Rooteo en Android

Rivera García Mauricio y Ruiz Flores Laura Andrea

Rooteo: ¿qué es? ¿cuál es su objetivo?

A rasgos generales, se trata de una técnica por la cual es posible obtener un control privilegiado sobre un dispositivo Android. Para hablar sobre control privilegiado es necesario comentar un poco de donde surge el sistema operativo de Android.

Android es un sistema operativo desarrollado principalmente para dispositivos móviles, el cual fue desarrollado tomando como base el kernel de Linux y diversos soportes en código abierto. Android, al venir de un entorno Unix, llega a tener ciertas características que comparte con los sistemas Linux tradicionales, entre ellos, una separación de *privilegios*.

La separación de privilegios básicamente dice que habrá dos tipos de usuarios en el sistema: un superusuario y un usuario. Los usuarios tienen bajos privilegios, y si se da el caso de que quieran instalar alguna herramienta para el equipo, necesitarán del permiso del superusuario. Y hablando de estos, los superusuarios tienen todos los privilegios a usar en el sistema, estos usuarios pueden realizar actualizaciones, instalar herramientas sin permisos o correr cualquier comando en un intérprete. Podría decirse entonces que un superusuario tiene un completo control sobre el sistema.

Y así como esto funciona en Linux, también funciona en Android, pero un tanto diferente. En Linux es muy fácil decir que el superusuario es el propietario del dispositivo en donde este corra este entorno, ya que se tienen todos los permisos habilitados en primera instancia y es posible también asignar ciertos privilegios a otros usuarios que empleen el equipo. Pero en Android no es exactamente así, ya

que de forma predeterminada el sistema no tiene activados los privilegios del superusuario, o en palabras más simples, todo lo que se puede hacer en un nuevo dispositivo Android es, por defecto, limitado. Lo cual quiere decir que técnicamente uno no es el verdadero dueño de ese dispositivo.

Es aquí en donde entra el rooteo, el cual tiene como principal objetivo acceder a estos privilegios de administrador.

Mecanismos de protección de Android

Una de las formas más conocidas para correr una versión alterada del sistema operativo es por el bootloader y el recovery system. El bootloader es una de las primeras actividades que un sistema ejecuta para dar arranque al sistema operativo. Este mecanismo es desarrollado por los fabricantes del sistema, en donde hay algunos dispositivos móviles en donde es posible, mediante una combinación de teclas a la hora de encender el móvil, acceder al recovery system en donde es posible instalar versiones alternativas de Android modificadas por terceros. Aún así, hay muchos dispositivos móviles en donde este acceso al bootloader está bloqueado. Esto hace que el móvil solo pueda ejecutar versiones del sistema que el fabricante desee, de tal manera que se evita que el dispositivo móvil pueda ser alterado a nivel del sistema operativo.

Otro método es utilizando el Android Debug Bridge o ADB, el cual es una línea de comandos para Android en el cual se puede acceder a un intérprete en el cual pueden ejecutarse una variedad de comandos sobre el dispositivo. Aún así, sucede que incluso estos comandos se encuentren limitados dependiendo del parámetro definido por `ro.secure` en el cual…

- … sí `ro.secure=0` entonces podrán ejecutarse comandos con privilegios de superusuario.
- … si `ro.secure=1` sólo podrán ejecutarse comandos con los privilegios de un usuario estándar.

Esto se puede revisar en ADB usando el comando `getprop ro.secure`

Por lo general, este parámetro no puede ser modificado, por lo hacer uso de ADB dependerá si el fabricante no estableció `ro.secure=1`

En entornos Linux es posible acceder a privilegios de superadministrador mediante `su` y `sudo`, la cual hace determinadas llamadas al sistema que pueden otorgar estos privilegios. Este par de comandos nunca han estado presentes en la configuración de fábrica del sistema operativo en los dispositivos Android, lo que imposibilita que un programa pueda ejecutarse en modo privilegiado y así acceder a estos privilegios de superadministrador.

