Writeup: Sala *Intro to Pipeline Automation*

Autor: Ismaeldevs

Plataforma: TryHackMe

4 de julio de 2025

Índice

1. Introducción **2**

2. Sala **2**

2.1. Tarea 1 – Introducción 2
2.2. Tarea 2 - Explicación de los pipelines de DevOps 2
2.3. Tarea 3 - Código fuente y control de versiones 2
2.4. Tarea 4 - Gestión de dependencias 3
2.5. Tarea 5 - Pruebas automatizadas 3
2.6. Tarea 6 - Integración y entrega continuas 4
2.7. Tarea 7 - Entornos 4
2.8. Tarea 8 - Reto 5
2.9. Tarea 9 - Conclusión 10

3. Conclusión sobre la Sala **10**

1. Introducción

Esta sala nos ofrece una visión completa del flujo moderno de desarrollo DevOps, desde la gestión del código fuente hasta el despliegue automatizado, haciendo énfasis en la seguridad en cada etapa.

2. Sala

2.1. Tarea 1 – Introducción

Nos explican cómo la automatización se integra al ciclo de desarrollo (SDLC) y resalta los beneficios y riesgos de introducir automatizaciones.

Pregunta: I'm ready to learn about pipeline automation and how to make sure it is secure!
Respuesta: **No requiere respuesta** (Hacemos clic en **Submit**).

2.2. Tarea 2 - Explicación de los pipelines de DevOps

En esta tarea aprenderemos qué es un **pipeline DevOps**, muestra sus etapas típicas (control de código, pruebas, despliegue, etc.) y contextualiza dónde puede fallar la seguridad, ofreciendo un panorama general antes de profundizar en cada fase.

Pregunta: Where in the pipeline is our end product deployed?
Respuesta: **Environments**

2.3. Tarea 3 - Código fuente y control de versiones

Esta tarea se centra en la gestión del código fuente usando sistemas de control de versiones (como Git o SVN), y enfatiza la importancia de autenticar y controlar accesos, además de alertar sobre el riesgo de exponer datos sensibles en commits.

Pregunta: Who is the largest online provider of Git?
Respuesta: **Github**

Pregunta: What popular Git product is used to host your own Git server?
Respuesta: **Gitlab**

Pregunta: What tool can be used to scan the commits of a repo for sensitive information?

Respuesta: **GittyLeaks**

2.4. Tarea 4 - Gestión de dependencias

Aprenderemos sobre la gestión de dependencias externas e internas mediante repositorios (PyPI, NuGet, Artifactory), y analizar los riesgos como vulnerabilidades en librerías (por ejemplo, Log4Shell) que pueden afectar múltiples proyectos.

Pregunta: What do we call the type of dependency that was created by our organisation? (Internal/External)
Respuesta: **Internal**

Pregunta: What type of dependency is JQuery? (Internal/External)
Respuesta: **External**

Pregunta: What is the name of Python's public dependency repo?
Respuesta: **PyPi**

Pregunta: What dependency 0day vulnerability set the world ablaze in 2021?
Respuesta: **Log4j**

2.5. Tarea 5 - Pruebas automatizadas

Nos introducimos en las pruebas automatizadas (unitarias e integración), así como pruebas de seguridad (SAST para análisis estático y DAST para análisis dinámico). También, nos explica también por qué no pueden reemplazar completamente al testing manual.

Pregunta: What type of tool scans code to look for potential vulnerabilities?
Respuesta: **SAST**

Pregunta: What type of tool runs code and injects test cases to look for potential vulnerabilities?
Respuesta: **DAST**

Pregunta: Can SAST and DAST be used as a replacement for penetration tests? (Yea,Nay)
Respuesta: **Nay**

2.6. Tarea 6 - Integración y entrega continuas

Conoceremos el concepto de **CI/CD**, describiendo sus componentes como:

- **Disparadores**
- **Construcción**
- **Pruebas**
- **Despliegue**

También, señala el riesgo de usar agentes y orquestadores compartidos entre entornos, lo cual puede escalar brechas de seguridad.

