OPINIÓN / DESPUÉS DE LA PANDEMIA

# *Vulnerabilidades de software: ¿el cuento de nunca acabar?*

por Enrique Dans · 15 diciembre, 2021 - 06:07 GUARDAR

La aparición de una vulnerabilidad de seguridad que afecta a muchísimos sistemas informáticos, conocida como Log4Shell, y que ha sido calificada como "**la vulnerabilidad más importante y más crítica** de la última década, y posiblemente la más grande en la historia de la informática moderna" está sacudiendo a los departamentos de tecnología de la gran mayoría de las compañías, que en muchos casos han tenido que pasarse el pasado fin de semana trabajando a toda velocidad. Revelada el pasado jueves 9 de diciembre en un tweet, ahora eliminado, por un ingeniero de seguridad de Alibaba, la vulnerabilidad ha sido aprovechada y convertida en herramienta de ataque en un tiempo absolutamente récord.

**¿Qué diablos es Log4Shell** y por qué es tan crítica? Fundamentalmente, porque permite el acceso sencillo y sin ningún tipo de contraseña a prácticamente cualquier servidor. El problema radica en una librería de código abierto desarrollada en Java por la Apache Software Foundation, Log4j, y utilizada en prácticamente todas partes para dejar constancia del comportamiento de un sistema (*logging*). ¿Por qué está presente en tantos sitios? Simplemente, porque hace su trabajo, lo hace razonablemente bien, y la primera regla de la ingeniería de software es tan sencilla como "no reinventar la rueda".

¿Qué ocurre cuando, en un sistema como este, utilizado en casi todas partes, surge una vulnerabilidad? Lógicamente, **depende de quién la encuentre y de cómo la comunique.** Si se trata de un sistema de código cerrado propiedad de una compañía, por ejemplo, y quien lo descubre es la propia compañía, lo habitual es que lo corrija, aunque si nadie más lo sabe, tampoco es que suelan hacerlo con una prisa loca, y que publique una actualización, o 'parche', que permita a los usuarios actualizarse. Si quien lo descubre es un tercero, todo depende de lo que quiera hacer con su descubrimiento: desde simplemente reportarlo a la compañía, hasta intentar explotarlo para fines que nunca suelen ser nada bueno.

> *La primera regla de la ingeniería de software es tan sencilla como 'no reinventar la rueda'*

Si el problema está en un programa de código abierto, en donde suele haber muchas menos vulnerabilidades porque, como dice la ley de Linus, enunciada por Linus Torvalds, creador de Linux, "dado un número suficientemente elevado de ojos, todos los errores se vuelven obvios", la cuestión está en quién lo encuentra, cómo lo comunica, quién lo corrige, y a qué velocidad. Una vez conocida la vulnerabilidad, se establece una **carrera entre los propietarios de sistemas** y los que intentan explotarla, carrera en la que llegar de los últimos puede tener un importante castigo.

En este caso, la vulnerabilidad está ya corregida, pero dada la enorme difusión de esta librería, el problema es si todos los afectados van a ser capaces de aplicar el correspondiente parche a tiempo para evitar ser atacados.

Un ataque, aunque suene violento, es en realidad algo muy sencillo e imperceptible: lo único que tiene que hacer un atacante es lograr enviar un mensaje para que el sistema lo registre, y que ese mensaje contenga instrucciones para acceder a un sitio gestionado por el atacante, donde el sistema hará y ejecutará lo que le digan, sea robar datos, ejecutar programas para minar criptomonedas, o lo que sea.

**Esta crisis de seguridad es muy seria**, ofrece enormes oportunidades a atacantes de todo tipo, y afecta a prácticamente todo el mundo. Lo que demuestra, fundamentalmente, es que **vivimos en un mundo en el que todos los sistemas tienen cadenas de dependencia** que incluyen todo tipo de piezas, como enormes castillos construidos, a veces, sobre auténticos alfileres. ¿Pero tiene realmente que ser así?

La realidad es que sí. No hay otra manera de desarrollar este tipo de sistemas que no sea apoyándose en componentes que han hecho otros: si intentásemos crearlo todo desde cero, sería no solo imposible, sino además, mucho peor. Podemos reducir razonablemente la incidencia de este tipo de problemas si hacemos las cosas bien, ordenadas, y almacenando un registro de todo el *software* que utilizamos, incluyendo las librerías, las librerías que usan esas librerías, las dependencias, etc. hasta el último componente.

El problema, por tanto, no es intentar construir *software* a prueba de bombas, porque siempre aparecerá alguna, sino tratar de que esas bombas, cuando aparezcan, sean simplemente un petardo: unas cuantas horas de trabajo, aplicar una actualización, y ya.

Valga el problema para que, como mínimo, **aprendamos a apreciar el trabajo de quienes se encargan de construir y mantener los sistemas** de los que dependen un número creciente de nuestras actividades. Casi todo lo que hacemos se apoya en el *software*, pero el *software* es un conjunto enorme de componentes y dependencias en los que siempre puede surgir algún problema o alguna vulnerabilidad.

Lo que tenemos que hacer no es intentar vivir en un mundo ideal en el que estas cosas no pasen -porque pasarán- sino **aprender a reaccionar de forma eficiente cuando aparezcan.**

SIGUE LOS TEMAS QUE TE INTERESAN
COLUMNAS DE OPINIÓN · SOFTWARE · TECNOLOGÍA

COMENTA

MÁS DE ENRIQUE DANS

- Fuga de cerebros
  8 diciembre, 2021 - 02:52
- Twitter y la segunda generación de directivos en las tecnológicas
  1 diciembre, 2021 - 06:06
- Los derechos de los antivacunas
  24 noviembre, 2021 - 06:06
- Renovables: presente y único futuro
  17 noviembre, 2021 - 06:16

LOS ÚLTIMOS

NOTAS DIGITALES

Apostar por la industria 4.0 y las tecnologías exponenciales
Paco Bree

GODIVACIONES

Ni Sánchez, ni Casado, ni Scholz, ni Biden: sin líder para salir de la crisis
María Blanco

Convergencia Digital ESG

Navidad y regalos: acertar con la sostenibilidad
Lluís Altés

Impulso digital

El 5G o el reto de España para mantenerse en el podio europeo de la conectividad
Alfonso Muñoz Fernández

Ahora en portada




Sé el primero en comentar

Escribe tu comentario

NORMAS DE USO

ENVIAR