Además, la UI de Android es un mecanismo de protección ya que las aplicaciones con las que se puede interactuar no pueden tener acceso privilegiado ni ejecutarse en modo privilegiado, o empezar un programa que se ejecute en modo privilegiado. Si el sistema se mantiene así, tiene alta inmunidad a los intentos de escalada de exploits.

Los exploits

Tomando un poco de la información de antes, se puede llegar a lo siguiente:

- Si el bootloader está desbloqueado, para rootear solo bastaría realizar la combinación de teclas necesaria e instalar toda la configuración deseada (es decir, una versión del sistema operativo alternativa) y ejecutarlo desde el recovery system.
- Si `ro.secure=0` se puede utilizar ADB e instalar `su` y `sudo` en el dispositivo, montar el sistema a modo lectura/escritura para posicionarse sobre la raíz, de tal manera que ahora es posible hacer uso de estos en el dispositivo en cuestión

Pero en el caso de que el bootloader está bloqueado y `ro.secure=1` se pueden hacer uso de las muchas actividades que realiza el sistema por defecto en el

usuario root, añadiendo un código arbitrario dentro de estos procesos, de tal manera que este código está siendo ejecutado en superadministrador. Si suponemos que este código arbitrario es la instalación de `su` y `sudo` sobre el sistema, entonces prácticamente el dispositivo ya estaría rooteado.

Hay muchas formas de realizar esto, de tal manera que surgieron los exploits. Algunos exploits populares son:

- **Exploit y GingerBreak:** Este exploit aprovecha una falta de verificación de verificación de las fuentes. Digamos que el kernel manda directamente un proceso desde el root a una aplicación con permisos de superadministrador. Para esto tiene que pasar por determinados buses. La arquitectura que se encarga de esto es de *Netlink,* la cual cuenta con una vulnerabilidad en sus estructura, y es que tiene un bus génerico del cual puede llegar información desde diferentes destinos (como del root o desde otra aplicación) hacia alguna aplicación. Eso quiere decir que una aplicación externa podría ejecutar algún proceso hacia otra aplicación con privilegios de superadministrador. Básicamente, Exploid y GingerBreak aprovechan esta vulnerabilidad en la arquitectura. Exploid sustituye en el proceso init (un proceso de inicio del sistema operativo) una ruta desde “hotplug” que lee y escribe sobre “data”, siendo básicamente esto un exploit. Gingerbreak en cambio aprovecha el manejador de volumen, en donde en un fragmento de código también se escriben instrucciones arbitrarias.
- **RageAgainstTheCage y ZimperLich:** Aprovecha un fallo en `setuid(uid);` al momento de llegar a un límite establecido de procesos. Entonces, se hace una “inflación” de procesos para llegar a dicha vulnerabilidad y con la ayuda de ADB obtiene los permisos de superusuario, esto creando varios procesos y ejecutando ADB al final para obtener los permisos sobre el límite establecido.

- **KillingInTheNameOf**: Aprovechan el sistema de memoria compartida de Android para modificar las protecciones de memoria de procesos y escribir sobre el atributo `ro.secure`, cambiarlo a cero, y permitiendo el rooteo.

Ventajas del rooteo

- Mayor control del dispositivo

Al rootear el dispositivo, tenemos un mayor control de éste. Un ejemplo de esto es que podemos visualizar ciertos "packages" de las aplicaciones que están ocultos normalmente en un dispositivo sin rootear.

- Instalación de apps adicionales

Se pueden instalar aplicaciones con características especiales en dispositivos rooteados, se conocen como “root apps”. Estas aplicaciones, a diferencia de las aplicaciones que se pueden instalar en Android, ocupan permisos de superusuario.

- Más características y opciones de personalización.

Al instalar ROMs personalizadas se pueden obtener mejores características que las dadas por el sistema operativo instalado por el fabricante.

