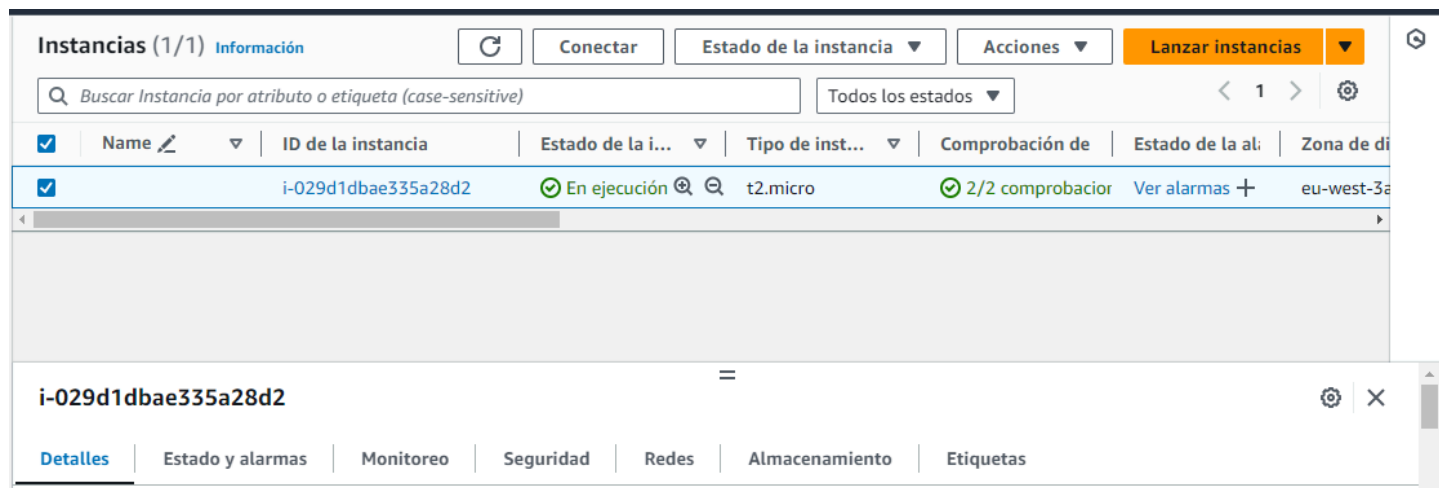


Práctica Desarrollo de Software Seguro DVWA con GitLab CI

Por: Mario Cantero Shimizu

Entorno de Despliegue

Para el despliegue he decidido usar AWS y hostearlo en una instancia EC2 con sistema Ubuntu Server.

