



ESCUELA POLITÉCNICA NACIONAL  
FACULTAD DE INGENIERÍA DE SISTEMAS  
CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE



**Nombre:** Belén Cholango.

**Curso:** GR2SW.

**Taller 08-09-10**

**Noticia escogida:** Vulnerabilidad en Unity (CVE-2025-59489)

**Fuente:** *Steam and Microsoft warn of Unity flaw exposing gamers to attacks* — BleepingComputer (6 Oct 2025)

**Link:**

<https://www.bleepingcomputer.com/news/security/steam-and-microsoft-warn-of-unity-flaw-exposing-gamers-to-attacks/>

## 1. Resumen del Caso

En octubre de 2025 se divulgó una vulnerabilidad de seguridad crítica en el motor de desarrollo de videojuegos Unity y fue identificada como CVE-2025-59489. Esta falla reside en el componente runtime del motor y permite que un atacante ejecute código arbitrario o acceda a información sensible en dispositivos que ejecuten juegos o aplicaciones construidos con versiones vulnerables del motor (desde Unity 2017.1 en adelante).

Plataformas como Steam y Microsoft hicieron advertencias de esta vulnerabilidad. Steam implementó actualizaciones en su cliente para mitigar posibles ataques y recomendó a los desarrolladores actualizar sus juegos o usar versiones parchadas, mientras que Microsoft sugirió que los usuarios desinstalen temporalmente títulos vulnerables hasta que reciban correcciones.

Unity publicó parches de seguridad y solicitó de forma urgente a los desarrolladores actualizar el editor, recompilar y redistribuir sus juegos o aplicaciones usando versiones que solucionan la vulnerabilidad.

## 2. Clasificación del Mantenimiento

Este caso corresponde principalmente a mantenimiento correctivo, definido como la actividad de corregir defectos o errores que ya existen en el software una vez este se ha entregado o distribuido. En este caso, la vulnerabilidad representa un fallo de seguridad que, si no se corrige, puede ser explotado por atacantes para ejecutar código malicioso en los dispositivos de los usuarios.

Aunque también puede incluir aspectos de mantenimiento preventivo, por ejemplo, la acción de actualizar versiones antiguas del motor antes de que se conviertan en una amenaza más activa. El foco principal fue corregir una falla concreta que ya estaba presente en cientos de juegos y aplicaciones.

### 3. Procesos de SCM Involucrados

#### Control de Versiones (VCS)

El uso de un Control de Versiones como Git o el sistema interno de Unity para versionado de su engine fue esencial para:

- Identificar las versiones vulnerables del motor de juego (desde Unity 2017.1).
- Crear una versión corregida del engine que elimina la vulnerabilidad (commits en ramas de mantenimiento).
- Publicar las versiones parcheadas para que los desarrolladores puedan migrar a ellas.
- Permitir a los equipos de desarrollo de juegos aislar los cambios críticos de seguridad sin afectar otras líneas de desarrollo (mediante el uso de *branches* específicos de parche).

#### Gestión de Cambios (Change Management)

El proceso formal de SCM definió cómo se gestionó esta corrección:

- Reporte y evaluación de la vulnerabilidad por parte de un investigador de seguridad (Ryotak de GMO Flatt Security).
- Aprobación de la solución, que incluyó desarrollar un parche y validarlo.
- Revisión por pares, pruebas y publicación del parche de seguridad.
- Coordinación con plataformas externas (Steam, Microsoft) para aplicar mitigaciones inmediatas mientras los juegos se actualizan.

Este flujo demuestra cómo SCM no es solo “usar Git”, sino coordinación de cambios formales con trazabilidad y control. Con un buen SCM fue posible saber exactamente qué artefactos necesitaban cambios y dónde aplicar las correcciones.

### 4. Impacto en el Ciclo de Vida del Desarrollo de Software (SDLC)

El evento impactó varias fases del SDLC:

- **Desarrollo:** Los equipos que usan Unity tuvieron que actualizar el motor, recompilar sus proyectos y generar nuevos builds de sus juegos. Esto implica cambios en el entorno de desarrollo y manejo de dependencias.
- **Pruebas:** Antes de liberar builds parcheados, los desarrolladores tuvieron que realizar pruebas de regresión para comprobar que la actualización no introdujera fallas adicionales.
- **Despliegue:** La corrección requirió un despliegue de emergencia en plataformas de distribución (Steam, Microsoft Store, etc.). Algunos juegos incluso fueron removidos temporalmente de tiendas hasta que recibieron una versión segura.

## **5. Beneficios del SCM**

El buen sistema de Gestión de Configuración de Software permitió:

- Trazabilidad: Identificar exactamente cuáles versiones del motor y qué juegos estaban afectados por la vulnerabilidad.
- Control y estabilidad: Las ramas de mantenimiento o control de versiones permitieron aplicar parches sin interferir con desarrollo en curso.
- Colaboración: Equipos de desarrollo distribuidos pudieron coordinarse para corregir el problema y distribuir actualizaciones en múltiples proyectos.
- Reducción de errores: La disponibilidad de parches específicos redujo el riesgo de introducir nuevos errores durante la corrección.