Desventajas del rooteo

- Compromete la seguridad de tu dispositivo

Cada aplicación tiene su propia sandbox con IDs de usuario para cada aplicación. Esta separación de IDs garantiza que una aplicación, con este ID en ejecución, no pueda acceder a los recursos de otras aplicaciones con un ID de usuario diferente en el mismo dispositivo. Por ello, en un dispositivo rooteado, una aplicación maliciosa con acceso al root no tendrá esa limitación y podrá acceder a los datos de otras aplicaciones, por ejemplo a los SMS, registro de llamadas, pantallas de bloqueo, contactos, etc.

- Brickeo del dispositivo

Un dispositivo se dice que está “brickeado” cuando sufre un daño en el software y queda inutilizable, dejándolo como un “ladrillo”. Por lo que realizar un rooteo de manera incorrecta puede llevar al dispositivo a este estado.

- Pérdida de la garantía

La mayoría de fabricantes no ofrece soporte a dispositivos rooteados, por lo que, aún al tener garantía en el dispositivo, se podrá requerir que se pague la reparación en caso de falla.

Sobre la protección de los fabricantes

La garantía de los dispositivos se ve afectada por el rooteo, ya que la anula. Sin embargo, si el dispositivo tiene una afectación en el hardware por error de fábrica y no tiene relación con el rooteo, la garantía puede seguir aplicando. Aún con esto, el rooteo también puede afectar el hardware, por lo que el fabricante deberá realizar las pruebas necesarias para garantizar que el fallo no se deba al rooteo y poder aplicar la garantía. Un ejemplo de esto es si se hace overclocking al procesador del dispositivo, si llega a sufrir daño la garantía no será aplicada.

Otra medida es que, con el descubrimiento de nuevos exploits, los fabricantes refuerzan sus mecanismos para que en nuevos dispositivos se vuelvan inutilizables. Por otro lado, al ser una medida cara y que los dispositivos en el mercado no están al corriente con las nuevas actualizaciones de software, es cuestión de tiempo que nuevos exploits sean descubiertos.

Los fabricantes protegen mucho estas cuestiones por cuestiones de seguridad de sus dispositivos, pero también puede darse por otras cuestiones legales. Un ejemplo de esto es Wi-Fi Theter, el cual es capaz de hacer que el móvil funcione como un modem y compartir internet con otros dispositivos, la cuestión con esta aplicación es que sale un poco de los servicios que puede proveer una compañía

de internet o telefónica, lo cual se traduce en que se pueden evitar ciertos gastos. Otra circunstancia parecida podría ser lo que ocurre entre Google y Huawei, donde podrían usarse aplicaciones de Google haciendo uso de una ROM modificada en un dispositivo Huawei.

Referencias:

Ji, Chuan. (2011) *How Rooting Works: A Technical Explanation of the Android Rooting Process*

Enlace:
https://jichu4n.com/posts/how-rooting-works-a-technical-explanation-of-the-android-rooting-process/

Srinivasa Rao Kotipalli y Mohammed A. Imran (2016) *Hacking Android.*

Enlace:

https://www.google.com.mx/books/edition/Hacking_Android/j86qDQAAQBAJ

Documentación de Android Debug Bridge (adb)

Enlace:

https://developer.android.com/studio/command-line/adb

Jon Oberheide y Zach Lanier (2011) *Don't Root Robots!*

Enlace:

https://jon.oberheide.org/files/bsides11-dontrootrobots.pdf

Rashid-Feroze. (2023). *Linux Privilege Escalation Guide.*

Enlace:

https://payatu.com/blog/a-guide-to-linux-privilege-escalation/

Luque S. (2016). *Los fabricantes, el root y la garantía, un mundo de confusión*

Enlace:

https://www.xatakandroid.com/sistema-operativo/los-fabricantes-el-root-y-la-garantia-un-mundo-de-confusion

Long Nguyen-Vu, Ngoc-Tu Chau, Seongeun Kang y Souhwan Jung. (2017)

Android Rooting: An Arms Race between Evasion and Detection.

Enlace:

https://www.hindawi.com/journals/scn/2017/4121765/