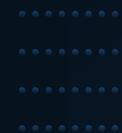


Universitat Oberta  
de Catalunya



Trabajo Final de Máster:  
Automatización de la seguridad  
en el desarrollo del Software

Máster en Ciberseguridad y Privacidad  
Área de Análisis de Datos

Jordi Nogués López

Enero de 2025

Tutor: Joan Caparrós Ramírez

Profesor: Pau Perea Paños

# Contexto

En la era digital actual, la seguridad del software es una prioridad crítica para las organizaciones de cualquier tamaño, desde diferentes puntos de vista:

- **Perspectiva empresarial:** Las brechas de seguridad pueden provocar pérdidas financieras significativas, afectar la **continuidad del negocio** y dañar la **reputación** de la organización.
- **Perspectiva legal y regulatoria:** Las organizaciones deben cumplir con leyes y regulaciones que protegen la privacidad y seguridad de los datos, como el GDPR o la Ley de Protección de Datos. El incumplimiento puede resultar en **sanciones** severas.
- **Perspectiva del usuario:** Los clientes esperan que sus datos personales y transacciones estén seguros. La **confianza** del usuario es fundamental para mantener y atraer **clientes**.
- **Perspectiva operativa:** Las vulnerabilidades pueden llevar a interrupciones del servicio, pérdida de datos y **costes adicionales para la recuperación** y mitigación de incidentes.

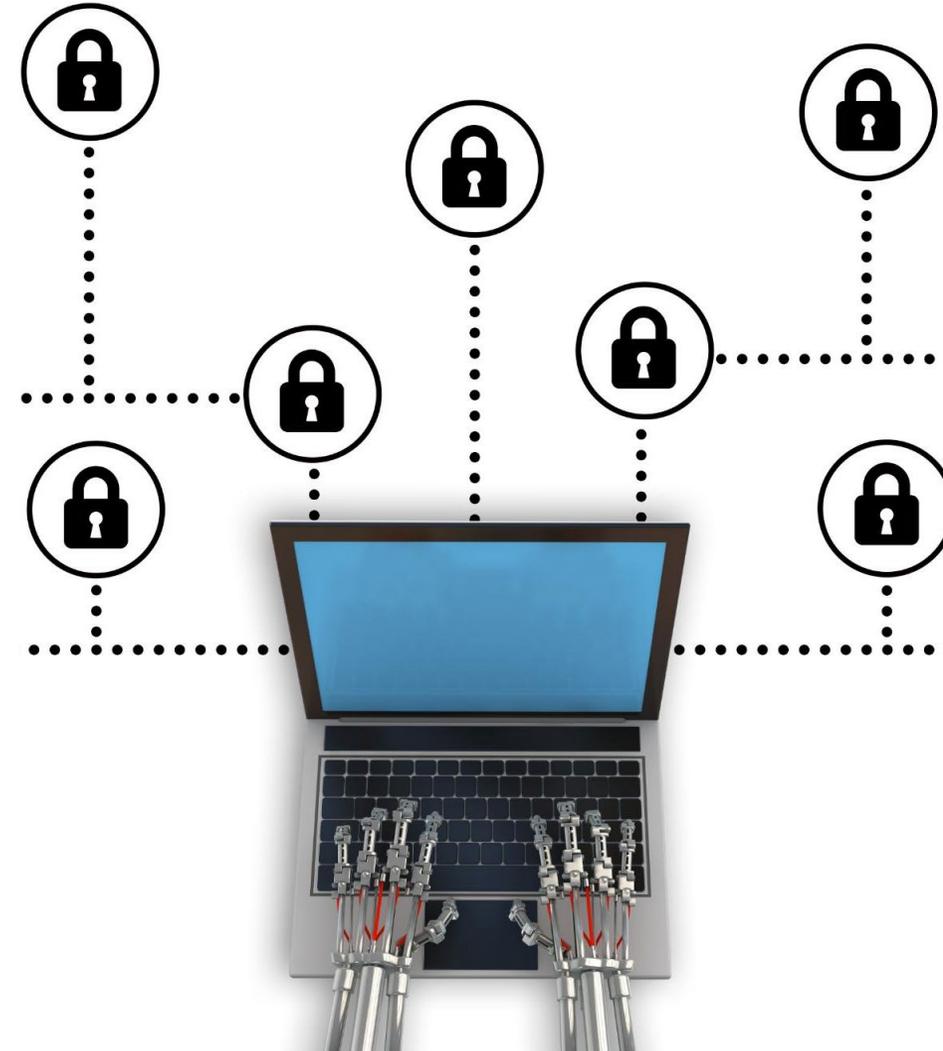


# Desafíos

Para muchas organizaciones resulta complejo integrar prácticas de verificación del software de manera efectiva.

Algunos de los principales obstáculos a los que se enfrentan son:

- **Falta de automatización:** La ausencia de procesos automatizados dificulta la integración de pruebas de seguridad en ciclos de desarrollo ágiles y continuos.
- **Falta de estandarización:** El uso de múltiples herramientas no integradas puede generar inconsistencias y complicar el análisis de resultados.
- **Desconexión entre equipos:** La falta de comunicación y colaboración entre los equipos de desarrollo, operaciones y seguridad puede conducir a retrasos en la detección y corrección de vulnerabilidades.



