UNIVERSIDAD
DE GRANADA

Universidad de Granada
ETS de Ingeniería Informática y Telecomunicaciones.
Ingeniería en Informática.

Escalado de privilegios

Eric Adrián Mendoza Martínez - men15002@uvg.edu.gt
Matilde Cabrera González - mati33198@gmail.com
Jorge Soler Padial - solerpadial@gmail.com
Gonzalo Frontela Hernández - gonzalofrontela@gmail.com

Documento presentado para la asignatura de: SEGURIDAD EN SISTEMAS OPERATIVOS.
20/12/2018

A. Índice

B. Introducción

Estamos en una época en la que los negocios se han a movido hacia el automatizado de tareas, esto significa que el número de computadoras en la empresa que guardan información sensible aumenta, por lo que se hace evidente la necesidad de asegurar los sistemas informáticos. Dicha necesidad aumenta cuando se combina con la necesidad de ser accesibles a través de una red insegura como el internet.

Los servicios que corren en ordenadores conectados a Internet son un objetivo para atacantes que comprometen su seguridad. Esto puede llevar a un acceso no autorizado a fuentes o datos sensibles. Los servicios que requieren privilegios especiales para su operación son extremadamente sensibles, en donde un error de programación puede permitir a un atacante obtener estos privilegios y abusar de ellos.

El grado de escalado depende de los privilegios que el atacante consiga y pueda obtener mediante un ataque exitoso. Por ejemplo, un error de programación dentro del proceso de verificación limita al atacante con los privilegios que el sistema otorgue después de ser verificado dentro de la plataforma, evitando el grado de escalado después de dicha verificación. Por otro lado, un atacante remoto que consigue privilegios de superusuario sin una autenticación, presenta un escalado de privilegios mayor.

Para los servicios que forman parte de una infraestructura crítica de Internet es particularmente importante protegerlos contra errores de programación. A veces estos servicios necesitan conservar privilegios especiales durante todo su funcionamiento. Estas operaciones requieren privilegios especiales duraderos ya que se les puede solicitar en cualquier momento de su operación durante la conexión SSH. Por lo tanto en las implementaciones actuales de SSH, un error de programación explotable permite a un atacante obtener privilegios de superusuario.

Se han propuesto varios enfoques para ayudar a prevenir problemas relacionados con errores de programación. Entre ellos están “type-safe languages” y mecanismos de sistemas operativos como la protección de dominios o el confinamiento de aplicaciones. Sin embargo, estas soluciones no se aplican a muchas aplicaciones existentes escritas en C corriendo en sistemas operativos genéricos Unix. Además los servicios que autentican a los usuarios son difíciles de confinar ya que la ejecución de operaciones privilegiadas depende del estado interno que el sandbox no conoce.

Nosotros en este trabajo vamos a hablar del tema del escalado de privilegios y algunos de los motivos que pueden dar pie a este problema, así como algún conocido ataque en distintos sistemas.

C. Escalado de privilegios

Como su nombre lo indica, el escalado de privilegios consiste en obtener privilegios superiores a los que se debería tener. Generalmente, los atacantes buscan vulnerabilidades que puedan aprovechar para obtener el control de un sistema informático, o bien intentan comprometer las cuentas de usuario de la organización objetivo. Dentro de los métodos que se utilizan para obtener acceso al sistema informático se encuentra el uso de exploits (fallos) en una determinada versión de kernel, utilizar los privilegios que un servicio del sistema tiene, o aprovechar la mala configuración de los trabajos cron. Una vez el hacker logra acceso a la organización, estudiará progresivamente el sistema que está comprometido para ganar más privilegios de los obtenidos inicialmente (escalado de privilegios). Esto lo logra accediendo a información sensible de otras cuentas, como por ejemplo, documentos root en ubuntu. Y esto con el fin de llegar a obtener el control total de nuestro sistema, pudiendo quedar el sistema informático a disposición de los hacker. (Craig, 2016)

Los ataques de escalado de privilegios se realizan para sacar algún tipo de beneficio financiero, social o político de alguna organización, es decir, está enfocado a empresas y organizaciones. Por lo tanto, las organizaciones deberían utilizar y hacer cumplir alguna política de autorización, como controles de acceso, permisos y privilegios de usuario. Dichas políticas están destinadas a proteger de accesos, uso compartido, modificación o eliminación no autorizados de información de la organización.

Tipos de escalado de privilegio

El escalado de privilegios se puede realizar de diferentes formas o una combinación de las que se describen a continuación.

Horizontal

El escalado de privilegios horizontal consiste en moverse entre cuentas que tienen la misma cantidad de privilegios. Por lo general, se inicia el proceso en una cuenta que tiene privilegios bajos, y lo que se busca es poder encontrar en alguna de las cuentas a las que se tiene acceso información sensible que permita moverse verticalmente. (Treaster et al., s.f.)

Vertical

Con el escalado vertical se busca el control total del sistema informático. Se empieza con una cuenta de usuario comprometida y se es capaz de ampliar o elevar los privilegios de usuario hasta obtener los privilegios administrativos completos. (Treaster et al., s.f.)

Temporal

Este tipo de escalado hace referencia a privilegios que logran ser obtenidos temporalmente debido a algún mecanismo de seguridad que impide su escalado permanente. Muchas veces los privilegios obtenidos temporalmente se utilizan para llegar al último peldaño del escalado de privilegios, ósea, el escalado permanente. (Craig, 2016)

Permanente

El escalado permanente se obtiene cuando ningún factor de tiempo removerá los privilegios obtenidos con los métodos anteriores. (Craig, 2016)

Métodos de ejecutarlo

No se pueden listar todos los métodos para lograr el escalado de privilegios en un sistema, sin embargo, la mayoría de ataques de escalado de privilegios caen en alguna de las siguientes categorías.