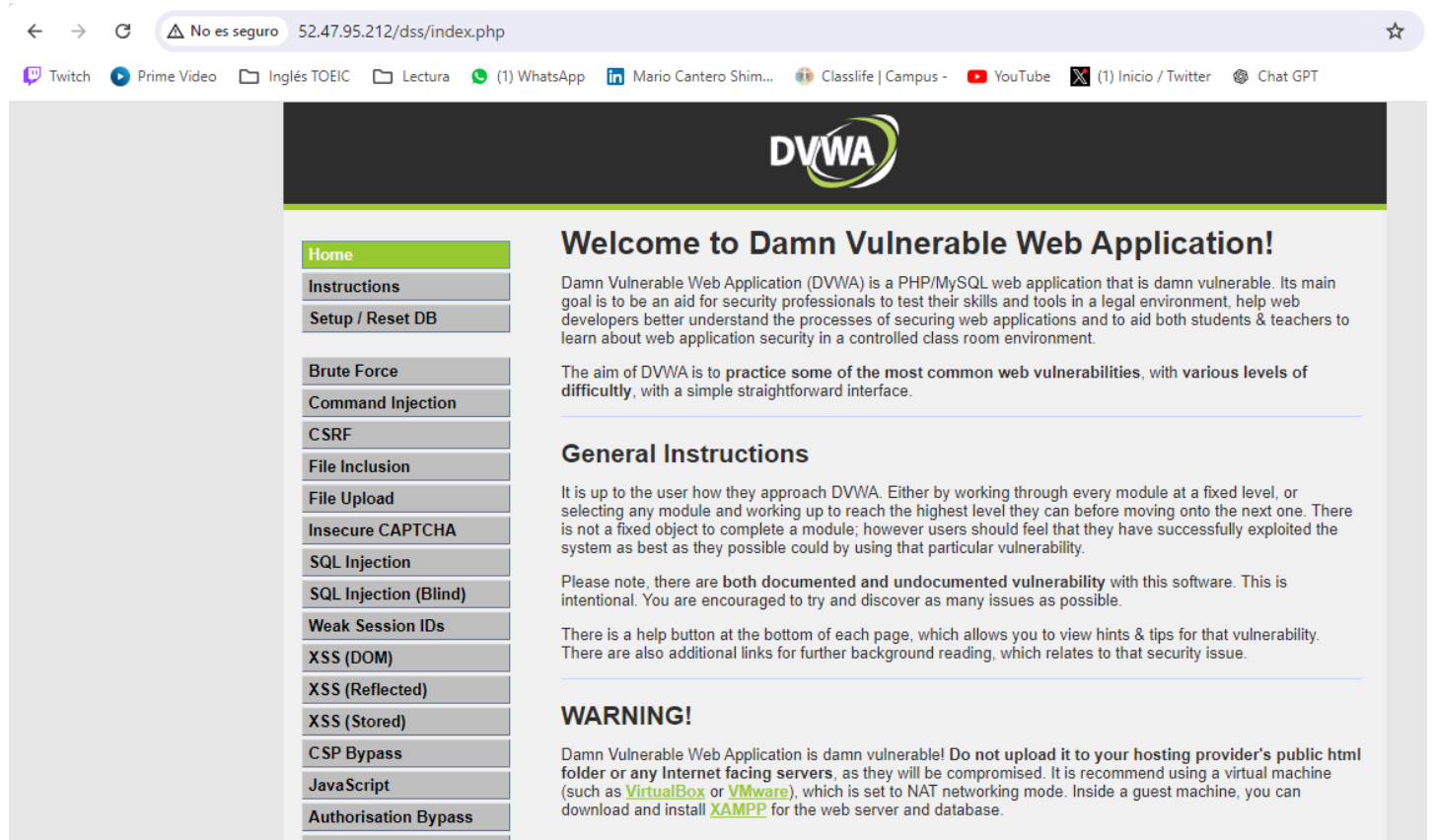


Tras esto decidí subir el proyecto de DSS mediante la terminal lo cuál lo hizo complicado no por la ejecución de comandos sino por los permisos a la hora de subir el Repositorio de dss al mio personal donde realizó las pruebas de integración continua por tema de permisos.

Además de la acomodación correspondiente de la base de datos **MySQL / MariaDB**.

Soy consciente de que AWS ya cuenta con un CI para sus propios proyectos pero he decidido no implementarlos en este proyecto con el fin de seguir el guión.

Accesible desde la IP pública mostrada en la segunda captura habilitando los puertos correspondientes.



Sonar Cloud

Para empezar me creo una cuenta en sonar cloud, y la enlace a mi proyecto de GitLab para poder trabajar y analizar con ello.

Tras eso configuro tal y como me dice sonar las variables de GitLab.

1 Add environment variables

a. Define the SonarCloud Token environment variable

In GitLab, go to [Settings > CI/CD > Variables](#) to add the following variable and make sure it is available for your project:

- 1 In the Key field, enter `SONAR_TOKEN`
- 2 In the Value field, enter `15f2579efd2907e2143717feeb034c00ce4af952`
- 3 Make sure that the Protect variable checkbox is unticked
- 4 Make sure that the Mask variable checkbox is ticked

b. Define the SonarCloud URL environment variable

Still in [Settings > CI/CD > Variables](#) [↗](#) add a new variable and make sure it is available for your project:

- 1 In the Key field, enter `SONAR_HOST_URL` [↗](#)
- 2 In the Value field, enter `https://sonarcloud.io` [↗](#)
- 3 Make sure that the Protect variable checkbox is unticked
- 4 No need to tick the Mask variable checkbox this time

Y las pongo como me dice en su respectivo sitio

Expanded variables with [↗](#) will be treated as the start of a reference to another variable.

CI/CD Variables </> 2		Reveal values	Add variable
Key ↑	Value	Environments	Actions
SONAR_HOST_URL ↗ Token de sonar URL Masked Expanded	***** ↗	All (default) ↗	✎ 🗑
SONAR_TOKEN ↗ Token de sonar Masked Expanded	***** ↗	All (default) ↗	✎ 🗑

A continuación configuro el respectivo apartado del archivo yaml de GitLab para poder usar Sonar

Create or update your `.gitlab-ci.yml` [↗](#) yaml file with the following content:

```
variables:
  SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar" # Defines the location of the analysis task cache
  GIT_DEPTH: "0" # Tells git to fetch all the branches of the project, required by the analysis task
sonarcloud-check:
  image:
    name: sonarsource/sonar-scanner-cli:latest
    entrypoint: [""]
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  script:
    - sonar-scanner
  only:
    - merge_requests
    - master
    - develop
```





