Hacking ético

Proyecto 1 - Hack-Proof Inc.

Grupo 3

Raúl Ladrón de Guevara López
Juan Manuel cumbrera
Christian Romero Oliva

Introducción

Objetivos de la presentación

- Exploración de vulnerabilidades en nuestro tema seleccionado que es Sistemas Operativos.
- Comprender en qué consisten y como pueden explotarse.
- Aumentar la conciencia sobre la importancia de medidas preventivas para garantizar un entorno digital seguro.

CVE-2016-2414 - Minikin Android

- Vulnerabilidad en *Minikin*, que es componente de Android encargado de renderizar las fuentes en la interfaz gráfica.
- Minikin tenía un error en su implementación que provocaba un error donde se intentaba escribir datos en bucle (con la tabla cmap) en una dirección de memoria inválida, lo que desemboca en una denegación de servicio (DoS)

¿Como podríamos explotarla?

Haciendo que un usuario o aplicación instalase archivo .TTF malicioso previamente modificado para causar el error.

¿Como podríamos prevenirla?

Esta vulnerabilidad se puede prevenir utilizando versiones de Android superiores a 5.0.2, 5.1.1, 6.0, 6.0.1 y también evitando utilizar archivos de fuentes no verificadas que podrían ser maliciosos.

¿Que impacto podría tener?

Si esta vulnerabilidad fuera explotada en un dispositivo el usuario perderia el control sobre el mismo y no podría acceder a sus datos, además podría perderlos debido a que el error por su naturaleza podría provocar una corrupción de memoria.

CVE-2022-2586 - Linux Kernel UAF

- Vulnerabilidad de uso después de la liberación *(UAF)* que se encuentra en el kernel de Linux
- Permite a un atacante local con privilegios provocar un fallo *UAF* tras la eliminación de una tabla, conduciendo a una escalada de privilegios
- Esta vulnerabilidad tiene su origen en el código del kernel que controla las *nf_tables*, que son utilizadas para implementar reglas del firewall así como filtros de paquetes
- Se produce la liberación de un elemento de la susodicha tabla después de su uso, lo que lo permite al atacante acceder a la memoria liberada y modificarla a su interés

¿Cómo podríamos explotarla?

- En una vulnerabilidad UAF (*Use-After-Free*), nos centramos en el proceso de manejo de memoria de un sistema, donde se libera un espacio por error
- Un exploit podría permitir acceder a ese espacio de memoria liberado en un proceso
- Esto otorgaría al atacante la oportunidad de escribir al nivel del kernel

¿Que impacto podría tener?

- Esta vulnerabilidad permite al atacante ejecutar código con privilegios dek kernel
- Esto daría prácticamente el control total del sistema al atacante
- La confidencialidad, la integridad y la disponibilidad de los datos se ven comprometidos por completo

¿Cómo se podría evitar?

Existen dos formas principales con las que un usuario de Linux puede mantenerse seguro:

- **Actualizar a la versión 5.18.11 o posterior del kernel de Linux.** Esta versión corrige la vulnerabilidad
- **Evitar ejecutar código de fuentes desconocidas.** Esto podría ayudar a prevenir la explotación de la vulnerabilidad por parte de un atacante

CVE-2023-41995 - Desbordamiento de búfer en el kernel de Linux

- La vulnerabilidad se encuentra en el subsistema de mensajería del kernel, el cual se utiliza para procesar los mensajes enviados entre los procesos.
- Se produce porque el kernel de Linux no libera correctamente la memoria después de que se haya utilizado, lo cual implica que un posible atacante pueda proporcionar un mensaje malicioso que el kernel procesará incorrectamente, provocando un desbordamiento de búfer.
- Afecta a las versiones de iOS y iPadOS anteriores a la 17 y a las versiones de macOS anteriores a la 14.

¿Cómo podriamos explotarla?

- Para explotar y hacer uso de esta vulnerabilidad, un atacante debe enviar un mensaje malicioso al subsistema de mensajería del kernel de Linux. Dicho mensaje debe ser lo suficientemente grande como para provocar un desbordamiento de búfer.

¿Qué impacto tendría?

- **Confidencialidad:** El atacante podría acceder a datos confidenciales, como contraseñas, información financiera o datos personales.
- **Integridad:** El atacante podría modificar o destruir datos.
- **Disponibilidad:** El atacante podría interrumpir los servicios o sistemas.

¿Cómo se podría evitar?