# Resumen ejecutivo

---

## OBJETIVOS DEL PROYECTO

- Ofrecer un análisis del **panorama actual de herramientas** de Análisis de Seguridad de Aplicaciones
- Demostrar cómo se pueden integrar esas herramientas en un **flujo DevSecOps automatizado**

## ETAPAS CLAVE

Las etapas clave del proyecto incluyen la **investigación** y **selección de herramientas**, la **implementación del entorno** de laboratorio y la **configuración del pipeline** para lograr una integración eficiente de las herramientas de análisis estático de seguridad **SAST** y **SCA**.

## LECCIONES APRENDIDAS

Entre las lecciones aprendidas, destacan la relevancia de la **automatización**, impulsada aún más por la **Inteligencia Artificial**, la importancia de **adaptar las herramientas** al flujo de trabajo del desarrollador y la necesidad de **priorizar las vulnerabilidades según el riesgo**.



# Planificación y Metodología





# Fundamentos de seguridad en el desarrollo del software

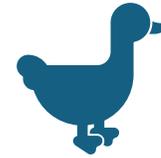
- La **seguridad por diseño** favorece que las aplicaciones sean concebidas desde su origen con medidas de protección integradas.
- **Modelado de amenazas**
  - ¿Qué es lo que se está construyendo?
  - ¿Qué es lo que puede ir mal una vez esté construido?
  - ¿Qué debe hacerse al respecto de las cosas que pueden ir mal?
- **Buenas prácticas**
  - ✓ Autenticación y autorización con protocolos seguros
  - ✓ Cifrado en reposo y en tránsito
  - ✓ Validación de entradas para evitar inyecciones de código
  - ✓ Control del manejo de errores
  - ✓ Uso de librerías confiables

A photograph of a desk setup. In the foreground, there is a silver laptop with a white keyboard. To the left of the keyboard is a magnifying glass. In the bottom left corner, there is a color calibration chart with various colored squares. The background is a solid blue color.

# Investigación - OWASP y Gartner

- La **OWASP DevSecOps Guideline** ofrece una visión completa de las herramientas disponibles para la seguridad del software, tanto comerciales como de código abierto.
- La guía muestra en qué fase del ciclo de vida del software se aplican las herramientas, cómo se integran y qué problemas solucionan.
- Los informes de **Gartner** permiten identificar a los fabricantes mejor valorados en el mercado y entender los criterios objetivos que se emplean para evaluar y posicionar cada herramienta.
- Esto aporta una perspectiva sólida para la selección de las soluciones más adecuadas en cada escenario.

# Herramientas seleccionadas



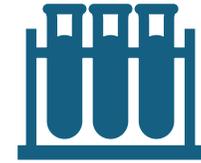
## Entorno Empresarial

SAST: Checkmarx SAST

SCA: Synopsys Black Duck

IAST: Synopsys Seeker IAST

DAST: HCL AppScan



## Laboratorio del Proyecto

SAST: CodeQL y Semgrep

SCA: Dependabot

### *SAST – Static Application Security Testing*

- *Analiza el código fuente para detectar vulnerabilidades antes de la compilación.*

### *SCA – Software Composition Analysis*

- *Identifica vulnerabilidades y problemas de licenciamiento en los componentes de terceros utilizados en la aplicación.*