[↗](#) Copy

```

53
54 variables:
55   SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar" # Defines the location of the analysis task cache
56   GIT_DEPTH: "0" # Tells git to fetch all the branches of the project, required by the analysis task
57 sonarcloud-check:
58   image:
59     name: sonarsource/sonar-scanner-cli:latest
60     entrypoint: [""]
61   cache:
62     key: "${CI_JOB_NAME}"
63     paths:
64       - .sonar/cache
65   script:
66     - sonar-scanner
67   only:
68     - merge_requests
69     - master
70     - develop
71
72

```

Y el archivo de configuración correspondiente en la raíz del proyecto

Name	Last commit	Last update
 <u>dss-master</u>	Añadir archivos desde el arc...	1 day ago
 .gitlab-ci.yml	Update .gitlab-ci.yml file	just now
 README.md	Initial commit	2 days ago
 sonar-project.proper...	Add new file	1 minute ago

Tras ejecutar la pipeline y el job de Sonar ha arrojado estos resultados

dvwa_vuln_group > DVWA_CL_checker > main

Summary **Issues** Security Hotspots Measures Code Activity

The last analysis has warnings. [See details](#)

Category	Count
Adaptability	29
Responsibility	2
Software Quality	
Security	22
Reliability	0
Maintainability	77
Severity	
High	99
Medium	249
Low	434

Add to selection

☐ Bulk Change
 Select issues
Navigate to issue
99 issues
2d 3h effort

dss-master/compose.yml

Responsibility

Make sure this MySQL database password gets changed and removed from the code. cwe +

Open Not assigned **Security** Vulnerability Blocker

30min effort 1 day ago

dss-master/database/create_mssql_db.sql

Adaptability

Define a constant instead of duplicating this literal 3 times. design +

Open Not assigned **Maintainability** Code Smell Critical

4min effort 1 day ago

Triage de los resultados de Sonar

Responsibility | Not trustworthy

Make sure this MySQL database password gets changed and removed from the code. [Link](#)

MySQL database passwords should not be disclosed [secrets:S6697](#)

Software qualities impacted: **Security**

Open Not assigned Vulnerability Blocker

Tags

cwe +

Line affected

L31

Effort

30 min

Introduced

1 day ago

Solución según sonar

Revocar el secreto

Revoca cualquier secreto filtrado y elimínalo del código fuente de la aplicación.

Antes de revocar el secreto, asegúrate de que ninguna otra aplicación o proceso lo esté utilizando. Otras utilizaciones del secreto también se verán afectadas cuando el secreto sea revocado.

Analizar el uso reciente del secreto

Cuando esté disponible, analiza los registros de autenticación para identificar cualquier uso no intencionado o malicioso del secreto desde su fecha de divulgación. Hacer esto permitirá determinar si un atacante ha aprovechado el secreto filtrado y hasta qué punto.

Esta operación debería formar parte de un proceso global de respuesta ante incidentes.

Los archivos de registro generales de MySQL contienen información sobre la autenticación de usuarios. Estos pueden ser utilizados para auditar el uso malicioso de las cuentas afectadas por la filtración de contraseñas.

Usar una bóveda de secretos

Se debe utilizar una bóveda de secretos para generar y almacenar el nuevo secreto. Esto garantizará la seguridad del secreto y evitará cualquier divulgación inesperada adicional.

Dependiendo de la plataforma de desarrollo y del tipo de secreto filtrado, actualmente existen múltiples soluciones disponibles.

Nunca codifiques secretos directamente, ni siquiera los valores predeterminados

Es importante que no codifiques secretos directamente, ni siquiera los valores predeterminados.

Primero, los secretos predeterminados codificados directamente a menudo son cortos y pueden ser fácilmente comprometidos incluso por atacantes que no tienen acceso a la base de código.

En segundo lugar, los secretos predeterminados codificados directamente pueden causar problemas si necesitan ser cambiados o reemplazados.

Y lo más importante, siempre existe la posibilidad de establecer accidentalmente secretos predeterminados para servicios de producción, lo que puede llevar a vulnerabilidades de seguridad y hacer que la producción sea insegura por defecto.

Para minimizar estos riesgos, se recomienda aplicar las estrategias anteriores, incluso para la configuración predeterminada.

dvwa_vuln_group > DVWA_CL_checker > ⓘ main

Summary Issues Security Hotspots Measures Code Activity

The last analysis has warnings. [See detail](#)

5 / 782 issues

the code.

dss-master/database/creat...