Kernel exploits

Los exploits en el kernel consisten en programas que aprovechan pequeñas vulnerabilidades en el kernel objetivo. El objetivo de dichos programas es poder obtener permisos de ejecución elevados. La facilidad de estos exploits es que se pueden encontrar bases de datos llenas de exploits y solo basta con compilarlos y luego ejecutarlos en la máquina objetivo (Treaster et al., s.f.).

Si se puede ejecutar código como un usuario no privilegiado el procedimiento que debería llevar a cabo para aprovechar esta técnica, sería la siguiente:

1) Engañar al kernel para que ejecute nuestro programa en modo kernel
2) Manipular los datos del kernel
3) Lanzar un nuevo shell con privilegios root

Para que se puedan llevar a cabo los pasos anteriores, se deberían cumplir las siguientes condiciones:

1) El kernel debe ser vulnerable
2) Debemos contar con un exploit que coincida con el kernel vulnerable
3) Debemos tener la habilidad de poder transferir el exploit al objetivo
4) Debemos poder ejecutar código en la máquina objetivo

La mejor forma de evitar este tipo de ataques es que el kernel se mantenga actualizado. Si no existe la posibilidad de actualizar el kernel, entonces se puede proceder a evitar que se transfieran ejecutables o archivos al sistema objetivo. De esa manera, el kernel ya no sería vulnerable. Por lo tanto, los administradores deberían de enfocarse en restringir cualquier programa que habilite el intercambio de archivos, como FTP o wget. Y si fuera necesario utilizar alguno de dichos programas, se debe restringir su uso a cierto usuario o a algún directorio.

Explotar servicios que son ejecutados como root

Los sistemas operativos hacen uso de servicios para todo. Por lo tanto, no es raro que si no se configuran dichos servicios correctamente, se puedan explotar para obtener privilegios. Los ataques EternalBlue y SambaCry (descritos más adelante) explotaron el servicio *smb* que generalmente se ejecuta con privilegios root. Con solo un exploit, un atacante puede obtener ejecución remota de código y escalamiento de privilegios locales. Este exploit se usaba mucho para difundir ransomware por todo el mundo, lo cual es una combinación mortal.

Otro de los mayores errores que cometen los administradores web es ejecutar un servidor web con privilegios root. Una vulnerabilidad de inyección de comandos en la aplicación web puede llevar a un atacante a root shell. Este es un ejemplo clásico de por qué nunca debe ejecutar ningún servicio como root a menos que sea realmente necesario (Jaafar et al., 2016).

Para evitar esta vulnerabilidad, solo se debe realizar una verificación de cómo están siendo ejecutados los servicios del sistema. Y si alguno tiene privilegios root, debe ser relanzado sin privilegios.

Explotar ejecutables SUID

SUID representa un cambio de privilegios para un usuario de bajos privilegios a alguno de mayor rango (en muchos casos, root). Esta característica de Linux permite la ejecución de un archivo con permisos de un usuario específico. Un ejemplo común de esto es el ejecutable *ping*, que necesita utilizar un socket de red para poder ejecutarse. Si se marcara el programa ping como SUID con el dueño del comando como root, este se ejecutaría con privilegios root en cualquier momento que alguien con privilegios limitados lo ejecute.

SUID es una característica que, usada de la manera apropiada, puede ayudar a mejorar la seguridad en Linux. El problema es cómo los administradores pueden introducir, sin saberlo, configuraciones peligrosas cuando se instala una aplicación de terceros o cuando se hace un cambio lógico de configuraciones. Un gran número de administradores no comprenden dónde colocar el bit SUID y donde no colocarlo. Este bit no debe de ser escrito, especialmente, en los editores de archivos por el hecho de sobreescritura de cualquier archivo presente en el sistema (Jaafar et al., 2016).

La mejor manera de evitar este inconveniente es no fijar el bit SUID para ningún programa que permita escapar al shell. Tampoco se debe fijar dicho bit para ningún editor, compilador o intérprete, porque un atacante podría acceder a un archivo existente, modificarlo y ejecutarlo maliciosamente.

Explotación de los servicios sudo

Si un atacante no puede obtener acceso, de manera directa al root, puede tratar de comprometer cualquier usuario que sea *sudoer*. Una vez que tenga acceso de cualquier usuario sudoer, básicamente puede ejecutar cualquier comando con privilegios root (Jaafar et al., 2016).

Como contra medida, los administradores pueden limitar los comandos que los usuarios pueden ejecutar con sudo, sin embargo, con esta configuración igual se pueden introducir vulnerabilidades desconocidas las cuales pueden terminar en escalado de privilegios.

Un ejemplo común de esta práctica es que se pueda ejecutar *find* con sudo con el fin de que los usuarios puedan encontrar archivos específicos dentro del sistema. El problema radica en que *find* acepta distintos parámetros que podrían ser explotados por un atacante para ejecutar comandos con privilegios root.

Explotar los trabajos cron

Cron es un demonio que se utiliza para programar la ejecución de tareas en un momento específico. Como en los casos anteriores, este puede configurarse de manera inapropiada, por lo que se recomienda realizarse las siguientes preguntas antes de crear uno:

- ¿Algun script o binario en el cron tiene permisos de escritura?
- ¿Podemos escribir un archivo de cron?
- ¿El directorio cron.d tiene permisos de escritura?

