

El desastre OpenSSL de Debian

CAJA NEGRA

Descubrimos lo que podemos aprender del desastre OpenSSL de Debian. **POR KURT SEIFREID**

Después de la caída de un avión se inicia un proceso de investigación. Estas investigaciones revelan que la mayoría de estos accidentes se deben a un fallo humano o a alguna confluencia de circunstancias que no se habían anticipado previamente. Consecuentemente, viajar en avión es una de las formas más seguras de viajar por milla por pasajero.

Con el software no ocurre lo mismo. A diferencia de lo que ocurre cuando cae un avión, cuando se produce una caída de software, una respuesta típica es simplemente tratar de inmediato el problema con un parche del código fuente, el cual no resuelve los problemas subyacentes.

Es decir, nos encontramos en un estado constante de tratar sólo los sínto-

mas, pero nunca los la verdaderas causas de los desastres del software.

Como estos problemas subyacentes no se corrigen jamás, continuamos viendo los mismos fallos una y otra vez (creación de ficheros temporales, desbordamientos de búfer, desbordamientos de pila, etc).

Este mes investigaré el desastre OpenSSL de Debian en un esfuerzo por desenterrar los problemas de raíz.

En pocas palabras, el problema sobrevino cuando un mantenedor de un paquete Debian para OpenSSL ejecutó Valgrind, una herramienta de análisis de código, contra OpenSSL y encontró algunos usos de memoria no iniciada. El uso de memoria no iniciada es particularmente peligroso porque no tenemos idea de lo que podría contener:

todo a 0, todos a 1, o un código inyectado por un atacante, sólo por nombrar algunas posibilidades.

El mantenedor fue luego online hasta la lista de correo de openssl-dev y preguntó sobre el siguiente código:

```
MD_Update(&m,buf ,j) ;
```

Algunas respuestas posteriores, el mantenedor decidió que era seguro eliminar el ofensivo código del paquete OpenSSL de Debian. Estos cambios se incorporaron a Debian y la vida continuó de forma habitual.

El código en cuestión ocurría dos veces: una vez en *ssleay_rand_bytes()* y una vez en *ssleay_rand_add()*. Aunque es idéntico, cumple dos funciones muy distintas. En *ssleay_rand_bytes()*, simplemente devuelve los datos al azar de la memoria a un búfer. Sin embargo, en la función *ssleay_rand_add()*, intenta hacer algo más ingenioso, añadiendo alguna memoria no inicializada al depósito entrópico. En el mejor caso, añade a entropía, y en el peor, no rompe nada.

Este búfer se usa como fuente primaria de entropía para cualquier aplicación usando OpenSSL (a menos que usen una fuente PRG a medida). Eliminandolo por completo con un comentario, el desarrollador eliminó virtualmente toda la aleatoriedad usada durante la creación de la clave por la mayoría de las aplicaciones. Los únicos datos al azar que quedaban usados durante la creación de la clave eran la ID del proceso, reduciendo el espacio de la clave de 2^{largo} (número largo, como 128 ó 1024) a 2^{16} (o menos, en algunos casos). ¡Ups!

¿Qué Fue Mal?

Algunas cosas fueron mal y, como en la mayoría de los desastres, todo pudo haber salido bien si la cadena de eventos se hubiera roto en cualquier punto. Pero no fue así, y cada administrador de sistemas con Debian en sus máquinas tuvo que parchear cada ordenador y

regenerar cada clave de encriptación creada en los dos últimos años.

Uno de los problemas inmediatos era que existía código que hacía algo innecesario de manera potencialmente peligrosa. Añadir memoria no inicializada a la aleatoriedad serial verdadera no ayuda mucho y hace que el código parezca que está haciendo algo totalmente diferente.

Además, el código no estaba bien documentado. Por ejemplo, hubiera sido apropiado el siguiente comentario:

```
01 /* Este código es crítico
02 * lee de un depósito
03 * de datos aleatorios,
04 * casi todo el azar
05 * crítico de OpenSSL
06 * viene de aquí.
07 * Si juegas con él
08 * romperás OpenSSL
09 * y todas las aplicaciones
10 * que lo utilicen.
11 */
```

Esta es la razón por la que los equipos militares tienen grandes avisos como “este lado hacia el enemigo”.

Lo más probable es que, si este código fuera fácil de comprender, el desarrollador no tendría necesidad de ir a la lista de correo de openssl_dev para buscar ayuda. Esto nos conduce al segundo problema: comunicaciones confusas.

