THE BRIDGE

CYBER SECURITY

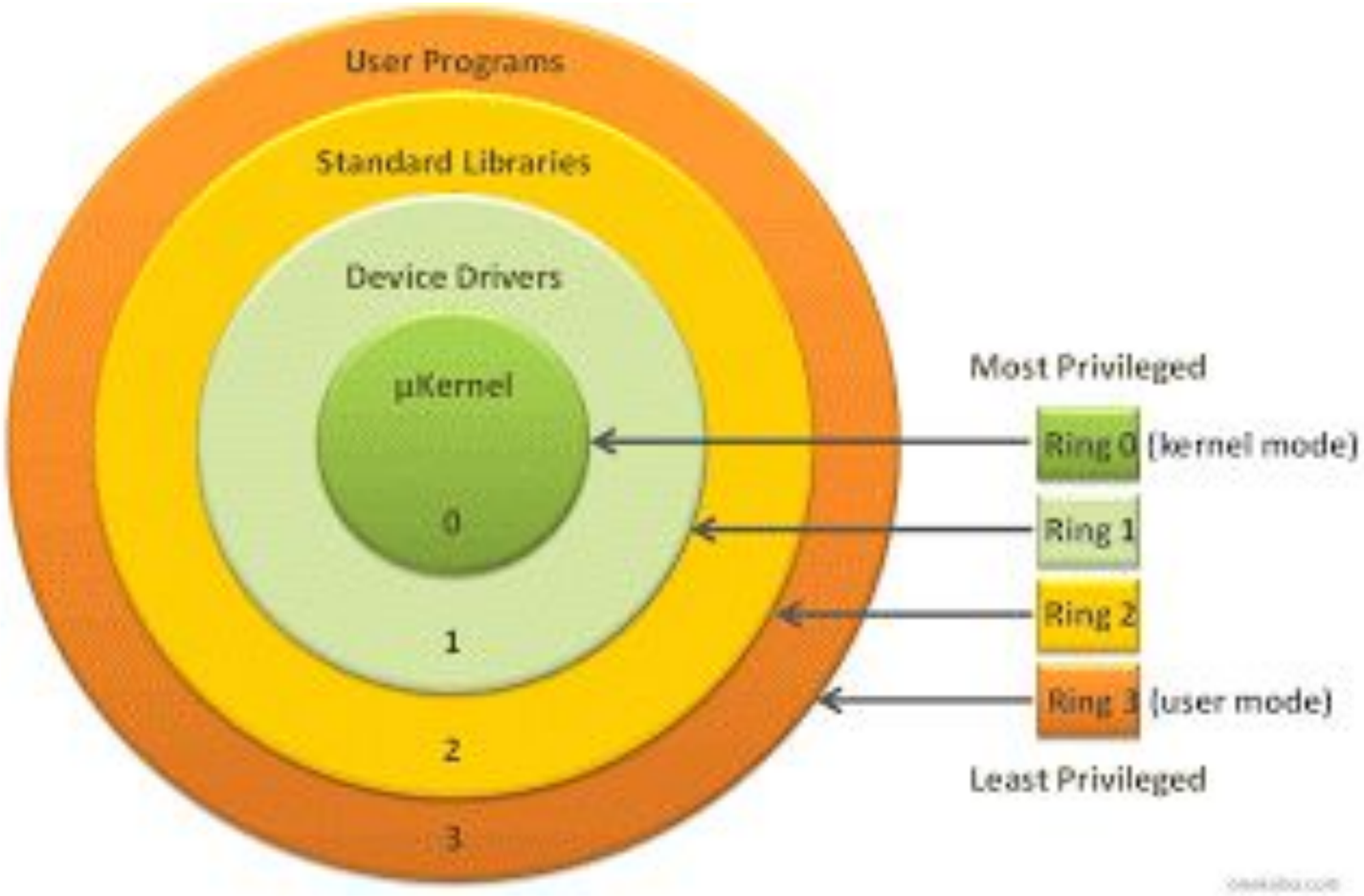
Teoría de Elevación de Privilegio

Elevación de Privilegios

- La **escalada de privilegios** es el acto de explotar un error, un fallo de diseño o una supervisión de la configuración en un sistema operativo o una aplicación de software para obtener un acceso elevado a los recursos que normalmente están protegidos de una aplicación o un usuario.