Define a constant instead of duplicating this literal 3 times.

2 locations

Duplication.

Duplication.

Navigate locations

Alt +

Adaptability | Not distinct

Define a constant instead of duplicating this literal 3 times.

String literals should not be duplicated [psql:S1192](#)

Software qualities impacted: Maintainability

Open Not assigned Code Smell Critical

Where is the issue? Why is this an issue? How can I fix it? Activity Open in IDE

14) PRIMARY KEY, comment VARCHAR(300), name VARCHAR(100),2);

15 INSERT INTO guestbook (comment, name) VALUES (

16 'This is a test comment.','test');

Tags design

Line affected L11

Effort 4 min

Introduced 1 day ago

Solución según sonar

Usar constantes para reemplazar los literales de cadena duplicados.
Las constantes pueden ser referenciadas desde muchos lugares, pero solo necesitan ser actualizadas en un solo lugar.

dvwa_vuln_group > DVWA_CI_checker > main

Summary Issues Security Hotspots Measures Code Activity

The last analysis has warnings. [See details](#)

31 / 782 issues

Extract this nested ternary operation into an independent statement.

1 location

Add curly braces around the nested statement(s).

Remove this closing tag ">".

dss-master/.../DBMS/PGSQ...

Replace all tab characters in this file by sequences of white spaces.

Intentionality | Not clear

Add curly braces around the nested statement(s).

Control structures should use curly braces [php:S121](#)

Software qualities impacted: Maintainability

Open Not assigned Code Smell Critical

Where is the issue? Why is this an issue? Activity

Open in IDE

Tags: pitfall

Line affected: L100

Effort: 2 min

Introduced: 1 day ago

```
73 dvwaMessagePush( "'guestbook' table was created." );
74
75
76 // Insert data into 'guestbook'
77 $insert =
  "INSERT INTO guestbook VALUES ('1','This is a test comment.','test');"
  ;
```

Indicaciones para el desarrollador a partir del origen del error

Omisión de las llaves: Aunque no es técnicamente incorrecto, la omisión de las llaves en una estructura de control puede resultar engañosa y llevar a la introducción de errores durante el mantenimiento del código.

En el ejemplo dado, las dos llamadas a funciones parecen estar vinculadas a la instrucción `if`, pero en realidad solo la primera está vinculada, mientras que la segunda (`checkSomething`) siempre se ejecutará. Esto puede llevar a confusión y errores si no se percibe correctamente la intención del código.

Uso de llaves: Agregar llaves mejora la legibilidad del código y su robustez. Las llaves delimitan claramente el bloque de código que está condicionado por la estructura de control, asegurando que todas las instrucciones dentro del bloque se ejecuten de acuerdo con la condición especificada. Esto evita malentendidos y errores potenciales al mantener o modificar el código.

En resumen, siempre es recomendable utilizar llaves incluso en estructuras de control simples para garantizar que el código sea claro y menos propenso a errores.

dvwa_vuln_group > DVWA_CI_checker > main

Summary Issues Security Hotspots Measures Code Activity

The last analysis has warnings. [See details](#)

36 / 782 issues

methods, which is greater than 20 authorized. Split it into smaller classes.

Rename this constant "version" to match the regular expression `^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$`.

Explicitly mention the visibility of this method "text".

Rename this local variable "\$Elements" to match the regular expression `^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$`.

Consistency | Not identifiable

Rename this constant "version" to match the regular expression `^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$`.

Constant names should comply with a naming convention [php:S115](#)

Software qualities impacted: **Maintainability**

Open Not assigned Code Smell Critical

Where is the issue? Why is this an issue? How can I fix it? Activity More info

9 `# http://erusev.com`

10 `#`

11 `# For the full license information, view the LICENSE file that was distributed`