- **Actualizar el kernel a la última versión.** Las últimas versiones del kernel de Linux corrigen la vulnerabilidad.
- **Instalar un firewall.** Esto permitiría bloquear el tráfico entrante no autorizado.
- **Mantener los sistemas actualizados con las últimas actualizaciones de seguridad.**
- **Mantener actualizado el antivirus.**

CVE-2021-3156 - Sudo (Baron Samedit)

Esta vulnerabilidad consiste en que en el comando sudo de las versiones v1.8.2 - v1.9.5 permite a un usuario común obtener privilegios de root con tan solo ejecutar una instrucción sudo.

¿Como podríamos explotarla?

Con un exploit adecuado que modifique el archivo de configuración de sudo se puede lograr conseguir una escalada de privilegios de un usuario sin privilegios a root.

¿Como podríamos prevenirla?

Para prevenirnos de esta vulnerabilidad debemos de actualizar nuestro sistema ya que es una vulnerabilidad interna de los sitemas basados en UNIX

¿Que impacto podría tener?

El impacto es crítico ya que permitiría conseguir de administrador y por tanto vulnerar los datos de todas las maneras, pero primero deberíamos de conseguir un usuario en el sistema para poder llevarla a cabo.

CVE-2023-26604 - Systemd 246

Esta vulnerabilidad se basa en que el daemon systemd no establece un parámetro para evitar que mediante el uso del comando less se ejecuten comandos. Esto sumado a los prilegios que tiene systemd de root provoca que se puedan conseguir privilegios de root fácilmente.

¿Como podríamos explotarla?

Para explotarla necesitaríamos tener una versión anterior a la 247 de systemd y poder ejecutar con sudo el comando systemctl. Por ejemplo podríamos hacer:

```
sudo /usr/bin/systemctl status cron.service
```

y en medio de la ejecución de less, activar por ejemplo una shell con `!sh` y ya tendríamos acceso a una shell con privilegios de root.

¿Como podríamos prevenirla?

Para prevenirnos de esta vulnerabilidad debemos establecer una variable de entorno llamada LESSSECURE en 1, que deshabilita el uso de varios comandos entre ellos el de shell (!sh)

¿Que impacto podría tener?

Tan solo los usuarios que puedan ejecutar systemctl podrían llegar a explotar esta vulnerabilidad, pero si un atacante pudiera llegar a conseguirlo podría conseguir el control total sobre el sistema.

CVE-2018-4087 - iOS Core Bluetooth

- Vulnerabilidad consistente en una escalada de privilegios a través de una App maliciosa en iOS
- En iOS y Android las Apps se ejecutan en un *sandbox* o espacio virtual, donde la App tiene acceso sólo a los recursos que necesite
- La vulnerabilidad tiene su origen en un compenente de iOS conocido como *Core Bluetooth*, en el cual existe el proceso *bluetoothd*, el cual permite a la App comunicarse con otros procesos
- Cuando una App trata de comunicarse con el susodicho proceso, se crea *session token*, que será usado por esta App para identificarse y mantener la conexión

- Entonces la vulnerabilidad reside en descubrir el puerto que se asigna para la conexión, cuando el *session token* se une al proceso *bluetoothd*
- Este puerto puede obtenerse por fuerza bruta, debido a que es del tipo *mach_port_t*
- Una vez obtenido el puerto, el atacante puede secuestrar la sesión entre la aplicación y el proceso *bluetoothd*

¿Cómo podríamos explotarla?

- El atacante crearía una App maliciosa que solicitara un puerto para comunicarse con el exterior de su *sandbox*
- Al solicitarlo, se podría obtener el *session token* por fuerza bruta, ya que sabemos que este se genera a través del nombre del puerto
- Una vez obtenido el susodicho token, se podrá alterar y/o manipular la comunicación entre el servicio *bluetoothd* y sus clientes

¿Cómo se podría evitar?

- La manera más efectiva de mantenerse a salvo de esta vulnerabilidad es actualizar el sistema a las versiones más recientes, en las que Apple corrigió este problema, haciendo que el token sea generado de manera aleatoria

¿Que impacto podría tener?

- Esta vulnerabilidad puede provocar la ejecución de código en dispositivos, lo que llevaría al borrado de datos, al bloqueo del dispositivo, a la filtración y robo de datos personales. Podría usarse también para realizar un espionaje tomando datos de las conexiones secuestradas.
- Podría llegar a afectar de manera crítica a los niveles de confidencialidad, integridad y disponibilidad de los datos.