Pregunta: What does CI in CI/CD stand for?
Respuesta: **Continuous Integration**

Pregunta: What does CD in CI/CD stand for?
Respuesta: **Continuous Delivery**

Pregunta: What do we call the build infrastructure element that controls all builds?
Respuesta: **Build Orchestrator**

Pregunta: What do we call the build infrastructure element that performs the build?
Respuesta: **Build Agent**

2.7. Tarea 7 - Entornos

Vamos a conocer los distintos entornos en el pipeline (DEV, UAT, PreProd, PROD, DR/HA), su nivel de estabilidad y seguridad, y cómo las estrategias como Blue/Green o Canary reducen riesgos mediante segregación y control progresivo de cambios.

Pregunta: Which environment usually has the weakest security configuration?
Respuesta: **DEV**

Pregunta: Which environment is used to test the application?
Respuesta: **UAT**

Pregunta: Which environment is similar to PROD but is used to verify that everything is working before it is pushed to PROD?
Respuesta: **PrePROD**

Pregunta: What is a common class of vulnerabilities that is discovered in PROD due to insecure code creeping in from DEV?

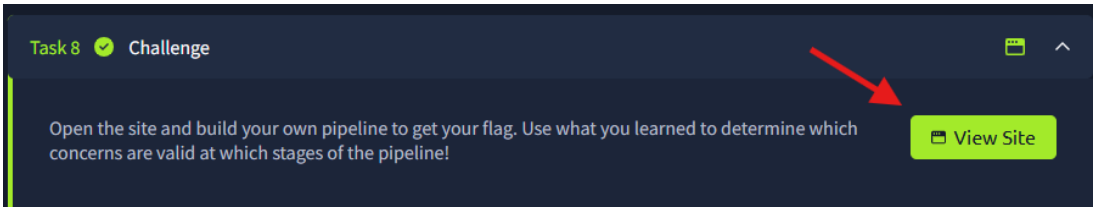
Respuesta: **Developer Bypasses**

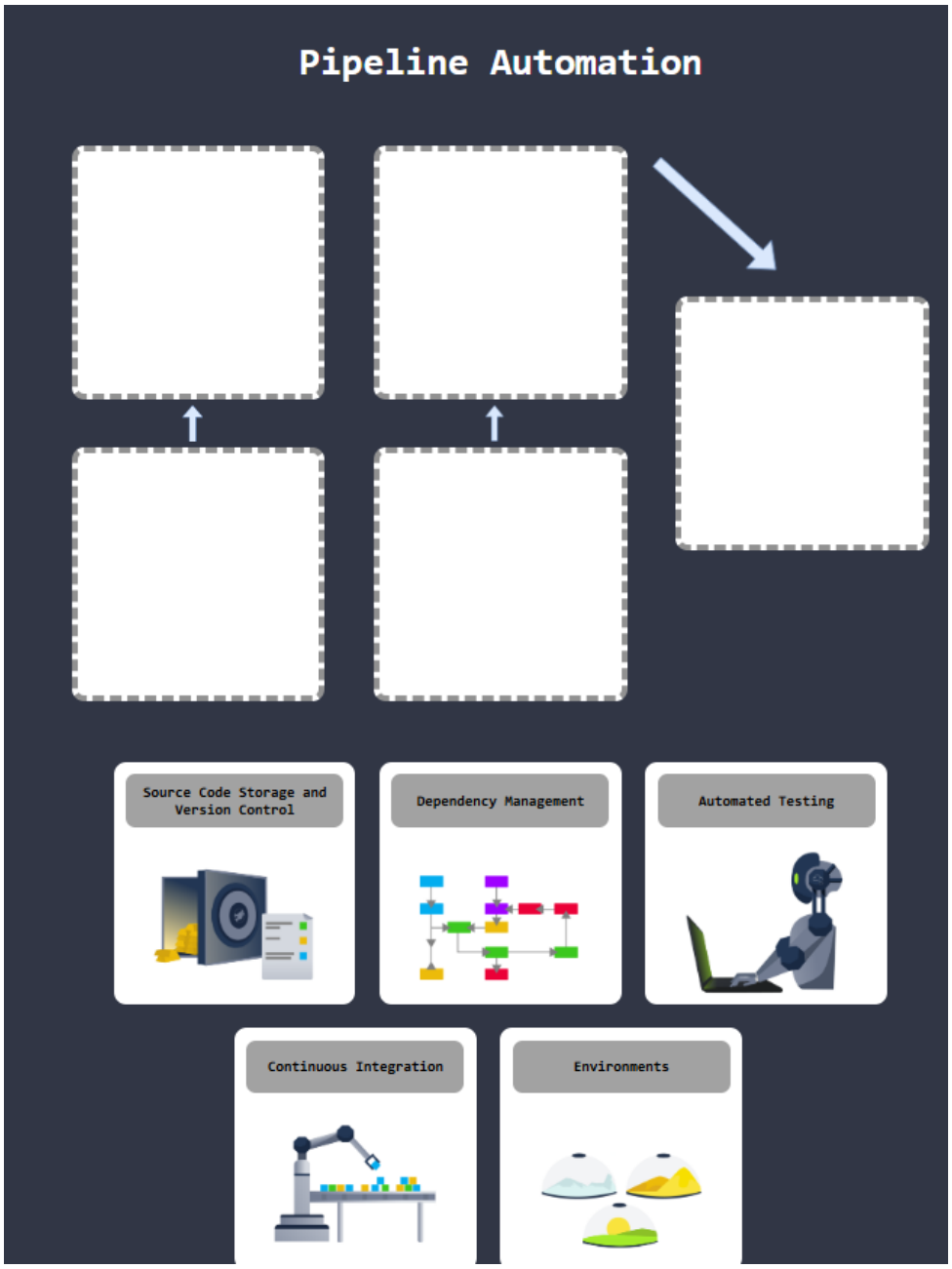
2.8. Tarea 8 - Reto

En esta tarea debemos crear nuestro propio pipeline, usaremos lo aprendido para determinar qué preocupaciones son válidas en cada etapa del pipeline.

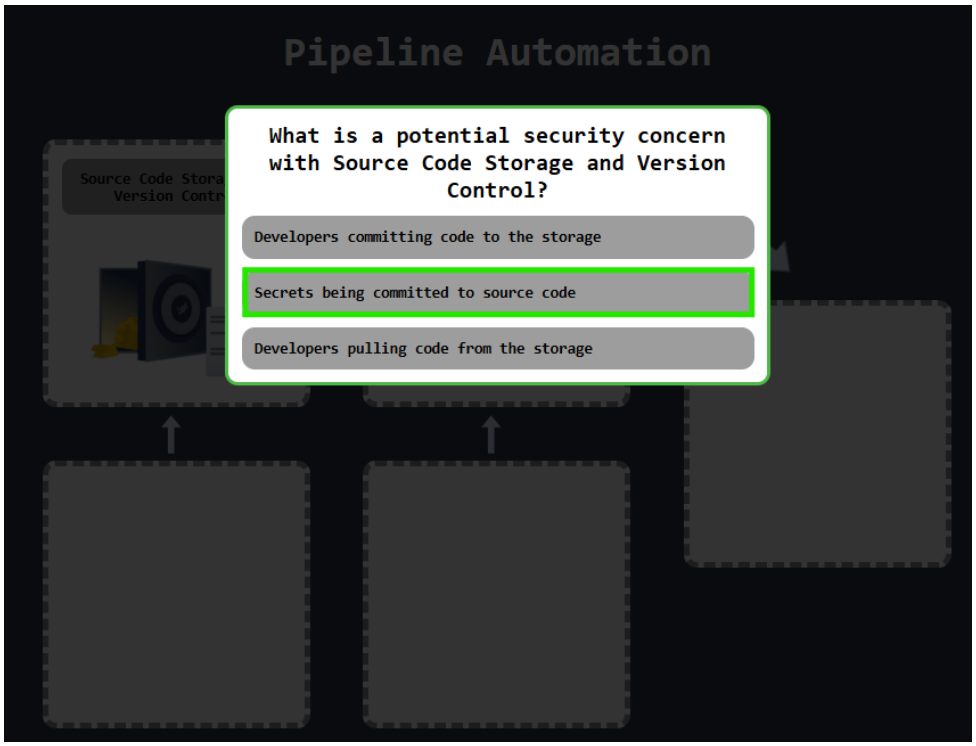
Pregunta: What is the flag received after successfully building your pipeline?