12 `# with this source code.`

13 `#`

Tags: convention

Line affected: L20

Effort: 2 min

Introduced: 1 day ago

Oper

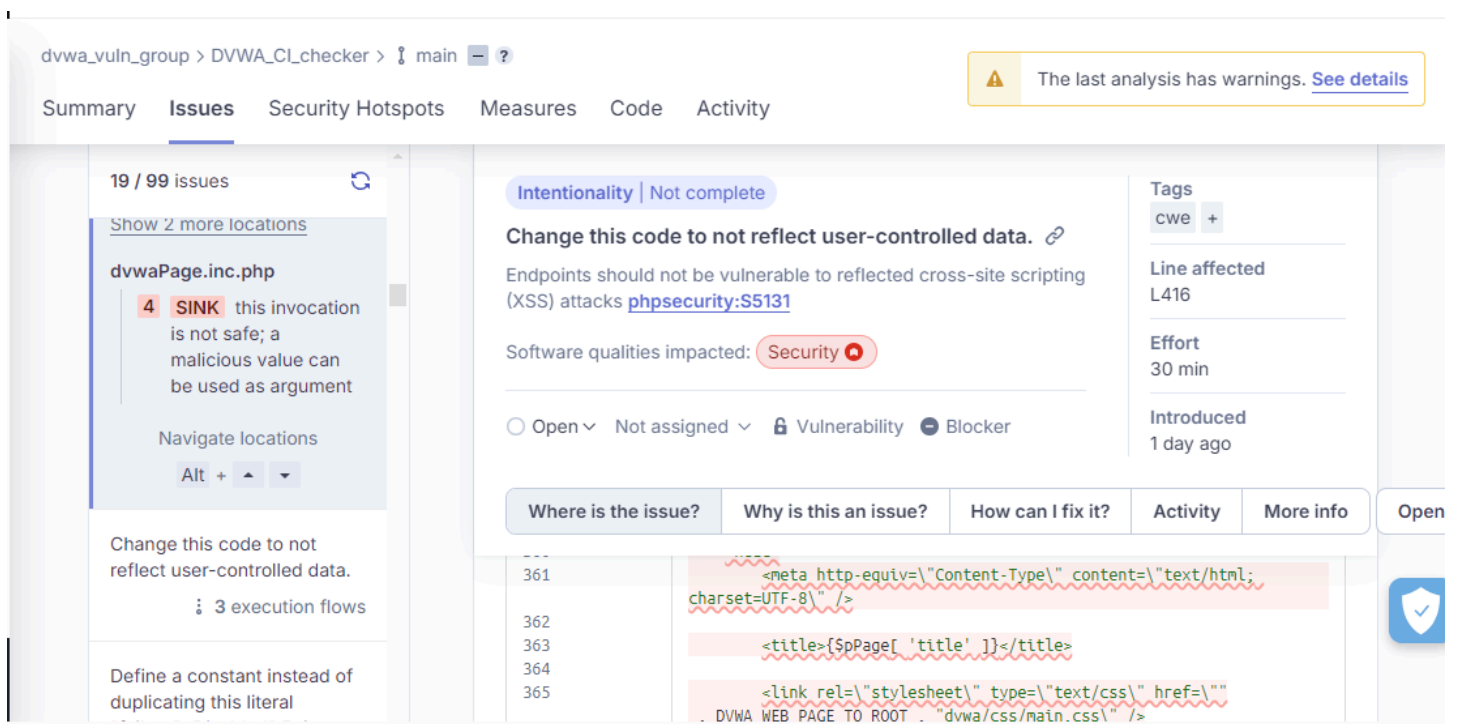
Solución según sonar

Primero, familiarízate con la convención de nombres particular del proyecto en cuestión. Luego, actualiza el nombre de la constante para que coincida con dicha convención, así como todos los usos del nombre. En muchos entornos de desarrollo integrados (IDE), puedes utilizar las funciones integradas de cambio de nombre y refactorización para actualizar todos los usos de una constante a la vez.

Explicación:

- Familiarización con la convención de nombres:** Cada proyecto puede tener sus propias reglas para nombrar constantes, variables, funciones, etc. Es importante conocer y seguir estas reglas para mantener la coherencia y la claridad en el código.
- Actualización del nombre de la constante:** Cambiar el nombre de la constante para que siga la convención del proyecto no solo mejora la legibilidad, sino que también facilita la colaboración entre los desarrolladores que trabajan en el mismo código.

3. **Actualización de todos los usos del nombre:** Asegúrate de que todos los lugares donde se usa la constante se actualicen para reflejar el nuevo nombre. Esto evita errores de referencia y garantiza que el código funcione correctamente después del cambio de nombre.
4. **Uso de características del IDE:** Muchos IDEs modernos tienen características que permiten renombrar y refactorizar elementos del código de manera automática y segura. Utilizar estas herramientas puede ahorrar tiempo y reducir la posibilidad de errores manuales al cambiar nombres en el código.



Solución según sonar

El siguiente código es vulnerable a ataques de cross-site scripting (XSS) porque devuelve una respuesta HTML que contiene datos ingresados por el usuario.

La entrada del usuario incrustada en el código HTML debe ser codificada en HTML para prevenir la inyección de código adicional. PHP proporciona la función incorporada `htmlspecialchars` para hacer esto.

GitLab

Para trabajar correctamente en GitLab he necesitado de la versión ultimate con el fin de poder usar todas sus funcionalidades.