CVE-2022-0847 - Dirty pipe

- Vulnerabilidad en el kernel de Linux de la versión 5.8 en adelante que permite sobreescribir datos en ficheros de sólo lectura
- Esto conlleva a una escalada de privilegios debido a que los procesos sin privilegios son capaces de inyectar código en los procesos raíz
- La vulnerabilidad se encuentra más concretamente en la función *copy_page_to_iter_pipe()* del kernel, la cual tiene la utilidad de copiar datos de una página de memoria al búfer de pipe ("|"). Este búfer se usa para almacenar los datos que se están transmitiendo a través del pipe

- La función anteriormente descrita no inicializa correctamente la variable donde se almacena la flag del pipe, que es donde se almacenan los permisos y estado del mismo
- Al no poder leer los permisos y el estado, un atacante podría utilizar la vulnerabilidad para escribir datos en el búfer del pipe
- Una vez que el atacante ha escrito datos en el búfer del pipe, podría enviar los datos a un archivo de solo lectura, como "/etc/passwd"

¿Cómo podriamos explotarla?

- Podemos detectar esta vulnerabilidad con software como Nmap, Metasploit, Burp Suite u OpenVAS
- También podemos usar un script alojado en GitHub conocido como "CVE-2022-0847-dirty-pipe-checker" (https://github.com/basharkey/CVE-2022-0847-dirty-pipe-checker#cve-2022-0847-dirty-pipe-checker) que se utiliza de la siguiente manera:
  - Ejecutando: ./dpipe.sh (El script simple comprueba si la versión del kernel es vulnerable)
  - Una vez ejecutado, nos mostrará la versión del kernel y si es vulnerable

- Para llevar a cabo la explotación podemos ejecutar alguno de los scripts alojados en la dirección siguiente: https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits
  - El primer script sobrescribe datos sobre el fichero /etc/passwd
  - El segundo eleva los privilegios mediante un binario que tenga permisos SUID. (Este permiso se utiliza para permitir a los usuarios del sistema ejecutar binarios con privilegios elevados temporalmente para realizar una tarea específica.

¿Qué impacto tendría?

Esta vulnerabilidad tiene graves impactos sobre la disponibilidad, integridad y confidencialidad de los datos:

- Disponibilidad: Se podría utilizar para interrumpir servicios críticos, instalar malware o tomar el control del sistema.
- Integridad: Podría ser usada para modificar archivos de solo lectura.
- Confidencialidad: Tiene la utilidad de recopilar información confidencial como contraseñas, archivos personales, etc...

¿Cómo se podría evitar?

Tenemos varias medidas a nuestra disposición para mitigar o directamente, evitar el uso de esta vulnerabilidad sobre nuestro sistema:

- Deshabilitar el acceso remoto al kernel. Esto se puede hacer editando el archivo /etc/sysctl.conf y comentando la siguiente línea: -kernel.unprivileged_userns_clone=1
- Utilizar un firewall para bloquear el acceso no autorizado a ciertos servicios como por ejemplo ssh (puerto 22)
- Implementar un sistema de detección de intrusiones (IDS) para que nos ayude a detectar el ataque
- La vulnerabilidad se solucionó en Linux 5.16.11, 5.15.25 y 5.10.102 por lo que usar una de estas versiones nos solucionaría el problema

CVE-2020-7384 - MSFVENOM

- El framework msfvenom de Metaexploit permite a un usuario malintencionado crear y publicar APK's que permiten la ejecución de comandos controlados por el atacante en el dispositivo android donde se ha instalado.
- Esta vulnerabilidad afecta a cualquier versión android.

¿Cómo podriamos explotarla?

Para explotar esta vulnerabilidad lo unico que necesitariamos es una máquina con kali linux para crear el apk que contendrá una reverse shell hacia nuestra máquina, de esta manera cuando se instale en un dispositivo, tendremos control total sobre ese dispositivo.

¿Qué impacto tendría?

- Disponibilidad: El atacante podría controlar el dispositivo por lo que podría borrar datos, bloquear el acceso, etc...
- Integridad: El atacante podría modificar o eliminar datos lo que podria dañar el dispositivo o la información que contiene.
- Confidencialidad: El atacante podría robar datos como contraseñas, fotos, datos bancarios, etc... lo que comprometeria la privacidad y los datos podrian ser utilizados para cometer fraude o robar su identidad.

¿Cómo se podría evitar?

- No descargar nunca apk's desconocidas que suelen ofrecernos servicios de pago de forma gratuita ya que estas suelen tener malware.
- Actualizar siempre el móvil a la última versión disponible.
- Tener instalado siempre un antivirus/antimalware en el dispositivo.