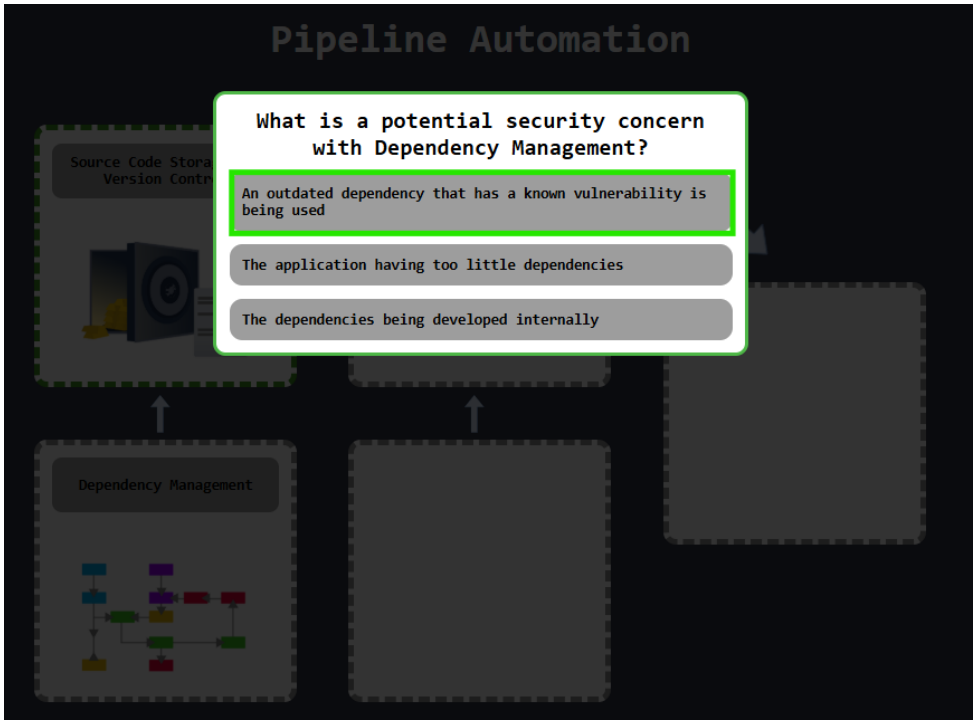
Para comenzar debemos desplegar el sitio, para ello, haremos clic en **View Site** en el lado superior de la tarea.