### *DAST – Dynamic Application Security Testing*

- *Analiza la aplicación en tiempo de ejecución, sin acceso al código fuente.*

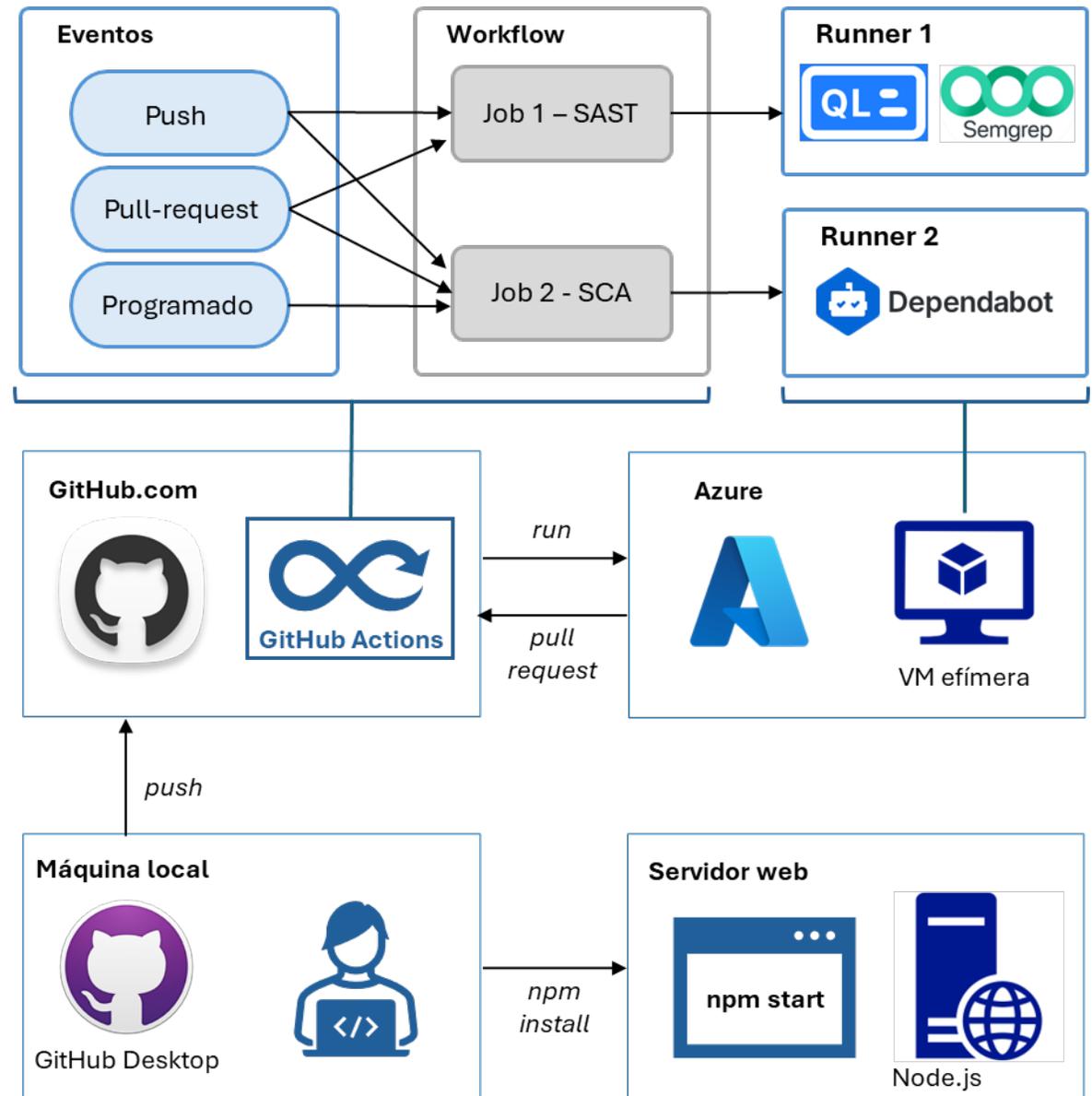
### *IAST – Interactive Application Security Testing*

- *Combina elementos de SAST y DAST proporcionando una visión más completa.*

# Diseño del laboratorio

El entorno de laboratorio consta de:

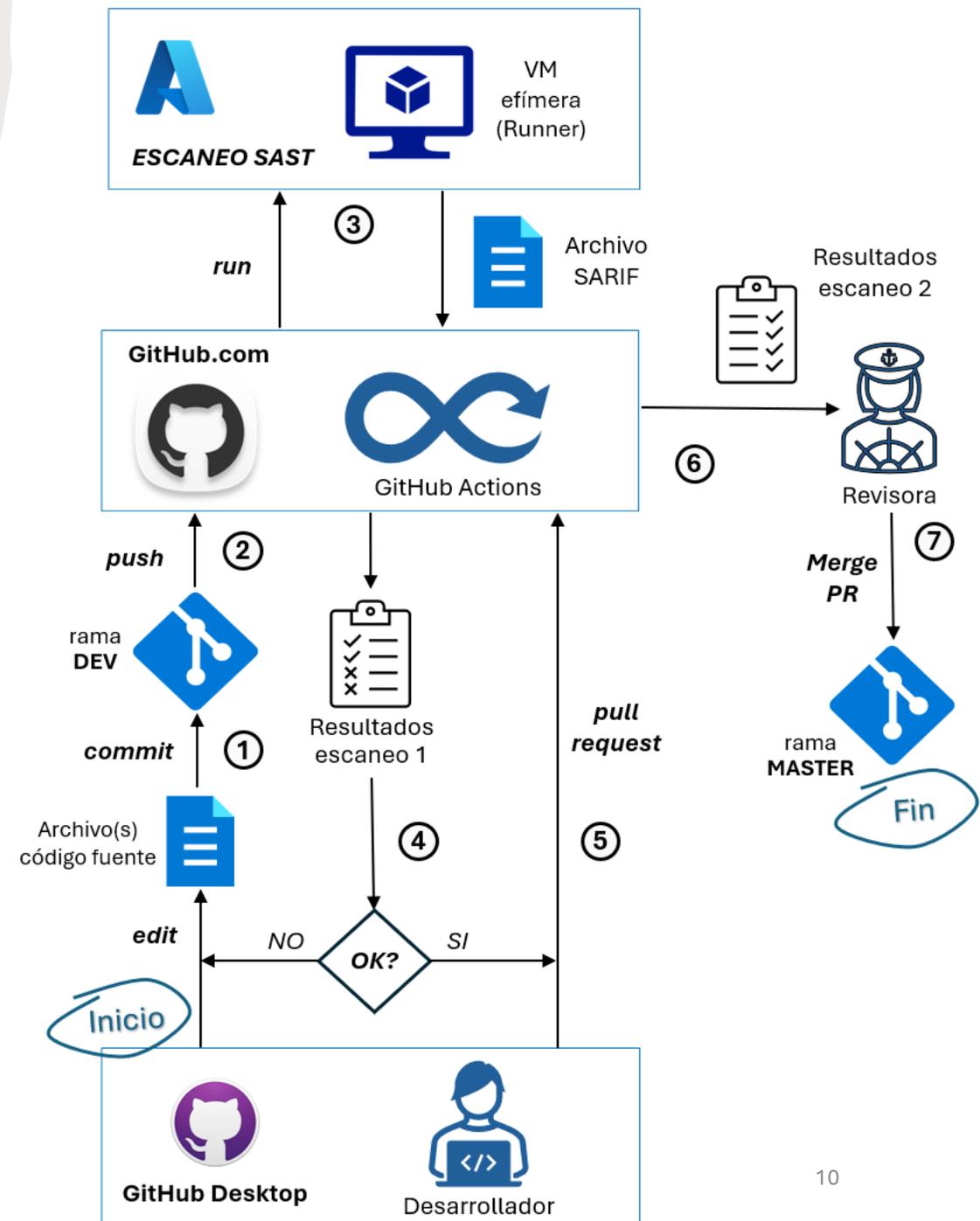
- Repositorio de código: **GitHub** (gestionado con **GitHub Desktop**)
- Plataforma *DevSecOps*: **GitHub Actions**
- Herramientas SAST: **CodeQL** y **Semgrep**
- Herramienta SCA: **Dependabot**
- Infraestructura de ejecución de las pruebas: **Azure IaaS** (proporcionada de forma transparente mediante GitHub Actions)
- Aplicación objetivo: **OWASP Juice Shop**
- Infraestructura de ejecución de la aplicación: **Node.js** (servidor web integrado)