El propio GitLab te pide que crees un grupo con propiedades premium ya que sino no te deja usar las funcionalidades completas del CI y más tarde en el repositorio creado para ello meterlo dentro de este grupo.

Para que GitLab pueda tener una hoja de instrucciones, usa un archivo yaml en el que vienen las instrucciones al completo de las pipelines.

Pipeline y justificación de stages

Stages

- **Dependency Scanning (build)**: Se coloca en la fase `build` porque el análisis de dependencias debe hacerse al inicio del ciclo para identificar y resolver problemas con las dependencias antes de proceder con el desarrollo y pruebas adicionales.
- **SAST (test)**: El análisis estático de seguridad (SAST) se realiza en la fase `test` porque es parte del análisis estático de código que debe ejecutarse antes de las pruebas dinámicas y el despliegue.
- **Secret Detection (test)**: La detección de secretos también se realiza en la fase `test` para identificar cualquier información sensible que se haya comprometido en el código antes de que avance a fases posteriores.
- **DAST (dast)**: El análisis dinámico de seguridad (DAST) se ejecuta en su propia fase `dast` después de que el código se haya construido y probado estáticamente para identificar vulnerabilidades en la aplicación en ejecución.
- **SonarCloud Check (test)**: Se ejecuta en la fase `test` para realizar un análisis estático de calidad del código y problemas de seguridad antes de proceder con el despliegue.

- **Dependency Check (test):** Se realiza en la fase test para asegurarse de que las dependencias están seguras antes de pasar a la fase de despliegue, similar al análisis estático de seguridad y la detección de secretos.

Pipeline

Para este apartado he considerado varias opciones y casos:

1. En este primero he llevado al error a drede al stage secret_detection con el fin de ver que al poner la variable allow_failure no continua con la ejecución de la pipeline.

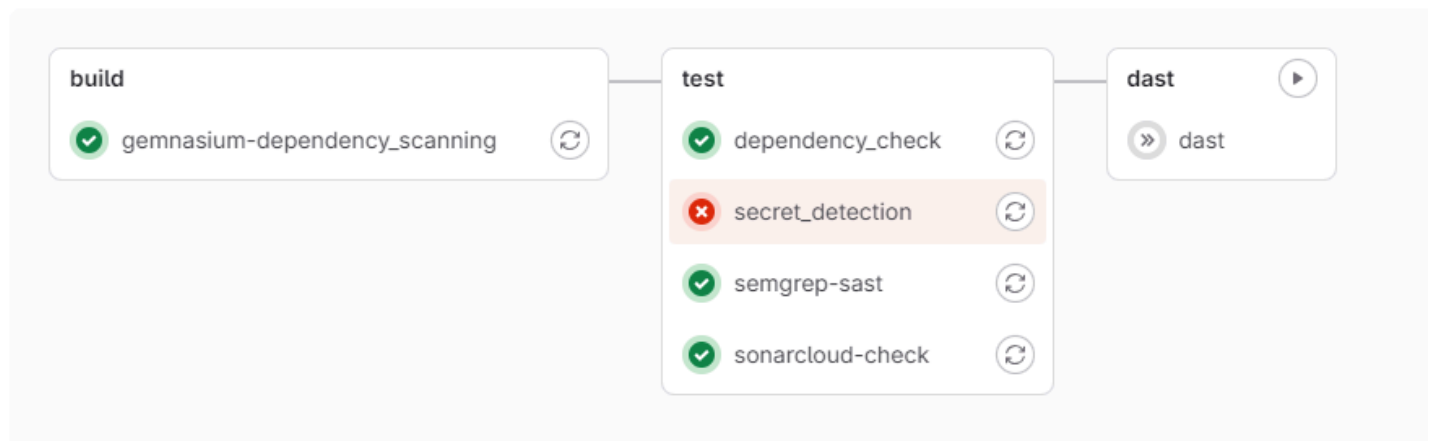
Update .gitlab-ci.yml file

✖ Failed Mario Cantero Shimizu created pipeline for commit `d6c73976` 11 minutes ago, finished 9 minutes ago