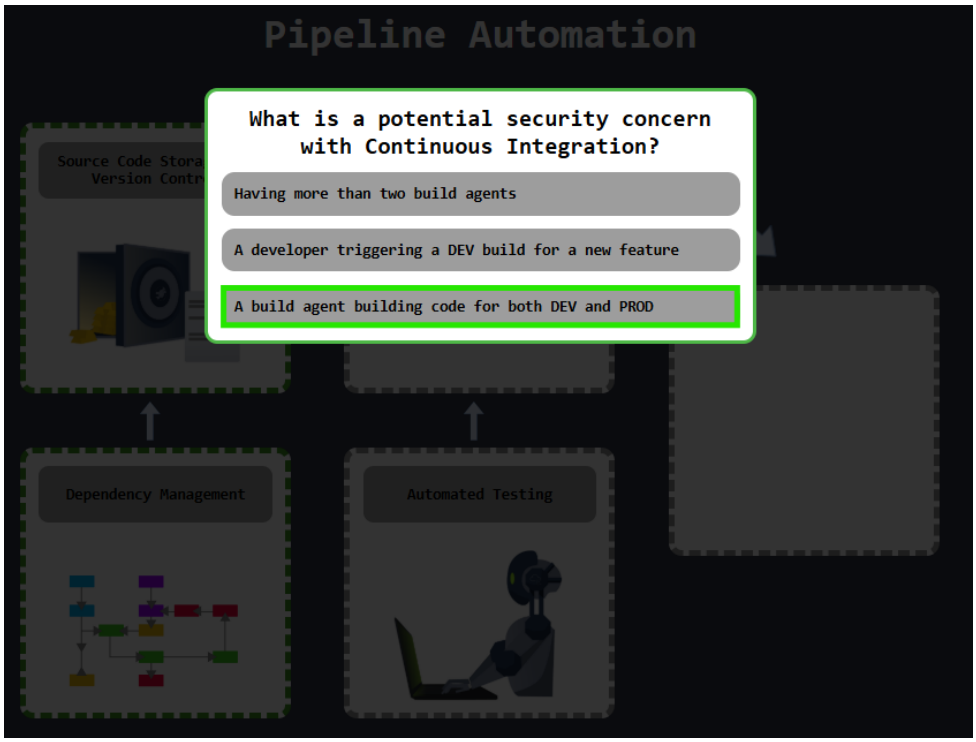


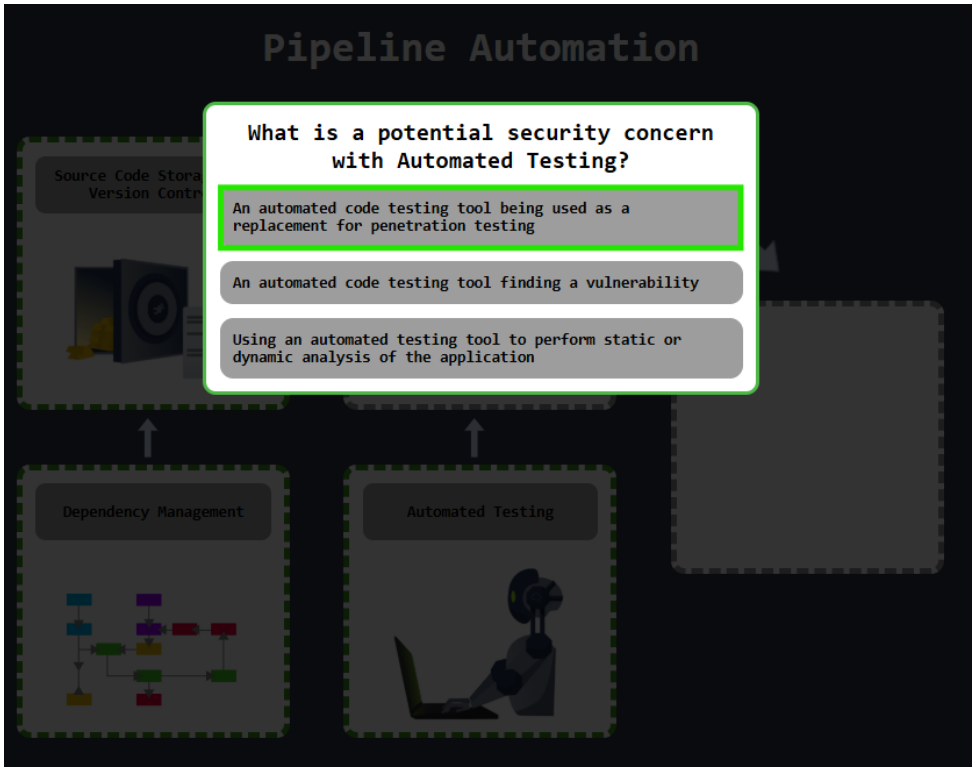


Una vez desplegado nuestro sitio, debemos arrastrar y soltar las fichas de imagen en el lugar correcto del diagrama y responder a las preguntas para lograr obtener la flag.