Los trabajos cron generalmente se corren con privilegios root. Si podemos, de manera exitosa, manipular algún script o binario que está definido como un “cron job”, entonces podemos ejecutar arbitrariamente código con privilegios de root (Jaafar et al., 2016).

Para evitar que los trabajos cron sean explotados, todos los scripts ejecutados por la tarea programada no deberían ser escribibles, el archivo cron solo debería poder ser editable por el root y el directorio cron.d solo debería ser modificable por el root.

D. Vulnerabilidades del Kernel de Linux

Como se mencionaba en el apartado anterior, los exploits al kernel de linux son un método común para realizar un escalado de privilegios en un sistema. Por lo tanto se realizará una descripción de las formas de abordar este tipo de ataques.

Missing pointer checks.

El kernel omite chequeos de access_ok o hace un uso indebido de operaciones "más rápidas" como __get_user, las cuales no validan el valor de los punteros provistos por los usuarios o índices de variables para asegurar que apuntan solo a su espacio de usuario en la memoria. Estos bugs permiten a procesos no privilegiados leer o escribir en lugares arbitrarios de la memoria, llevando a corrupción de la misma, divulgación de información, DoS o escalado de privilegios si el proceso controla qué datos escribe.

Missing permission checks

El kernel realiza operaciones privilegiadas sin chequear si el proceso que llama tiene permiso para realizarlo. Una vulnerabilidad de esta categoría desemboca en una violación de la política de seguridad del kernel. Los ataques que pueden explotar esta vulnerabilidad dependen de qué política de seguridad sea, pudiendo ir desde una ejecución de código arbitrario o escalado de privilegios, hasta sobreescribir un archivo "append-only" (al que solo se pueden añadir cosas).

Buffer overflow

El kernel chequea incorrectamente los límites superior o inferior cuando accede al buffer, asigna un buffer más pequeño de lo que debería, utiliza funciones que manipulan strings inseguras, o define variables locales que son demasiado grandes para la pila del kernel. Los ataques que pueden explotar esta vulnerabilidad son, corrupción de memoria (escritura) o ataques de revelación de información (para lectura). Un atacante puede montar ataques de escalado de privilegios sobreescribiendo punteros de funciones cercanas y alterando la integridad del control de flujo de datos del kernel.

Integer overflow

El kernel realiza una operación con enteros incorrecta, resultando en un desbordamiento de enteros, o un error de signo. El atacante puede engañar al kernel para que use el valor incorrecto para asignar o acceder a memoria, permitiendo ataques similares a los mencionados en las vulnerabilidades de buffer overflow.

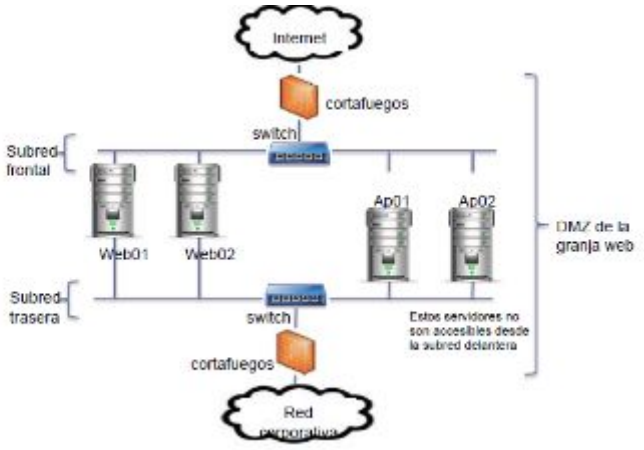
E. Contramedidas al escalado de privilegios

La mejor forma de evitar que cualquier intento de ataque con escalado de privilegios funcione es aplicando configuraciones robustas en el sistema. Algunas de las mejores prácticas se detallan a continuación.

Segregación de funciones de usuarios y del personal de IT.

En las empresas y compañías grandes son unos de los principales objetivos para los hackers, es habitual manejar datos de clientes de especial sensibilidad, o datos de la propia compañía de gran valor, es por ello que los administradores de sistemas tienen que poner especial interés en proteger sus sistemas informáticos y evitar la elevación de privilegios entre roles y sistemas, para ello:

- Las cuentas de administración del sistema y de usuario para actividades diarias serán distintas, cada una tendrá su papel y sus restricciones dependientes de sus labores, como por ejemplo, la de administración no tendrá buzón de correo, permisos de navegación.
- Entre las comprobaciones de si un administrador es legítimo se puede usar su situación geográfica. Un mismo administrador no tendrá acceso a la plataforma, el almacenamiento y el backups al mismo tiempo.
- Para las tareas de administración se deben usar máquinas aisladas y dedicadas exclusivamente para ello.
- Las vlan de administración deben estar separadas del resto, es decir, no tendrán conectividad directa al resto. Como mostramos en la figura los equipos de la red corporativa no debe tener conexión directa con los servidores. Deben crearse máquinas de salto para compartimentalizar la administración en distintos segmentos.



- Cada cuenta de servicio tendrá el mínimo privilegio que necesite para realizar su tarea. Se puede monitorizar la ubicación dentro de la red corporativa para que salte una alarma y bloquee cuando se inicie sesión en una ubicación distinta a la esperada.

- Se pueden usar herramientas de monitorización para la detección de desplazamiento de un sistema a otro y de escalada de privilegios.

Bloqueos y controles de seguridad a nivel de red

Usar un cortafuegos, comprender su funcionamiento y los beneficios del mismo. Uso de subredes privadas.