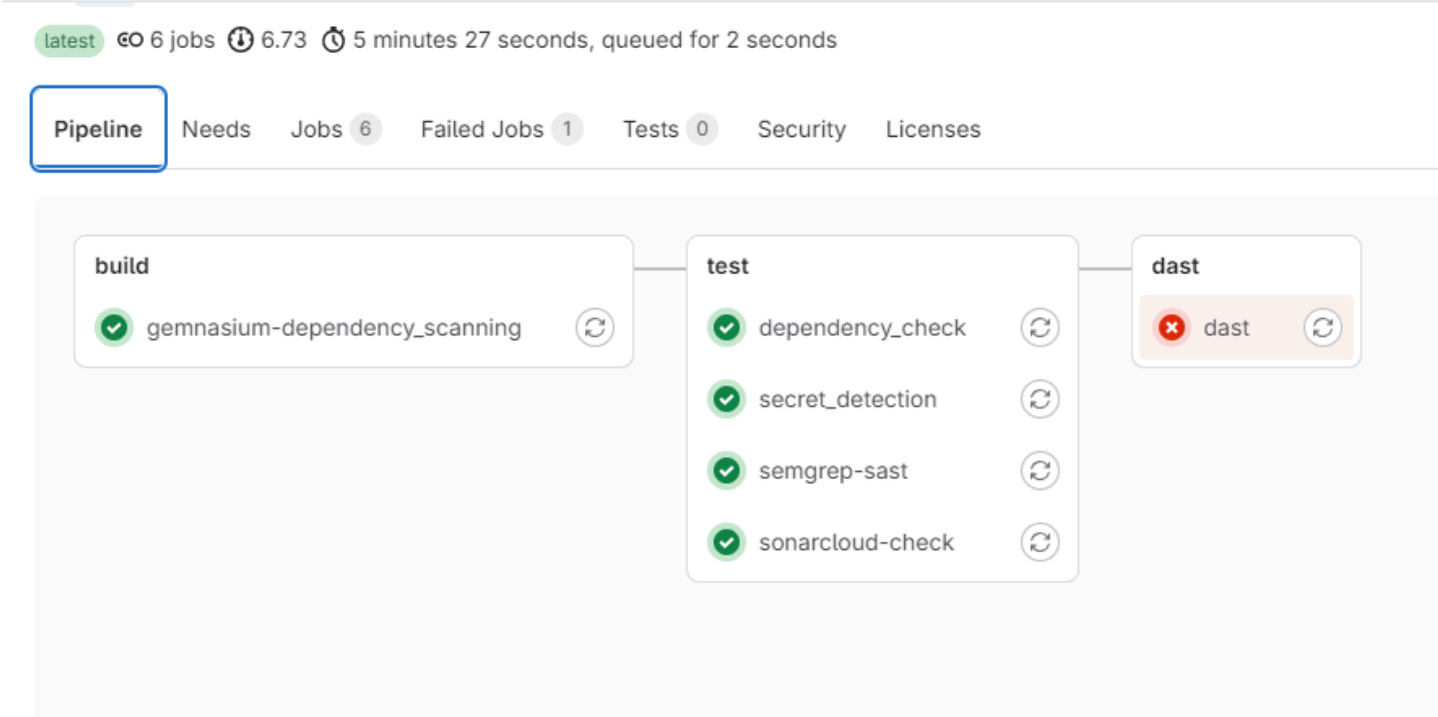
For `main`

latest 6 jobs 3.12 1 minute 56 seconds, queued for 1 seconds

Pipeline Needs Jobs 6 Failed Jobs 1 Tests 0 Security Licenses



2. Y así se vería el stage en correcto funcionamiento



Problemas con DAST

Enable DAST to automatically test for vulnerabilities in your project's running application, website, or API, in the CI/CD pipeline. Configuration changes must be applied to your `.gitlab-ci.yml` file to take effect. For details of all configuration options, see the [GitLab DAST documentation](#).

A scanner profile defines the configuration details of a security scanner. [Learn more](#).

DVWA_AWS ✓ In use

Scan mode:	active
Crawl timeout:	1 minute
Target timeout:	60 seconds
Debug messages:	Show debug messages

[Change scanner profile](#)

Site profile

A site profile defines the attributes and configuration details of your deployed application, website, or API. [Learn more](#).

DVWA_AWS ✓ In use

Target URL:	http://52.47.95.212/dss/
Site type:	Website
Authentication URL:	http://52.47.95.212/dss/login.php
Username:	admin
Password:	

A pesar de haber configurado correctamente DAST desde el code snippet de gitlab no he podido averiguar el error que no me permite el análisis de mi página.

