Google ha anunciado que su cambio hacia lenguajes seguros para la memoria, como Rust, dentro de su estrategia de seguridad por diseño, ha provocado una disminución en el porcentaje de vulnerabilidades relacionadas con la memoria descubiertas en Android, que han pasado del 76% al 24% en un periodo de seis años.

El gigante tecnológico explicó que enfocar el desarrollo en prácticas de codificación segura para nuevas características no solo disminuye el riesgo general de seguridad de una base de código, sino que también hace que la transición sea más *«escalable y rentable».*

Con el tiempo, esto reduce las vulnerabilidades de seguridad de la memoria, ya que el desarrollo de código inseguro para la memoria disminuye después de un cierto tiempo y el desarrollo de código seguro gana terreno, según explicaron Jeff Vander Stoep y Alex Rebert de Google en una publicación.

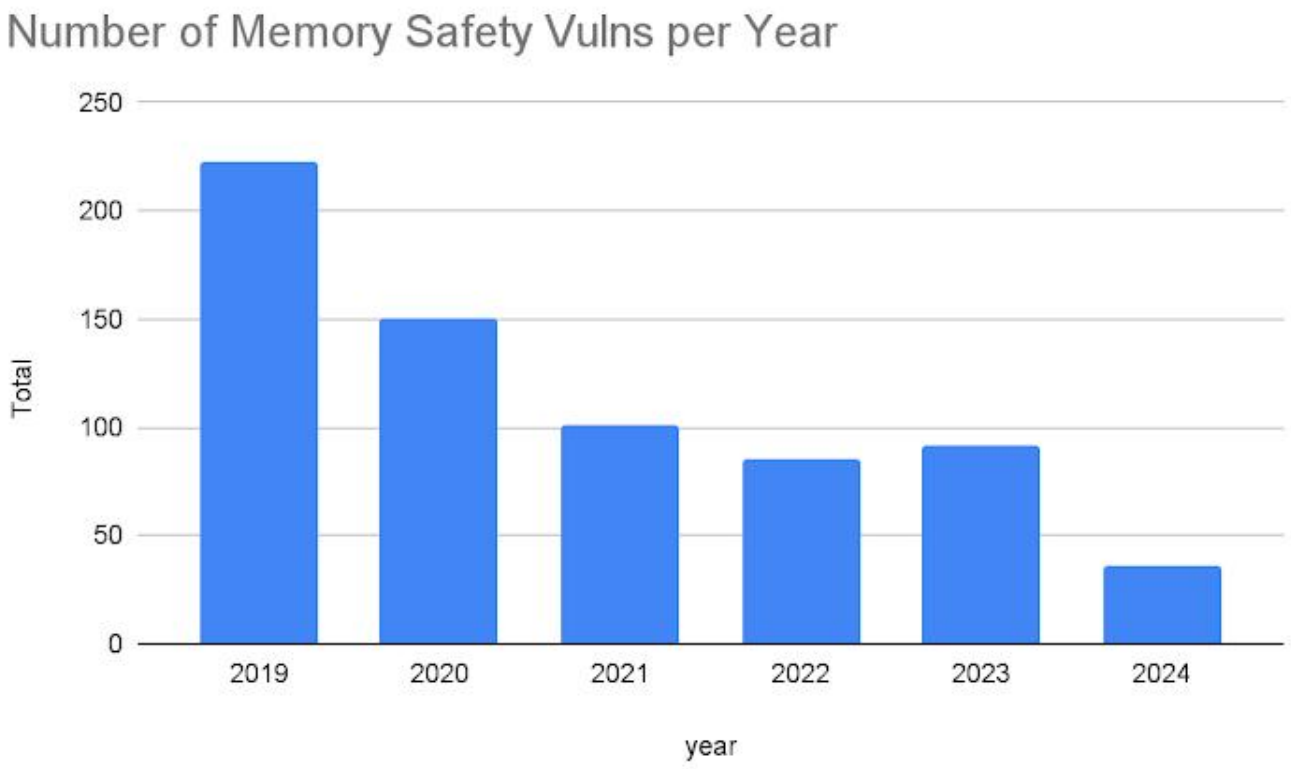
Curiosamente, el número de vulnerabilidades de seguridad de la memoria también puede reducirse, incluso si aumenta la cantidad de código nuevo que no es seguro para la memoria.