# Diseño del flujo SAST

Este flujo debe validar la seguridad de la aplicación de forma continua, a medida que el desarrollador progresa en la construcción del **código fuente**:

1. El desarrollador realiza cambios en la rama de desarrollo de su repositorio local, y los incorpora al historial de versiones de su entorno local mediante “*commit*”.
2. Posteriormente, integra los cambios en el repositorio central mediante el mandato “*push*”.
3. La plataforma *DevSecOps* (GitHub Actions) dispara el proceso de escaneo del código, clonándolo previamente en una Máquina Virtual temporal (*runner*) en Azure.
4. El resultado del escaneo se publica en GitHub y se le notifica al desarrollador para que lo verifique y depure los posibles errores detectados.
5. Mediante “*pull-request*”, el desarrollador solicita la integración de los cambios en la rama principal del proyecto.
6. GitHub Actions dispara nuevamente el escaneo para verificar si está todo correcto, y genera el informe correspondiente.
7. Tras completar las verificaciones, la jefatura del proyecto fusiona los cambios del “*pull-request*” en la rama principal del repositorio.



# Diseño del flujo SCA

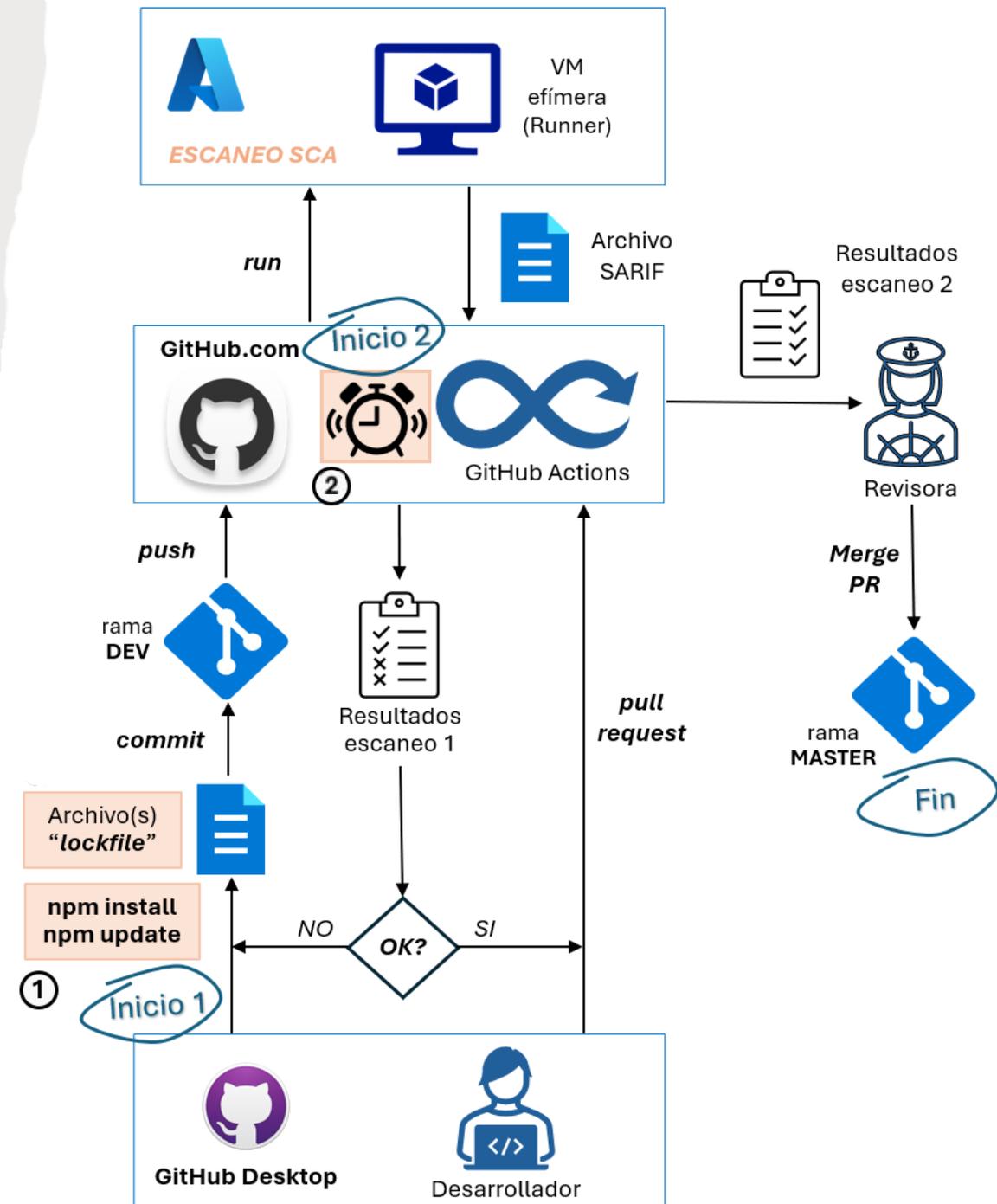
El análisis SCA se centra en verificar la seguridad de las **librerías de terceros y open-source** utilizadas en el proyecto.