CVE-2023-39191 - Kernel de Linux eBPF

- Esta vulnerabilidad se produce debido a una falta de validación de entrada en el subsistema *eBPF*.
- *eBPF* se puede utilizar para una variedad de propósitos, como la supervisión del rendimiento, el filtrado de tráfico y la creación de reglas de firewall.
- Significa que un atacante puede proporcionar datos maliciosos a un programa eBPF que el kernel ejecutará sin verificar.

¿Cómo podriamos explotarla?

- Para explotar esta vulnerabilidad, un atacante debe tener privilegios *CAP_BPF*. (Cargar y moficar programas BPF en el kernel)
- Una vez que el atacante tenga dichos privilegios, podrá proporcionar datos maliciosos a un programa eBPF, los cuales serán ejecutados por el kernel.

¿Qué impacto tendría?

- **Confidencialidad:** El atacante podría acceder a datos confidenciales, como contraseñas, información financiera o datos personales.
- **Integridad:** El atacante podría modificar o destruir datos.
- **Disponibilidad:** El atacante podría interrumpir los servicios o sistemas.

¿Cómo se podría evitar?

- **Actualizar los sistemas a la última versión del kernel de Linux.** Esta versión corrige la vulnerabilidad CVE-2023-39191.
- **Limitar el acceso a los privilegios CAP_BPF.** Solo los usuarios autorizados deben tener estos privilegios.
- **Implementar controles de acceso para proteger los programas eBPF.** Estos controles pueden ayudar a evitar que los atacantes proporcionen datos maliciosos a los programas eBPF.
- **Monitorear los sistemas en busca de signos de explotación de esta vulnerabilidad.** Los administradores de sistemas deben estar atentos a los signos de actividad maliciosa, como intentos de ejecutar código arbitrario en el contexto del kernel.

CVE-2023-44466 - Ceph

- Es una vulnerabilidad de desbordamiento de búfer en el módulo de mensajería del kernel de Linux.
- Esto tiene su origen en el sistema de ficheros *Ceph*, ya que el módulo de mensajería es utilizado por Ceph para recibir paquetes TCP de una dirección IP.
- Antes de que se complete cualquier autorización, cualquier dispositivo con esa dirección IP puede enviar un paquete que produzca desbordamiento de búfer en el kernel.

¿Cómo podriamos explotarla?

- Para explotar esta vulnerabilidad, un atacante debe tener acceso al sistema vulnerable.
- Una vez que este haya conseguido acceso al sistema, podrá enviar un mensaje malicioso al subsistema de mensajería del kernel de Linux, el cual lo procesará de forma incorrecta, produciendo así un desbordamiento del búfer. Esto podría permitir al atacante ejecutar código arbitrario en el contexto del kernel.

¿Qué impacto tendría?

- **Confidencialidad:** El atacante podría acceder a datos confidenciales, como contraseñas, información financiera o datos personales.
- **Integridad:** El atacante podría modificar o destruir datos.
- **Disponibilidad:** El atacante podría interrumpir los servicios o sistemas (*DoS*).

¿Cómo se podría evitar?

- **Actualizar los sistemas a la última versión del kernel de Linux.** Esta versión corrige la vulnerabilidad CVE-2023-44466.
- **Limitar el acceso al subsistema de mensajería del kernel de Linux.** Solo los usuarios autorizados deben tener acceso a este subsistema.
- **Monitorear los sistemas en busca de signos de explotación de esta vulnerabilidad.** Los administradores de sistemas deben estar atentos a los signos de actividad maliciosa, como intentos de enviar mensajes maliciosos al subsistema de mensajería del kernel de Linux.

Conclusión:

- Enfatiza la importancia de entender componentes de sistemas a bajo nivel para una ciberseguridad efectiva.
- Destaca la importancia de mantener sistemas actualizados para prevenir vulnerabilidades.
- Fomenta el uso de medidas de seguridad como firewalls, antivirus y sistemas de detección de intrusiones.

Recomendaciones:

- Utilizar antivirus/antimalware y escáneres de vulnerabilidades.
- Implementar firewalls y cambiar puertos de servicios.
- Mantener sistemas actualizados y monitorear signos de actividad maliciosa.
- Fomentar una cultura de conciencia en ciberseguridad para minimizar riesgos.

¡Muchas gracias por su atención!

IES Rafael Alberti - Ciberseguridad en las TI

2023-2024