Triaje de Vulnerabilidades relacionadas con DAST, SCA y Detección de Secretos

Needs triage Detected · 22 hours ago in pipeline [1395420612](#)

RSA private key

Description

RSA private key

Severity: ● Critical

Project: [dvwa_vuln_group](#) / [DVWA_CI_checker](#)

Tool: Secret Detection

Scanner: Gitleaks

Location

File: [dss-master/id_rsa:1](#)

1	-----BEGIN RSA PRIVATE KEY-----
---	---------------------------------

Triaje de Vulnerabilidades Detectadas

1. RSA Private Key

- **Descripción:** Se ha encontrado una clave privada RSA expuesta en el archivo [dss-master/id_rsa:1](#).
- **Severidad:** Crítica.
- **Herramienta:** Detección de Secretos (Gitleaks).
- **Recomendación:** Debes eliminar la clave privada expuesta y generar una nueva. Además, revoca cualquier acceso asociado a la clave comprometida.

Prototype Pollution in minimist

Description

Minimist <=1.2.5 is vulnerable to Prototype Pollution via file index.js, function setKey() (lines 69-95).

Severity: ● Critical

Project: [dvwa_vuln_group](#) / [DVWA_CI_checker](#)

Tool: Dependency Scanning

Scanner: Gemnasium

2. Prototype Pollution en Minimist

- **Descripción:** La versión 1.2.5 o anterior de la biblioteca [minimist](#) es vulnerable a un ataque de contaminación de prototipos.
- **Severidad:** Crítica.
- **Herramienta:** Dependency Scanning (Gemnasium).
- **Recomendación:** Actualiza la biblioteca [minimist](#) a la versión 1.2.6 o una más reciente para resolver esta vulnerabilidad.

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Description

Due to improper type validation in attachment parsing the Socket.io js library, it is possible to overwrite the _placeholder object which allows an attacker to place references to functions at arbitrary places in the resulting query object.

Severity: ● Critical

Project: [dvwa_vuln_group](#) / [DVWA_CI_checker](#)

Tool: Dependency Scanning

Scanner: Gemnasium

Location

File: [dss-master/package-lock.json](#)

3. SQL Injection en Socket.io

- **Descripción:** La biblioteca [Socket.io](#) tiene una vulnerabilidad que permite inyección SQL debido a una validación inadecuada.

- **Severidad:** Crítica.
- **Herramienta:** Dependency Scanning (Gemnasium).
- **Recomendación:** Actualiza [Socket.io](#) a la versión más reciente que corrija esta vulnerabilidad.

Needs triage Detected · 22 hours ago in pipeline [1395376533](#)

Status Needs triage ▾

decode-uri-component vulnerable to Denial of Service (DoS)

Description

decode-uri-component 0.2.0 is vulnerable to Improper Input Validation resulting in DoS.

Severity: ◆ High

Project: [dvwa_vuln_group](#) / [DVWA_CI_checker](#)

Tool: Dependency Scanning

Scanner: Gemnasium

Location

File: [dss-master/package-lock.json](#)

Links

4. Denial of Service en decode-uri-component

Descripción: La biblioteca decode-uri-component versión 0.2.0 es vulnerable a una validación incorrecta de entradas que puede llevar a un ataque de denegación de servicio.

Severidad: Alta.

Herramienta: Dependency Scanning (Gemnasium).

Recomendación: Debes actualizar decode-uri-component a una versión más reciente que haya solucionado esta vulnerabilidad.

crypto-js PBKDF2 1,000 times weaker than specified in 1993 and 1.3M times weaker than current standard

Description

crypto-js is a JavaScript library of crypto standards. Prior to version 4.2.0, crypto-js PBKDF2 is 1,000 times weaker than originally specified in 1993, and at least 1,300,000 times weaker than current industry standard. This is because it both defaults to SHA1, a cryptographic hash algorithm considered insecure since at least 2005, and defaults to one single iteration, a 'strength' or 'difficulty' value specified at 1,000 when specified in 1993. PBKDF2 relies on iteration count as a countermeasure to preimage and collision attacks. If used to protect passwords, the impact is high. If used to generate signatures, the impact is high. Version 4.2.0 contains a patch for this issue. As a workaround, configure crypto-js to use SHA256 with at least 250,000 iterations.

Severity: ● Critical

Project: [dvwa_vuln_group](#) / [DVWA_CI_checker](#)

Tool: Dependency Scanning

Scanner: Gemnasium

Location

File: [dss-master/package-lock.json](#)

5. Criptografía Débil en crypto-js PBKDF2

- **Descripción:** La biblioteca **crypto-js** antes de la versión 4.2.0 utiliza un algoritmo de derivación de clave mucho más débil de lo esperado, lo que compromete su seguridad.
- **Severidad:** Crítica.
- **Herramienta:** Dependency Scanning (Gemnasium).
- **Recomendación:** Actualiza **crypto-js** a la versión 4.2.0 o superior. Si no puedes actualizar de inmediato, configura **crypto-js** para que use SHA256 con al menos 250,000 iteraciones.