- Segmentar la red en vlans y aplicar reglas de filtrado a nivel de red y de puestos de trabajo y servidores, limitando las conexiones.
- Colocar cortafuegos entre subredes para realizar la tarea de bloqueo y filtrado de paquetes de red, inspeccionando las direcciones y puertos entre las subredes.
- Hay que permitir el tráfico autorizado y denegar el resto de tráfico, es decir, se establecerán reglas para permitir solo cierto tipo de tráfico.
- Configurar reglas relativas para controlar protocolos de aplicación a través del control de puertos. Como HTTP, FTP, ssh o telnet.
- Se puede hacer uso de servicios gratuitos comerciales que identifiquen VPNS o IPs de dudosa reputación.
- Ocultar las diferentes direcciones de la red corporativa, como por ejemplo las ip de los servidores, actuando como proxy y evitando ataques directos.
- Realización de una auditoría de seguridad periódica que no exceda los dos años.
- Agregar doble autenticación para las conexiones remotas, o verificación en dos pasos.

Medidas de seguridad a nivel de sistemas

Acciones que se pueden llevar a cabo por los distintos trabajadores de una corporación.

- Hay que mantener el software de los equipos actualizado, pero esto a veces es engorroso para las empresas, por ello hay que planear las actualizaciones de software lo antes posible, siempre en un entorno de red específico y nunca en el entorno de producción.
- Que sus usuarios o clientes utilicen el método de autenticación más fuerte posible y lo utilicen de manera inteligente (por ejemplo, con contraseñas largas, fuertes o complejas), dicho de otra forma, educar a los trabajadores para que tengan buenas prácticas en la gestión y uso de sus contraseñas. Añadir entre nuestras directivas de seguridad esas buenas prácticas, como por ejemplo un determinado tiempo de validez para las contraseñas. Comprobar periódicamente que se llevan a cabo.
- Bloqueo de herramientas de volcado de contraseñas de memoria(como la protección LSA Protección frente a mimikatz) mediante políticas de dominio (GPO) y con bloqueo de ejecución de software (applocker).
- Las comunicaciones han de ser cifradas, el cifrado de correo no es muy recomendable, casi obligatorio, es importante evitar MITM, evitar la recopilación de información de nuestra empresa es la primera medida a llevar a cabo.
- Analizar sus aplicaciones web en busca de vulnerabilidades conocidas para minimizar los ataques de abuso.

- Validar los datos en cada envío de formularios utilizado en su sitio web. Aplique esto y reducirá la exposición de su organización a los ataques de escalamiento de privilegios.

F. Ataques de escalado de privilegios

Ahora, se procederá a presentar algunos ejempplos de escalado de privilegios.

Escalado de privilegios en móviles

iOS Apple

"Jailbreak", ejemplo de escalado vertical, es el término que se acuñó para denominar el proceso de traspasar las limitaciones que el desarrollador de IOS (Apple) impone sobre el sistema operativo móvil. "Jailbreakear" un dispositivo móvil IOS permite acceso a nivel root al sistema, como en sistemas operativos basados en el kernel de UNIX. Al romper la seguridad del sistema operativo accedemos a funciones restringidas, como podría ser instalar aplicaciones no autorizadas, "retoques" y temas que no están disponibles normalmente.

Esto se consigue en los dispositivos iphone parcheando /private/etc/fstab, archivo que indica con qué configuración montar cada partición. El parche debía modificar el archivo para que la partición del sistema se montase con permisos de lectura-escritura.

El tipo de "Jailbreak" se divide en tres: "tethered" , "semi-tethered" y "untethered". El primero de ellos sólo dura en el dispositivo hasta que éste se reinicia, por lo que hay que repetir el proceso completo si el dispositivo se apaga. De ahí su nombre "tethered" (atado en español), ya que es necesario conectar el dispositivo al ordenador, porque no arrancará si no se vuelve a parchear. Los de tipo "untethered" son capaces de mantener el dispositivo parcheado sin uso de ordenador, a pesar de que el terminal se apague.

Las primeros métodos de "jailbreak" pasaron desapercibidos hasta el 2008, con la llegada del modelo Iphone 3G. Éste fue el primer modelo distribuido masivamente y que captó la atención del mundo entero. Fue entonces cuando un jóven de 17 años, George Hotz (GeoHot), consiguió romper la seguridad de iphone OS 3.0. Este logro le proporcionó fama mundial en el ámbito de la seguridad informática, lo que posteriormente le dio el suficiente reconocimiento como para conseguir trabajo en las compañías más grandes del sector.

El equipo de desarrollo del SO no tardó en tomar medidas para paliar la brecha de seguridad, lanzando una actualización del mismo muy controvertida, pues llegó a bloquear y dejar inservibles muchos de los dispositivos. Comenzó así una batalla que todavía dura, entre hackers y Apple, por romper la seguridad del sistema operativo en cada nueva actualización.

El vector de ataque usado en el "Jailbreak" de "GeoHot" consiste en sobrescribir las posiciones de memoria de 0x0 hasta 0x2000. Con esto es posible reemplazar las direcciones del vector que maneja las irq del sistema, en la posición de memoria 0x38, por la dirección del "payload" a ejecutar.

Al ejecutar la orden:

- *usb_control_msg(iDev, 0x21, 2, 0, 0, 0, 0, 1000);*

se conseguía:

- *memcpy(0, LOAD_ADDR, 0x2000);*

Esto permite reemplazar las posiciones desde 0x0 a 0x2000.

A día de hoy el software más usado para "Jailbreak" es Electra, que se basa en los exploits IOSurface (CVE-2017-13861), mptcp_usr_connectx (CVE-2018-4241) y getvolattrlist (CVE-2018-4243).