Por consiguiente, no es necesario ejecutar el escaneo ante cualquier cambio en el código, sino solamente cuando se instala o actualiza alguna dependencia.

Además, también hay que verificar regularmente que las librerías y sus licencias de uso siguen estando vigentes.

Por consiguiente, el flujo debe activarse en respuesta a 2 eventos distintos:

1. Cuando el desarrollador **añada o actualice** la versión de **una librería** ya instalada en el proyecto.
2. De forma automática **a una hora programada**, independientemente de si ha habido cambios en las librerías.



# Configuración del flujo

## Parámetros principales

- Eventos de activación: *push*, *pull-request*, *schedule*
- Permisos: De lectura para acceder al repositorio de código, y de escritura sobre el registro de eventos de seguridad
- Sistema operativo del *runner*: *Ubuntu-latest*
- Clonado del código del repositorio en la VM de Azure:
  - `actions/checkout@v4`
- Instalación y ejecución de la herramienta SAST en la VM:
  - `run: pip install semgrep`
  - `run: semgrep scan --sarif --output=...`
- Asignación del token de autenticación mediante variable de entorno:
  - `Env: SEMGREP_APP_TOKEN: ${ secrets.SEMGREP_APP_... }`
- Carga en GitHub del archivo SARIF con las alertas detectadas:
  - `github/codeql-action/upload-sarif@v3`

```
C:\Users\nogue\OneDrive\Documentos\GitHub\JuiceShop\.github\workflows\semgrep.yml - Note...
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar
Complementos  Pestañas  ?
javascript.sarif  README.md  semgrep.yml  semgrep.sarif
6   # This workflow file requires a free account on Semgrep.dev to
7   # manage rules, file ignores, notifications, and more.
8   #
9   # See https://semgrep.dev/docs
10
11  name: Semgrep
12
13  on:
14    push:
15      branches: [ "master" ]
16    pull_request:
17      # The branches below must be a subset of the branches above
18      branches: [ "master" ]
19    schedule:
20      - cron: '21 11 * * 4' #jn ejecución cada jueves a las 11:21 (UTC)
21
22  permissions:
23    contents: read
24
25  jobs:
26    semgrep:
27      permissions:
28        contents: read # for actions/checkout to fetch code
29        security-events: write # for github/codeql-action/upload-sarif to upload SARIF
30        actions: read # only required for a private repository by github/codeql-action/
31      name: Scan
32      runs-on: ubuntu-latest #jn s.o. de la VM que GitHub Actions asignará en Azure
33      steps:
34        # Checkout project source
35        - uses: actions/checkout@v4 #jn clona el código en la VM temporal
36
37        - name: Instalar Semgrep con pip en la VM #jn (gestor de paquetes para Python)
38          run: pip install semgrep
39
40        # Scan code using project's configuration on https://semgrep.dev/manage
41        - name: Run Semgrep scan
42          run:
43            semgrep scan --sarif --output=semgrep.sarif --config=auto
44          env:
45            SEMGREP_APP_TOKEN: ${ secrets.SEMGREP_APP_TOKEN } #jn token de semgre
46
47        #jn Verificar si se ha grabado el archivo SARIF
48        - name: Listar los archivos SARIF
49          run: ls -l semgrep.sarif
50
51        # Upload SARIF file generated in previous step
52        - name: Upload SARIF file
53          uses: github/codeql-action/upload-sarif@v3
54          with:
55            sarif_file: semgrep.sarif
56          if: always()
```



# Implementación

A continuación, se demuestra la implementación de 2 casos de uso en el entorno de laboratorio:

1. SAST – Detección de vulnerabilidades en el código con **CodeQL** y remediación con **Copilot Autofix**
2. SCA – Detección y resolución automática de librerías no seguras u obsoletas con **Dependabot**.

# 1. SAST – Publicación de los cambios en el código

The image shows two overlapping screenshots of the Visual Studio Code interface. The top screenshot displays the 'Commit' dialog for the file 'README.md'. The commit message is 'Cambio mínimo (introducción comentario) para provocar la activación de los flujos.' The diff view shows a new line added at line 27: '#modificado jn para forzar el lanzamiento de flujos'. The bottom screenshot shows the 'Push origin' button in the toolbar, indicating that the local commit is ready to be pushed to the remote repository.

