Informe de Laboratorio

Detección y Remediación de Secretos en Repositorios con Gitleaks

Alexander Moreno
Sebastián Cisneros
Ricardo Tobar

Universidad Remington
ingeniería de Sistemas
línea de énfasis

Introducción

En los entornos de desarrollo colaborativo, la gestión de la seguridad de la información es un aspecto crítico. Con frecuencia, los repositorios de código contienen datos sensibles como contraseñas, tokens o claves de API que, si se exponen, pueden generar riesgos graves para la organización. Para mitigar esta amenaza, existen herramientas como Gitleaks, diseñadas para detectar de manera automatizada secretos dentro de proyectos versionados con Git.

El presente laboratorio tiene como propósito simular un escenario real de exposición de secretos en un repositorio local, utilizando Gitleaks para realizar la detección, análisis y posterior remediación. Además, se exploran prácticas complementarias como la configuración de .gitignore, la eliminación de archivos comprometidos y la limpieza del historial con BFG Repo-Cleaner. De esta manera, se busca reforzar la importancia de mantener repositorios seguros y libres de información confidencial, aplicando controles preventivos y correctivos en el ciclo de vida del software.

Funcionamiento técnico

El funcionamiento técnico de Gitleaks se basa en la detección automatizada de secretos mediante reglas y expresiones regulares que permiten identificar patrones comunes de credenciales, contraseñas o claves en un repositorio. La herramienta puede ejecutarse en dos modos principales: escaneo del historial de Git, revisando todos los commits y ramas, o análisis del workspace actual, verificando únicamente los archivos presentes en el directorio de trabajo. Durante el proceso, Gitleaks examina línea por línea los archivos y genera reportes estructurados (como JSON) que incluyen información detallada sobre la regla que activó la alerta, el archivo afectado y el fragmento comprometido. Cabe destacar que Gitleaks solo realiza la detección, siendo necesaria la intervención manual para la remediación, como la eliminación de archivos, actualización de .gitignore o la limpieza del historial con herramientas adicionales como BFG Repo-Cleaner.

Demo controlada

Detección y Mitigación

La **detección de secretos** en el repositorio se realizó mediante el uso de **Gitleaks**, ejecutando escaneos tanto en el historial de commits como en el espacio de trabajo actual. Los reportes generados en formato JSON permitieron identificar con precisión el tipo de secreto expuesto, su ubicación en los archivos y el contexto en el que fueron detectados. Esta fase evidenció el riesgo de mantener credenciales sensibles dentro del código y la importancia de contar con un mecanismo de control automatizado que alerte oportunamente a los desarrolladores.
En cuanto a la **mitigación**, se aplicaron diversas acciones para reducir la exposición y evitar futuros incidentes. Primero, se eliminaron los archivos comprometidos del repositorio y se

actualizó el archivo .gitignore para prevenir que información sensible volviera a ser versionada. Posteriormente, se realizó un commit de limpieza y, como medida complementaria, se empleó **BFG Repo-Cleaner** para remover los secretos de todo el historial de Git, asegurando que no quedaran rastros en versiones anteriores. Finalmente, se ejecutó un nuevo escaneo con Gitleaks, confirmando que ya no existían filtraciones, lo cual validó la efectividad del proceso de remediación aplicado.

Conclusiones

El laboratorio permitió evidenciar la importancia de contar con herramientas de seguridad en el ciclo de vida del software, ya que la exposición de secretos en repositorios Git representa un riesgo crítico para la integridad y confidencialidad de la información. Gitleaks demostró ser una solución eficaz para la detección temprana de credenciales sensibles, facilitando la identificación de vulnerabilidades tanto en el historial de commits como en los archivos actuales del proyecto.

Asimismo, el proceso de remediación resaltó que la eliminación de secretos no se limita a borrar archivos de la versión vigente, sino que requiere también la limpieza completa del historial para garantizar que no existan rastros que puedan ser explotados. El uso complementario de herramientas como BFG Repo-Cleaner, junto con buenas prácticas como el uso de .gitignore y gestores de secretos, fortalecen la seguridad de los repositorios.

En conclusión, mantener repositorios libres de información confidencial no solo depende de las herramientas, sino también de la conciencia y disciplina de los desarrolladores, quienes deben adoptar medidas preventivas y correctivas. Este ejercicio contribuyó a comprender cómo integrar controles de seguridad en el flujo de trabajo y cómo responder de forma efectiva ante incidentes de exposición de datos.