"Mptcp_usr_connectx" y "getvolattrlist", son los nombres de las funciones que hacen posible dichos exploits y comparten el mismo vector de ataque, "buffer overflow". Estos exploits permiten que se pueda ejecutar código en un contexto privilegiado, como puede ser el ring 0. Un ejemplo de esto es el código que desencadena el exploit en "mptcp_usr_connectx":

```
  if (dst->sa_family == AF_INET &&
      dst->sa_len != sizeof(mpte->__mpte_dst_v4)) {
    mptcplog((LOG_ERR, "%s IPv4 dst len %u\n", __func__,
        dst->sa_len),
       MPTCP_SOCKET_DBG, MPTCP_LOGLVL_ERR);
    error = EINVAL;
    goto out;
  }

// verify sa_len for AF_INET6:

  if (dst->sa_family == AF_INET6 &&
      dst->sa_len != sizeof(mpte->__mpte_dst_v6)) {
    mptcplog((LOG_ERR, "%s IPv6 dst len %u\n", __func__,
        dst->sa_len),
       MPTCP_SOCKET_DBG, MPTCP_LOGLVL_ERR);
    error = EINVAL;
    goto out;
  }

// code doesn't bail if sa_family was neither AF_INET nor AF_INET6

  if (!(mpte->mpte_flags & MPTE_SVCTYPE_CHECKED)) {
    if (mptcp_entitlement_check(mp_so) < 0) {
      error = EPERM;
      goto out;
    }

    mpte->mpte_flags |= MPTE_SVCTYPE_CHECKED;
  }

// memcpy with sa_len up to 255:

  if ((mp_so->so_state & (SS_ISCONNECTED|SS_ISCONNECTING)) == 0) {
    memcpy(&mpte->mpte_dst, dst, dst->sa_len);
  }
```

Se puede apreciar como se comprueba si "dst->sa_family" es igual a "AF_INET6" o a "AF_INET", pero no se tiene en cuenta si no es igual a ninguna de estas dos. Por lo que se podría llegar a la ejecución de "memcpy()" sin haber revisado los parámetros de la llamada, pudiendo provocar un ataque de "buffer overflow".

Android

Android es un sistema operativo orientado inicialmente a dispositivos móviles. Fundada como empresa en el 2003, y adquirida en 2005 por Google es actualmente el sistema operativo más extendido del mundo con unos 2,2 billones de usuarios activos mensualmente. Está basado en el kernel Linux y se pueden encontrar versiones para dispositivos móviles, relojes inteligentes, tablets, televisores y sistemas embebidos en coches.

Ser el sistema operativo más extendido mundialmente le hace ser blanco de las miradas de los expertos en seguridad informática, tanto de un bando como del otro. Solo en 2008 se han notificado y hecho públicas 471 vulnerabilidades, casi cuadruplicando a su mayor competidor IOS con 125.

La arquitectura de android se divide en tres capas. La primera capa compuesta por el kernel de linux modificado, que se encarga de la abstracción del hardware y proporciona los drivers necesarios, además de encargarse de la gestión de la red. Android al usar el kernel de linux, se expone a la mayoría de vulnerabilidades que afectan a este. La segunda capa, el middleware, se compone de librerías, que dan soporte a las aplicaciones, y de la máquina virtual Dalvik, que es la encargada de ejecutar las aplicaciones de la capa de usuario. La tercera capa es la que componen las aplicaciones de usuario, en Android están escritas en Java y son ejecutadas por la máquina virtual de Dalvik.

El kernel de linux y la máquina virtual Dalvik funcionan conjuntamente para proporcionar un entorno de ejecución seguro. El kernel de linux proporciona su estructura de usuarios que será usada para proporcionar un id único a cada proceso los cuales se ejecutan cada uno en un “sandbox” de Dalvik prohibiendo por defecto la comunicación, acceso a datos de otros procesos y hardware.

Escalado de privilegios por parte del usuario:

A diferencia de los sistemas operativos basados en el kernel linux para ordenadores, en Android el acceso root no está disponible para el usuario encargado del dispositivo. El nivel de privilegios root sólo es accesible por el kernel y aplicaciones con privilegios para ello, por lo que desde la aparición del sistema operativo móvil han surgido técnicas para hacerse con estos privilegios.

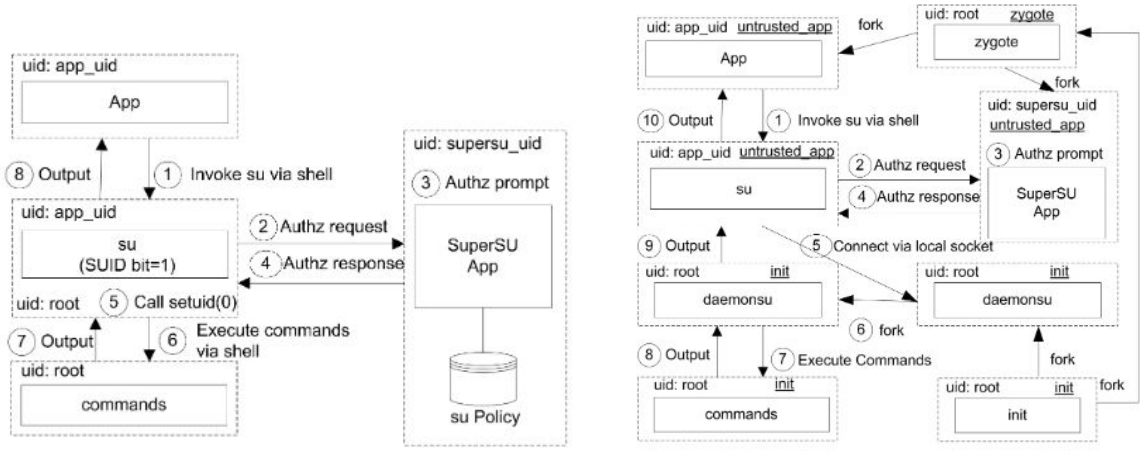
```
mount -o rw,remount /system /* remount system as  writable */
---snip--- /* place files and set permissions*/
cp_perm 0 0 06755 <src>/su /system/xbin/su
cp_perm 0 0 0755 <src>/su /system/xbin/daemonsu
cp_perm 0 0 0644 <src>/Superuser.apk /system/app/Superuser.apk
cp_perm 0 0 0755 <src>/supolicy /system/xbin/supolicy
cp_perm 0 0 0755 <src>/install-recovery.sh
                    /system/etc/install-recovery.sh
/* content of  install-recovery.sh */
#!/system/bin/sh
/system/xbin/daemonsu --auto-daemon &
```