Current repository: JuiceShop  
Current branch: master  
Fetch origin: Last fetched 7 minutes ago

Check out the new [accessibility settings](#) to control the visibility of the link underlines and diff check marks.

Changes 1 | History | README.md

1 changed file

24 24 @@ -24,7 +24,7 @@  
> ([[kramse](https://twitter.com/kramse)) -  
25 25 > [But this doesn't have anything to do with juice.](https://twitter.com/coderPatros/status/1199268774626488320)  
26 26 > ([[coderPatros' wife](https://twitter.com/coderPatros))

27 -  
27 + #modificado jn para forzar el lanzamiento de flujos

28 28 OWASP Juice Shop is probably the most modern and sophisticated insecure web application! It ca  
n be used in secur  
29 29 trainings, awarenes  
s vulnerabilities  
30 30 entire

Update README.md

Cambio mínimo (introducción comentario) para provocar la activación de los flujos.

Commit to master

Current repository: JuiceShop  
Current branch: master  
Push origin: Last fetched 8 minutes ago

Changes | History

0 changed files

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote  
You have 1 local commit waiting to be pushed to GitHub.  
Always available in the toolbar when there are local commits waiting to be pushed or  
Ctrl + P

Push origin

# 1. SAST – Consulta de alertas

The screenshot displays the GitHub Security Center interface for the repository 'JordiTFM / JuiceShop'. The 'Security' tab is highlighted in the top navigation bar with a red box, showing 88 alerts. The 'Code scanning' section is active, displaying a summary of 88 open alerts and 0 closed alerts. A search bar contains the query 'is:open branch:master'. Below the search bar, a table of alerts is shown with a red box highlighting the filter controls: Language, Tool, Branch, Rule, Severity, and Sort. The table lists three critical alerts:

Alert ID	Rule	Severity	Branch
#81	Type confusion through parameter tampering	Critical	master
#80	Type confusion through parameter tampering	Critical	master
#71	Code injection	Critical	master

# 1. SAST – Gestión de la incidencia

github.com/JordiTFM/JuiceShop/security/code-scanning/81

## Type confusion through parameter tampering

**Open** in `master` 2 hours ago

Speed up the remediation of this alert with [Copilot Autofix for CodeQL](#) **Generate fix**

```
routes/search.ts:22
19  module.exports = function searchProducts () {
20    return (req: Request, res: Response, next: NextFunction) => {
21      let criteria: any = req.query.q === 'undefined' ? '' : req.query.q ?? ''
22      criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200)

```

Potential type confusion as this HTTP request parameter may be either an array or a string.

CodeQL [Show paths](#)

```
23  models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%${criteria}%' OR description LIKE '%$
24    .then(([products]: any) => {
25    const dataString = JSON.stringify(products)

```

Severity  
**Critical**

Affected branches  
**master**

Tags  
**security**

Weaknesses  
**CWE-843**

Tool	Rule ID	Query
CodeQL	js/type-confusion-through-parameter-tampering	<a href="#">View source</a>

Sanitizing untrusted HTTP request parameters is a common technique for preventing injection attacks such as SQL injection or path traversal. This is sometimes done by checking if the request parameters contain blacklisted substrings.

# 1. SAST – Remediación automática con COPILOT

## Type confusion through parameter tampering Dismiss

Open in `master` 4 days ago

**Copilot Autofix** AI generated a fix less than a minute ago

To fix the problem, we need to ensure that the `req.query.q` parameter is a string before using it. This can be done by adding a type check for `req.query.q` and handling cases where it is not a string. If the parameter is not a string, we should treat it as an invalid input and handle it accordingly.

1. Add a type check for `req.query.q` to ensure it is a string.
2. If `req.query.q` is not a string, set `criteria` to an empty string or handle it as an invalid input.
3. This change should be made in the `routes/search.ts` file, specifically around lines 21-22.

routes/search.ts:22 Autofix

...	<code>@@ -20,3 +20,3 @@</code>	
20	<code>return (req: Request, res: Response, next: NextFunction) =&gt; {</code>	20 <code>return (req: Request, res: Response, next: NextFunction) =&gt; {</code>
21	<code>let criteria: any = req.query.q === 'undefined' ? '' : req.query.q ?? ''</code>	21 <code>let criteria: any = (typeof req.query.q === 'string') ? req.query.q : ''</code>
22	<code>criteria = (criteria.length &lt;= 200) ? criteria : criteria.substring(0, 200)</code>	22 <code>criteria = (criteria.length &lt;= 200) ? criteria : criteria.substring(0, 200)</code>

Severity: **Critical**

Affected branches: `master`

Tags: `security`

Weaknesses: **CWE-843**

## 2. SCA – Consulta de alertas

The screenshot shows the GitHub interface for the repository 'JordiTFM / JuiceShop'. The 'Security' tab is active, displaying 'Dependabot alerts'. The page shows 13 open alerts and 0 closed alerts. A search filter 'is:open' is applied. A red box highlights the filter controls: 'Package', 'Ecosystem', 'Manifest', 'Severity', and 'Sort'. A yellow arrow points to the first alert: 'Verification Bypass in jsonwebtoken' (Critical). Other alerts include 'jsonwebtoken unrestricted key type could lead to legacy keys usage' (High), 'Command Injection in marsdb' (Critical), and 'Sanitize-html Vulnerable To REDoS Attacks' (High).