Cuando se trata con problemas de código relacionados con seguridad, es crítico comunicar con la gente o foro adecuados. De lo contrario recibiremos lo que parece una respuesta con autoridad aunque sea incorrecta, o bien porque la pregunta no se comprendiera o porque la respuesta era errónea. En el caso que nos ocupa, todo lo que podría haber ido mal, fue mal.

El equipo de OpenSSL reivindica que la pregunta se hizo en la lista de correo errónea, mientras que otra gente reivindica que la supuestamente lista de correo correcta no está anunciada.

Además, la pregunta tenía algo que no estaba claro: El pedazo de código del ejemplo no ofrece un contexto completo, y a causa de la naturaleza del código, podría haber llevado a un malentendido de lo que exactamente estaba ocurriendo.

Probablemente sea más efectivo disponer un prefacio a nuestro correo con

algo como, “Si conoces a alguien o a otro foro mejor, por favor, déjame conocerlos”, en vez de “Este es el sitio adecuado para preguntar”, lo que es ciertamente mejor que remitir un correo a ciegos y esperar una respuesta autorizada o fidedigna. Adicionalmente, con un poco de investigación se pueden comprobar el CVS (o subversión, o git) y los mensajes para encontrar quién remite muchos de los cambios en el código afectado para deducir quién es probablemente responsable del código en cuestión.

Como puede verse, localizar a la persona adecuada puede ser una paliza, así que documentando claramente cómo contactar – y con quién – sobre varias cuestiones ayudará mucho en la prevención de problemas con nuestro software.

Cambios de Código

Como los empaquetadores de Debian mantienen sus propios repositorios de código, el cambio del código fuente no lo notó nadie fuera del proyecto Debian, a pesar de lo cual tuvo una amplia distribución. Si el parche del código fuente se hubiera enviado oficialmente corriente arriba a OpenSSL a través del Proyecto Debian, probablemente habría hecho sonar alarmas y habría sido eliminado. Esto desemboca en una situación en la cual incluso si el proyecto continúa corriente arriba para actualizar y mantener el software, un único parche mantenido por Debian podría introducir una sutil – y, en este caso, un importante – fallo. Sin embargo, la solución de enviar todos los parches corriente arriba para su inclusión no es simple. Otra posibilidad es remitir oficialmente todos los parches corriente arriba para revisión.

Carente de Comprobación

Finalmente hablaremos del problema más grave. En la industria aeronáutica, los materiales, diseños y, a menudo, componentes completos (como montajes de alas) se testean hasta el punto de su destrucción para ver qué soportan antes de fallar. Por lo que sé, una infraestructura de comprobación para OpenSSL que genera un número de claves estadísticamente importante – por ejemplo, algunos cientos de miles o millones – y luego las analiza para com-

probar su aleatoriedad serial no existe oficialmente.

Adicionalmente, incluso si una infraestructura existió, necesitará aplicarse regularmente a nuevas versiones, y no sólo a la versión oficial, sino a cualquier versión que haya sufrido modificaciones aplicadas por los proveedores.

Claro que, este tipo de infraestructura de comprobación debería aplicarse a todos los productos, por ejemplo, un protocolo de comprobación de cortafuegos que aplica una variedad de rulesets (desde simples a complejas) y envía luego tráfico que intenta evadir los rulesets.

Hasta que existan estas infraestructuras, con toda seguridad los fallos serios continuarán apareciendo.

Conclusión

Desafortunadamente, es más barato en un período corto tratar simplemente los síntomas más dañinos de una ingeniería de software malo que tratar los problemas y causas subyacentes. Sin embargo, a largo plazo, esto conduce a importantes cantidades de tiempo gastadas para los usuarios, que han de aplicar parches y actualizaciones, y para los desarrolladores, que necesitan resolver los mismos problemas una y otra vez.

Las buenas noticias son que muchas de las soluciones a estos problemas no son caras, y la mayoría requieren poca tecnología para su implementación.

Simplemente comentando código, documentando canales de comunicaciones y realizando preguntas con claridad y con todo el contexto que sea posible, llegaremos lejos. También es importante recordar que el código abierto no es sólo acceder al código fuente, sino el acceso a la cultura que escribe el código fuente, lo que significa que todo el mundo tiene la oportunidad de ayudar a hacer que sea mucho más fácil. ■

RECURSOS

- [1] DAS-1571-1 openssl: <http://www.debian.org/security/2008/dsa-1572>
- [2] Key rollover: <http://www.debian.org/security/key-rollover/>
- [3] SSLkeys: <http://wiki.debian.org/SSLkeys>
- [4] Informe del bug OpenSSL: <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=363516>