Ejemplo de montaje del sistema de archivos de Android.

En las primeras versiones de Android el método utilizado para adquirir privilegios de root pasaba por encontrar una vulnerabilidad en el sistema que dé acceso a la partición del sistema /system. Android al usar el kernel linux es susceptible de heredar las vulnerabilidades de este. Un ejemplo de esto es “DirtyCow” vulnerabilidad del kernel de linux que se ha usado para escalar privilegios en Android.

Modificar la partición "/system" para añadir puertas que permitan adquirir acceso root en el sistema dejó de ser viable con la llegada de Android 4.3. Con esta nueva versión se introdujeron importantes mejoras de seguridad para evitar el acceso root, entre ellas montar la partición "/system" de forma que se impide a los ejecutables lanzados desde ella que obtengan un UID además de añadir SEAndroid un equivalente a SELinux. Los hackers e investigadores debieron entonces buscar otro método de conseguir acceso sin modificar el sistema, surge así "systemless root".

Este nuevo método usado a partir de Android 4.3, requiere poder modificar el bootloader del terminal. Una vez conseguido el permiso por parte del fabricante, que suele conllevar la pérdida de garantía, se añade a la secuencia "init", la primera en lanzarse dentro del kernel, y que tiene privilegios elevados, un "daemon" que recibirá las peticiones de root de las aplicaciones. Por lo que la secuencia para escalar privilegios cambió completamente de una versión a la siguiente, aunque para las aplicaciones es compatible.



Proceso de escalar privilegios en versiones anteriores a Android 4.3 y posteriores.

Adquirir privilegios root en Android puede conllevar que aplicaciones no deseadas se aprovechen de estos privilegios. Para evitar esto es común la instalación de aplicaciones intermediarias que delegan en en el usuario la decisión de dar privilegios a una aplicación que los reclama, un ejemplo es "SuperSu".

The infamous DirtyCow exploit

Vulnerabilidad que tiene su propia web, aprovecha un fallo de seguridad antiguo presente en el kernel de linux desde hace 11 años, aunque salió a la luz en 2016 con la esperanza de que sea parcheada. Está provocada por problemas en el sistema de memoria del kernel de Linux cuando se realizan ciertas operaciones de memoria, permitiendo a un usuario sin permisos utilizar esta vulnerabilidad para conseguir permisos de escritura y escalar privilegios en el sistema hasta llegar a ser root.

Código para averiguar si nuestro sistema Linux tiene esta vulnerabilidad

```
#!/bin/bash
```

```
# - Matches on source and compiled code
# - Searches in user home directories by default
# - Detects certain strings in files smaller 300 kbyte
# - Does not print anything if nothing was found
# - Appends the file's time stamp of the files in question > good indicator to
spot false positives
# - Should work on most Linux systems with bash
# Old version
# for f in $(find /home/ -type f -size -300 2> /dev/null); do if [[ $(strings
-a "$f" 2> /dev/null | egrep "/proc/(self|%d)/(mem|maps)") != "" ]];then
m=$(stat -c %y $f); echo "Contains DirtyCOW string: $f MOD_DATE: $m"; fi;
done;
for f in $(find /home/ -type f -size -300 2> /dev/null); do if [[ $(egrep
"/proc/(self|%d)/(mem|maps)" "$f") != "" ]];then m=$(stat -c %y "$f"); echo
"Contains DirtyCOW string: $f MOD_DATE: $m"; fi; done;
```

DirtyCow también afecta a Android debido a que se basa en gran parte a Linux. Es muy probable que si ejecutamos un kernel de Linux superior a 2.6.22 en nuestro Smartphone. Además, en este caso no se necesita acceso local al dispositivo, con la instalación de una aplicación maliciosa que ejecute el exploit será suficiente.

Referencia oficial a DirtyCow: CVE-2016-5195

Pasos a seguir para ejecutar esta vulnerabilidad:
Lo primero que debemos hacer ver la fuente del archivo en 40616.c, con cualquier editor de texto, vemos que este exploit funciona de la siguiente manera, notaron que hay 2 payloads de 64 y 32 bits, por default el de 64bits esta sin comentarios cabe decir que funcionara solamente para esa arquitectura, ya que el de 32 bits esta comentado y no ejercerá ninguna función. Quitamos los comentarios del archivo, luego pasaremos a compilar el archivo .c con “gcc 40616.c 4016 -pthread”. Hecho esto ya podemos subir el archivo al sistema de la víctima. Una vez tengamos el archivo ejecutado en la víctima “./40616” ya hemos escalado privilegios root y tenemos todos los permisos.