Overview

Reporting

Policy

Advisories

Vulnerability alerts

- Dependabot 13
- Code scanning 215
- Secret scanning 26

### Dependabot alerts

Give feedback Configure

is:open

13 Open  0 Closed

Package Ecosystem Manifest Severity Sort

- Verification Bypass in jsonwebtoken** Critical  
#1 opened 8 minutes ago • Detected in jsonwebtoken (npm) • package.json
- jsonwebtoken unrestricted key type could lead to legacy keys usage** High  
#8 opened 8 minutes ago • Detected in jsonwebtoken (npm) • package.json
- Command Injection in marsdb** Critical  
#5 opened 2 months ago • Detected in marsdb (npm) • package.json
- Sanitize-html Vulnerable To REDoS Attacks** High #5  
#12 opened 2 months ago • Detected in sanitize-html (npm) • package.json

## 2. SCA – Gestión de la incidencia

### Verification Bypass in jsonwebtoken #1 Dismiss alert ▾

**Open** Opened 2 months ago on jsonwebtoken (npm) · package.json

Upgrade jsonwebtoken to fix 4 Dependabot alerts in package.json  
Upgrade jsonwebtoken to version 9.0.0 or later. For example:

```
"dependencies": {
  "jsonwebtoken": ">=9.0.0"
}

"devDependencies": {
  "jsonwebtoken": ">=9.0.0"
}
```

**Create Dependabot security update**

Package	Affected versions	Patched version
jsonwebtoken (npm)	< 4.2.2	4.2.2

Versions 4.2.1 and earlier of jsonwebtoken are affected by a verification bypass vulnerability. This is a result of weak validation of the JWT algorithm type, occurring when an attacker is allowed to arbitrarily specify the JWT algorithm.

### Recommendation

Update to version 4.2.2 or later.

**Severity**  
Critical

**Tags**  
Runtime dependency Patch available

**Weaknesses**  
CWE-20

**Related alerts**

- jsonwebtoken unrestricted key type could lea...
- jsonwebtoken's insecure implementation of k...
- jsonwebtoken vulnerable to signature validati...