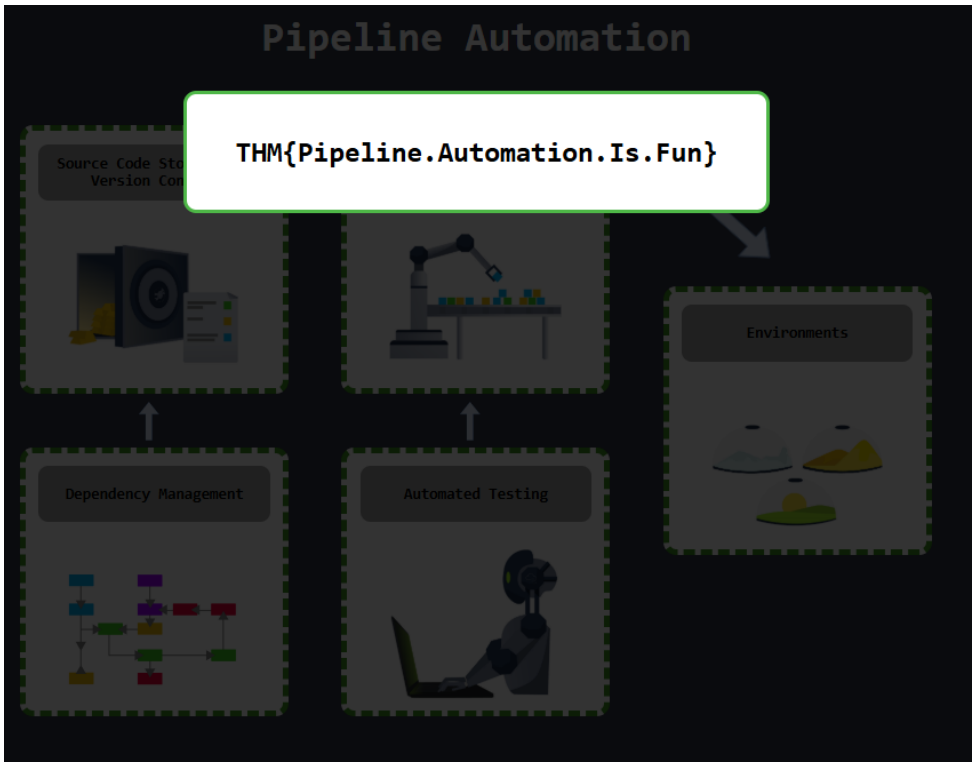

Pipeline Automation
What is a potential security concern with Source Code Storage and Version Control?
Developers committing code to the storage
Secrets being committed to source code
Developers pulling code from the storage


Pipeline Automation
What is a potential security concern with Dependency Management?
An outdated dependency that has a known vulnerability is being used
The application having too little dependencies
The dependencies being developed internally
Dependency Management


Pipeline Automation
What is a potential security concern with Continuous Integration?
Having more than two build agents
A developer triggering a DEV build for a new feature
A build agent building code for both DEV and PROD
Source Code Stora
Version Contr
Dependency Management
Automated Testing


Pipeline Automation
What is a potential security concern with Automated Testing?
An automated code testing tool being used as a replacement for penetration testing
An automated code testing tool finding a vulnerability
Using an automated testing tool to perform static or dynamic analysis of the application
Source Code Stora
Version Contr
Dependency Management
Automated Testing

Después de completar el diagrama y responder las preguntas, obtendremos la flag en pantalla.



Respuesta: **THM{Pipeline.Automation.Is.Fun}**

2.9. Tarea 9 - Conclusión

Recapitula que, si bien la automatización acelera el desarrollo, también amplía la superficie de ataque y reafirma la importancia de integrar seguridad en cada etapa del pipeline para mitigar estos riesgos.

Pregunta: I understand the basic pipeline structure, and I'm ready to do a deep dive into each element!

Respuesta: **No requiere respuesta** (Hacemos clic en **Submit**).

3. Conclusión sobre la Sala

Al finalizar la sala, no solo habremos aprendido cómo automatizar la entrega de software, sino también cómo integrar controles de seguridad en cada fase, reduciendo la superficie de ataque y fortaleciendo el flujo completo de desarrollo.