EternalBlue

EternalBlue es el nombre dado a una vulnerabilidad en el sistema operativo Windows. En esta vulnerabilidad se basa el ransomware WannaCry, Microsoft lanzó una actualización que solucionaba esta vulnerabilidad poco antes de que este ransomware surgiese, por lo que los sistemas que ya habían sido actualizados estaban protegidos pero los demás no, y éste fue el gran problema.
Esta vulnerabilidad funciona explotando el Microsoft Server Message Block 1.0. El SMB es un protocolo de transferencia de archivos en red y permite a ciertas aplicaciones en un ordenador, leer y escribir en archivos y realizar peticiones de servicios que están en la misma red.
Microsoft dice que la actualización de seguridad que emitió es crítica y después de WannaCry lanzó un parche “raro” de Windows XP, después de finalizar oficialmente el soporte para el software en 2014.

SambaCry

El paquete Samba proporciona un sistema de impresión y compartición de archivos para clientes de Windows y puede correr en la mayoría de las variantes de sistemas UNIX. Samba configura las impresoras y los recursos compartidos en red que aparecen como discos en Windows y como impresoras bajo un sistema operativo Windows.
Lanzaron RedHat 9.0 con un paquete Samba version 2.2.7a, el cual tenía una vulnerabilidad que podía ser explotada en red para ganar privilegios de superusuario.

Por tanto, al igual que hemos hablado del exploit EternalBlue, de Windows, debemos hablar del exploit de Samba (al que algunos han apodado SambaCry), que al igual que el de Windows ataca una vulnerabilidad en el paquete Samba de los sistemas Unix, esta vulnerabilidad es del tipo buffer overflow.
Teniendo en cuenta el número de sistemas vulnerables y la facilidad para explotar la vulnerabilidad, el fallo de Samba se podía explotar a gran escala. El desbordamiento del búfer se producía cuando el servicio Samba intenta copiar los datos proporcionados por el usuario en un buffer estático sin verificar.

Los sistemas del hogar con dispositivos NAS (Network-Attached Storage) también podían ser vulnerables a este fallo.

El fallo residía en la forma en que Samba manejaba las librerías compartidas. Un atacante remoto podría usar este esta vulnerabilidad del módulo de Samba para cargar una librería compartida en un recurso compartido con permiso de escritura y luego hacer que el servidor cargase y ejecutase código malicioso.

Una vez lanzado el parche que solucionaba este problema, tenías dos opciones, actualizar, o si no tenías acceso a una actualización inmediatamente, podías solucionar la vulnerabilidad agregando la siguiente línea a tu fichero de configuración de Samba, “smb.conf”: nt pipe support = no

Este cambio prevenía que los clientes accediesen a algunas máquinas de la red y deshabilitasen algunas funciones para los sistemas Windows que estuvieran conectados.

G. Conclusiones

Se concluye que a medida que se automatizan actividades comerciales y aumenta el número de ordenadores que almacenan información sensible, se hace más aparente la necesidad de sistemas seguros. Esta necesidad es mucho más aparente según los sistemas y las aplicaciones se distribuyen más y más, y se accede a ellos a través de redes no seguras, como Internet.

El Internet se ha convertido en algo crítico para los gobiernos ,empresas, instituciones financieras y millones de usuarios. Las redes de ordenadores soportan multitud de actividades cuya pérdida supondría una paralización de estas organizaciones. Como consecuencia, los temas relacionados con la ciberseguridad se han convertido en un asunto nacional. Proteger el internet es una tarea difícil.

La ciberseguridad solo se puede obtener a través de un desarrollo sistemático, no se puede conseguir de golpe. Aplicar técnicas de ingeniería de software es un paso en la dirección correcta. Sin embargo, los ingenieros de software necesitan estar al tanto de los peligros y problemas de seguridad asociados al diseño, desarrollo y despliegue de redes basadas en software, y sobre todo deben estar actualizados ya que sino se pueden ocasionar grandes problemas (como por ejemplo el ransomware WannaCry, que por una actualización tardía de los sistemas afectó a miles de computadoras). Por esto último, es que debemos tener unas sólidas políticas de seguridad, y mantenerlas lo más estrictamente posible, ya que sino alguna empresa que trate datos de terceros puede dejar expuesta su información.

H. Bibliografía

- Craig, P. (2016). *Local Privilege Escalation*. [en línea] Gsec.hitb.org. Disponible en: https://gsec.hitb.org/materials/sg2016/COMMSEC%20D2%20-%20Paul%20Craig%20-%20Local%20Privilege%20Escalation%20in%202016.pdf [Obtenido el 18 Nov. 2018].

- Treaster, M., Koenig, G., Meng, X. and Yurcik, W. (s.f.). *Detection of Privilege Escalation for Linux Cluster Security*. [en línea] University of Illinois at Urbana-Champaign (UIUC). Disponible en: https://pdfs.semanticscholar.org/1112/e58d4a899e78ada1b041e266ad42405d236d.pdf [Obtenido el 1 Dic. 2018].

- Jaafar, Fehmi & Nicolescu, Gabriela & Richard, Christian. (2016). A Systematic Approach for Privilege Escalation Prevention. 101-108. 10.1109/QRS-C.2016.17.

- Craig, P. (2016). *Local Privilege Escalation*. [online] Gsec.hitb.org. Available at: https://gsec.hitb.org/materials/sg2016/COMMSEC%20D2%20-%20Paul%20Craig%20-%20Local%20Privilege%20Escalation%20in%202016.pdf [Accessed 18 Nov. 2018].