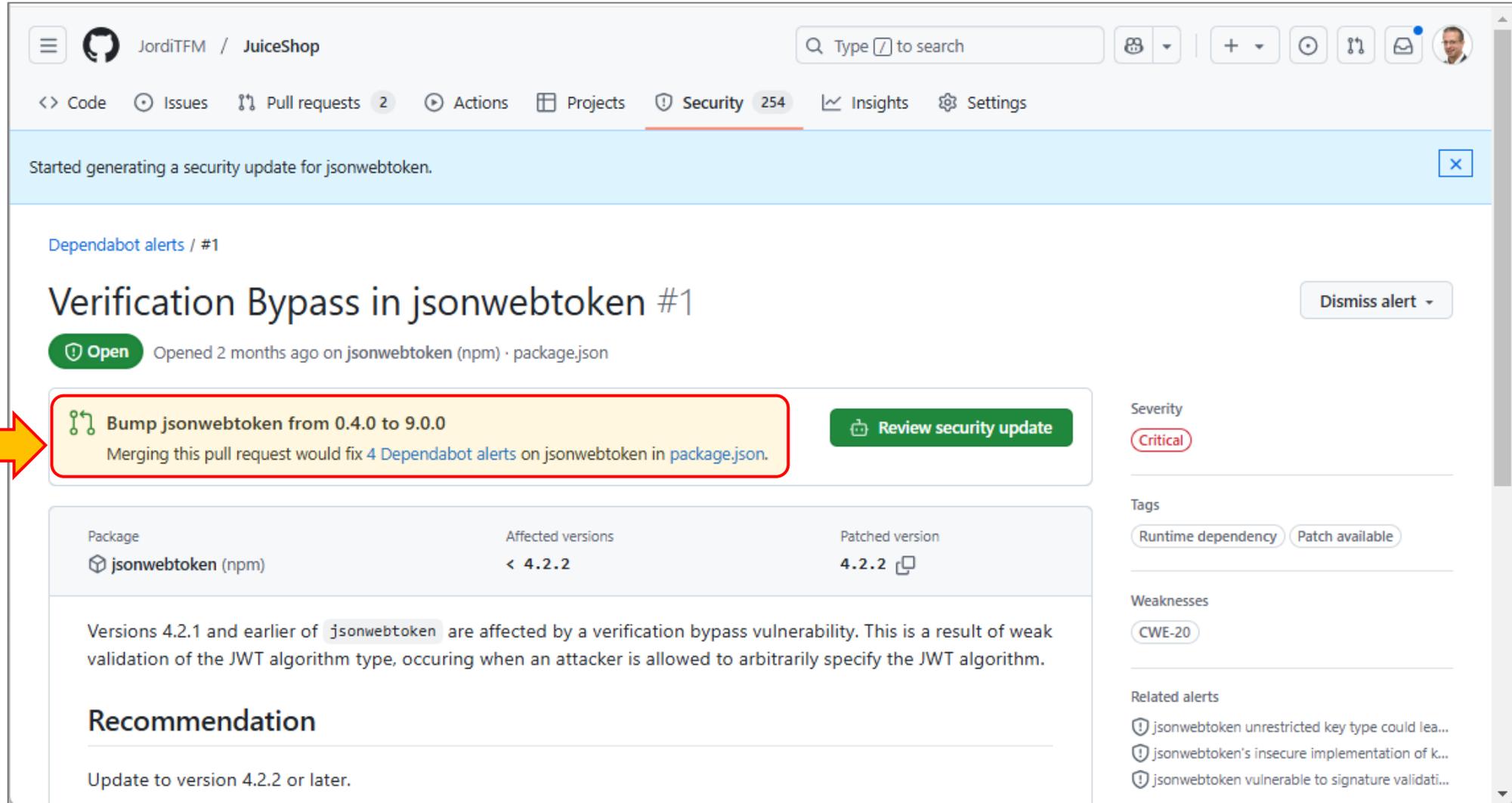
**CVE ID**  
CVE-2015-9235

**GHSA ID**  
GHSA-c7hr-j4mj-j2w6

See advisory in GitHub Advisory Database

See all of your affected repositories

## 2. SCA – Actualización automática de la librería



The screenshot shows a GitHub repository page for 'JordiTFM / JuiceShop'. A notification bar at the top states 'Started generating a security update for jsonwebtoken.' Below this, a 'Dependabot alerts / #1' section displays a 'Verification Bypass in jsonwebtoken #1' alert. The alert is marked as 'Open' and was opened 2 months ago. A yellow box highlights a suggested pull request: 'Bump jsonwebtoken from 0.4.0 to 9.0.0', which would fix 4 Dependabot alerts. A green 'Review security update' button is next to it. The alert's severity is 'Critical', and it has tags for 'Runtime dependency' and 'Patch available'. The weakness is identified as 'CWE-20'. A table shows the affected versions (< 4.2.2) and the patched version (4.2.2). A recommendation section advises updating to version 4.2.2 or later. The right sidebar shows related alerts for jsonwebtoken.

Started generating a security update for jsonwebtoken.

Dependabot alerts / #1

### Verification Bypass in jsonwebtoken #1

Dismiss alert ▾

**Open** Opened 2 months ago on jsonwebtoken (npm) · package.json

 **Bump jsonwebtoken from 0.4.0 to 9.0.0**  
Merging this pull request would fix 4 Dependabot alerts on jsonwebtoken in package.json.

**Review security update**

Severity  
**Critical**

Tags  
Runtime dependency Patch available

Weaknesses  
CWE-20

Related alerts

- jsonwebtoken unrestricted key type could lea...
- jsonwebtoken's insecure implementation of k...
- jsonwebtoken vulnerable to signature validati...

Package	Affected versions	Patched version
jsonwebtoken (npm)	< 4.2.2	4.2.2

Versions 4.2.1 and earlier of `jsonwebtoken` are affected by a verification bypass vulnerability. This is a result of weak validation of the JWT algorithm type, occurring when an attacker is allowed to arbitrarily specify the JWT algorithm.

### Recommendation

Update to version 4.2.2 or later.

## 2. SCA – Fusión del cambio

The screenshot displays a GitHub pull request for the title "Bump jsonwebtoken from 0.4.0 to 9.0.0 #7". The pull request is from the repository "dependabot/npm\_and\_yarn/j..." to the "master" branch. A comment from "dependabot" indicates that the "dependencies" label was added 26 minutes ago. A comment from "JordiTFM" (owner) states: "Merge del PR automático de Dependabot para resolver la vulnerabilidad de la librería jsonwebtoken //jordi 19-dic-24". The pull request status is "All checks have passed" (1 successful check) and "This branch has no conflicts with the base branch". A green "Merge pull request" button is highlighted with a red arrow. The right sidebar shows notification settings and a "Lock conversation" option.

**Open** Bump jsonwebtoken from 0.4.0 to 9.0.0 #7  
dependabot wants to merge 1 commit into `master` from `dependabot/npm_and_yarn/j...`

**dependabot** (bot) added the **dependencies** label 26 minutes ago

**JordiTFM** commented now (Owner) ...  
Merge del PR automático de Dependabot para resolver la vulnerabilidad de la librería jsonwebtoken //jordi 19-dic-24

**Require approval from specific reviewers before merging**  
Rulesets ensure specific people approve pull requests before they're merged. **Add rule** ×

**All checks have passed** (1 successful check) [Show all checks](#)

**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Successfully merging this pull request may close these issues.  
None yet

Notifications [Customize](#)  
**Unsubscribe**  
You're receiving notifications because you're watching this repository.

0 participants

**Lock conversation**



# CONCLUSIONES

**La automatización es posible,** y permite producir un código seguro sin penalizar el ritmo de desarrollo que exige el negocio digital.

**Impulsada por la Inteligencia Artificial,** acelera el proceso de remediación y democratiza el acceso a las buenas prácticas de ciberseguridad.



# Posibles trabajos futuros

## Integración de herramientas DAST e IAST

El uso de herramientas DAST e IAST puede ofrecer las máximas garantías de eficacia en los controles de seguridad del software. El reto principal será transformar las alertas generadas a un formato compatible que permita integrarlas en el entorno *DevSecOps*.

## Benchmarking de herramientas

La evolución de plataformas como el *OWASP Benchmark Project*, actualmente limitada a aplicaciones desarrolladas con el SDK de Java 11, abriría la puerta a una evaluación objetiva de la eficiencia de herramientas SAST sobre aplicaciones modernas.

*Gracias!*