Esta paradoja se aclara por el hecho de que las vulnerabilidades tienden a reducirse de manera exponencial, ya que un estudio demostró que la mayoría de las vulnerabilidades se encuentran en código nuevo o recientemente modificado.

> *«El problema está principalmente en el código nuevo, lo que exige un cambio profundo en la forma en que desarrollamos software. Con el tiempo, el código madura y se vuelve más seguro, lo que significa que los beneficios de grandes inversiones como las reescrituras se reducen a medida que el código envejece»,* destacaron Vander Stoep y Rebert.

Google, que anunció oficialmente en abril de 2021 su intención de admitir Rust como lenguaje de programación en Android, señaló que empezó a priorizar la migración del nuevo desarrollo a lenguajes seguros para la memoria alrededor de 2019.

Como resultado, la cantidad de vulnerabilidades de seguridad de la memoria descubiertas en el sistema operativo Android disminuyó de 223 en 2019 a menos de 50 en 2024.




Cabe destacar que gran parte de esta reducción de fallos se debe a los avances en las formas de enfrentarlos, pasando de aplicar parches de manera reactiva a mitigar proactivamente y descubrir vulnerabilidades utilizando herramientas como los sanitizadores de Clang.

Google también subrayó que las estrategias de seguridad de la memoria deben evolucionar aún más, enfocándose en la *«prevención de alta confianza»* al incorporar principios de seguridad por diseño que integren la seguridad desde la base.

> *«En lugar de enfocarnos únicamente en intervenciones específicas (mitigaciones, fuzzing) o usar el desempeño pasado para predecir la seguridad futura, la codificación segura nos permite hacer afirmaciones sólidas sobre las propiedades del código y prever lo que puede o no puede ocurrir según esas propiedades»,* afirmaron Vander Stoep y Rebert.

Además, Google indicó que su enfoque se centra en ofrecer interoperabilidad entre Rust, C++ y Kotlin en lugar de realizar reescrituras completas de código, adoptando una *«estrategia práctica e incremental»* para integrar lenguajes seguros para la memoria y eliminar, a largo plazo, clases enteras de vulnerabilidades.

> *«La adopción de prácticas de codificación segura en el nuevo código representa un cambio radical, que nos permite aprovechar la reducción natural de las vulnerabilidades con el tiempo, incluso en sistemas grandes y existentes»,* concluyó Google.

El concepto es claro: al cerrar la fuente de nuevas vulnerabilidades, estas disminuyen de manera exponencial, lo que hace que nuestro código sea más seguro, mejora la eficiencia del diseño de seguridad y reduce los problemas de escalabilidad relacionados con las estrategias actuales de protección de la memoria, permitiendo que se apliquen de forma más efectiva y precisa.

Este avance se produce mientras Google destaca una mayor colaboración con los equipos de ingeniería de seguridad de productos y de unidades de procesamiento gráfico (GPU) de Arm, con el objetivo de identificar fallas y mejorar la seguridad general del software y firmware de las GPU en el ecosistema de Android.

Entre los hallazgos se incluyen dos problemas de memoria en la personalización del código del controlador de Pixel (CVE-2023-48409 y CVE-2023-48421) y otro en el firmware de las GPU Arm Valhall y en la arquitectura de firmware de las GPU de 5.ª generación

(CVE-2024-0153).

> *«Realizar pruebas proactivas es una buena práctica, ya que puede ayudar a detectar y solucionar vulnerabilidades antes de que sean aprovechadas»*, comentaron Google y Arm.