- S. Christey and R. A. Martin, 10 de Diciembre de 2010. *Vulnerability type distributions in CVE*. [online] Disponible en: http://cve.mitre.org/docs/vuln-trends/ vuln-trends.pdf, 2007.

- Haogang Chen, Yandong Mao, Xi Wang, Dong Zhou, Nickolai Zeldovich, and M. Frans Kaashoek. 2011. *Linux kernel vulnerabilities: state-of-the-art defenses and open problems*. [online] Disponible en: https://dl.acm.org/citation.cfm?doid=2103799.2103805

- Swati Khandelwal, May 24, 2017. *7-Year-Old Samba Flaw Lets Hackers Access Thousands of Linux PCs Remotely.* [online] Disponible en: https://thehackernews.com/2017/05/samba-rce-exploit.html.

- Niels Provos, Markus Friedl, Peter Honeyman, 27 Aug. 2003. *Preventing Privilege Escalation.* [online] Disponible en: https://www.usenix.org/legacy/event/sec03/tech/full_papers/provos_et_al/provos_et_al_html/.

- Scott Brookes, May 2018. *Mitigating Privilege Escalation.* [online] Disponible en: https://search.proquest.com/openview/8e847f4ec90b7f8213d687acdf3476fd/1?pq-origsite=gscholar&cbl=18750&diss=y

- H. Güneş Kayacık, A. Nur Zincir-Heywood, 23 May 2014. *Mimicry Attacks Demystified: What Can Attackers Do To Evade Detection?*. [online] Disponible en: https://www.researchgate.net/publication/224333634 .

- Ellen Nakashima, Craig Timberg, May 16. *NSA officials worried about the day its potent hacking tool would get loose. Then it did.* [online] Disponible en: https://cyber-peace.org/wp-content/uploads/2017/05/NSA-officials-worried-about-the-day-its-potent-hacking-tool-would-get-loose.-Then-it-did.pdf

- John R. Johnson and Emilie A. Hogan, 2012. *A Graph Analytic Metric for Mitigating Advanced Persistent Threat.* [online] Disponible en: https://cybersecurity.pnnl.gov/documents/A_Graph_Analytic_Metric.pdf

- Richard A. Kemmerer, 2011. *Cybersecurity*. [online] Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.591.577&rep=rep1&type=pdf

- Levent Ertaul, Mina Mousa, 2018. *Applying the Kill Chain and Diamond Models to Microsoft Advanced Threat Analytics* . [online] Disponible en: https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/SAM9723.pdf

- Matt Burgess, 2017. *Everything you need to know about EternalBlue – the NSA exploit linked to Petya.* [online] Disponible en: https://www.wired.co.uk/article/what-is-eternal-blue-exploit-vulnerability-patch

- Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel disponible en la web oficial del sitio https://dirtycow.ninja/

- Script de ejemplo para la explotación de la vulnerabilidad, disponible en https://gist.github.com/Neo23x0/e800b698dd8739c957144722dc5195c8

- Historia de Jailbreak en IOS,Danny Key, disponible en https://medium.com/@dannykey/the-history-of-ios-jailbreaking-d1a42f48e462

- Descripción del exploit usb_control_msg(0x21,2), Wiki fundada por GeiHot, disponible en https://www.theiphonewiki.com/wiki/Usb_control_msg(0x21,_2)_Exploit

- Noticia sobre la venta del primer iphone3G jailbreakeado, Fuente de la misma Blog personal de GeoHot, disponible en https://www.wired.com/2007/08/geohot-trades-h/

- Wiki sobre Jailbreak fundada por GeoHot, disponible en https://www.theiphonewiki.com

- Repositiorio con codigo sobre CVE-2018-4241, disponible en https://github.com/GeoSn0w/Osiris-Jailbreak/tree/master/multi_path

- San-Tsai Sun,Andrea Cuadros,Konstantin Beznosov,2015 ,Android Rooting: Methods, Detection, and Evasion, disponible en https://dl.acm.org/citation.cfm?id=2808126

- Repositorio de DirtyCow sobre Android, disponible en https://github.com/timwr/CVE-2016-5195

- Davi L., Dmitrienko A., Sadeghi AR., Winandy M. (2011) Privilege Escalation Attacks on Android. In: Burmester M., Tsudik G., Magliveras S., Ilić I. (eds) Information Security. ISC

2010. Lecture Notes in Computer Science, vol 6531. Springer, Berlin, Heidelberg, disponible en https://link.springer.com/chapter/10.1007/978-3-642-18178-8_30

- Post en foro sobre systemless root, Chainfire, disponible en https://forum.xda-developers.com/showpost.php?p=63197935&postcount=2

- Nikolay Elenkov, Android Security Internals: An In-Depth Guide to Android's Security Architecture, disponible en https://books.google.es/books?hl=es&lr=&id=y11NBQAAQBAJ&oi=fnd&pg=PR5&dq=android+security&ots=nVWzBWvQ2B&sig=ZfvEUZonWdbAi-pKPLo3vCZadD0#v=onepage&q=android%20security&f=false

- Stefan Brahler, Analysis of the Android Architecture, disponible en https://www.it.iitb.ac.in/frg/wiki/images/2/20/2010_braehler-stefan_android_architecture.pdf

- Mohammed Rangwala , Ping Zhang, Xukai Zou, Feng Li , A taxonomy of privilege escalation attacks in Android applications https://pdfs.semanticscholar.org/a9a8/5f28f1fb24a9689fe4491cf781363527fa91.pdf

- Foro de desarrolladores de Android, disponible en https://www.xda-developers.com/root/