- El resultado es que una aplicación con más privilegios de los previstos por el desarrollador de la aplicación o el administrador del sistema puede realizar acciones no autorizadas.

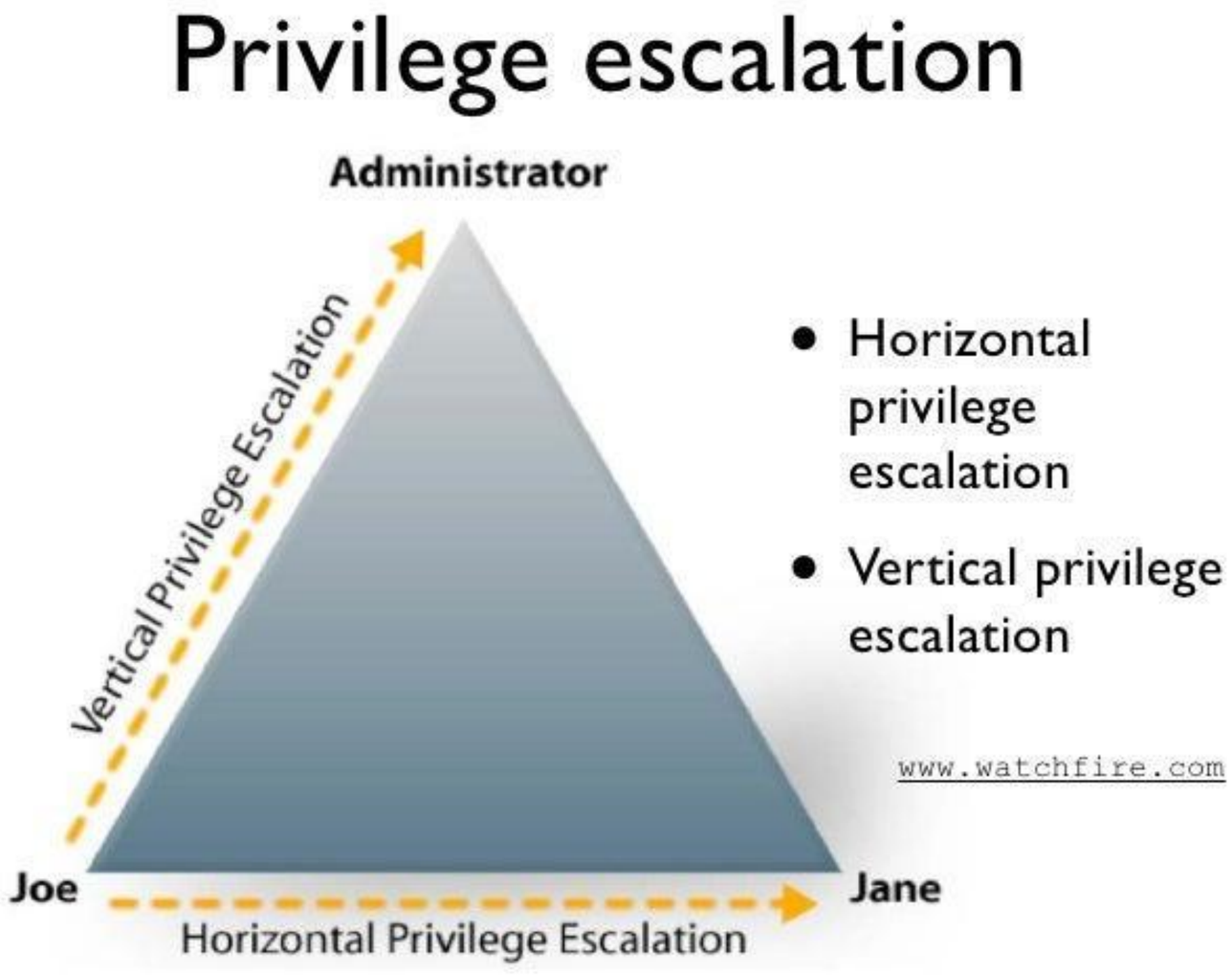
- Consiste en aprovechar **vulnerabilidades** del sistema como **archivos** o **servicios mal configurados** para poder ejecutar **scripts** o **exploits** con privilegios de **superusuario**



https://steemit.com/spanish/@sethroot/post-explotacion-pentesting-escalado-de-privilegios

Elevación de Privilegios

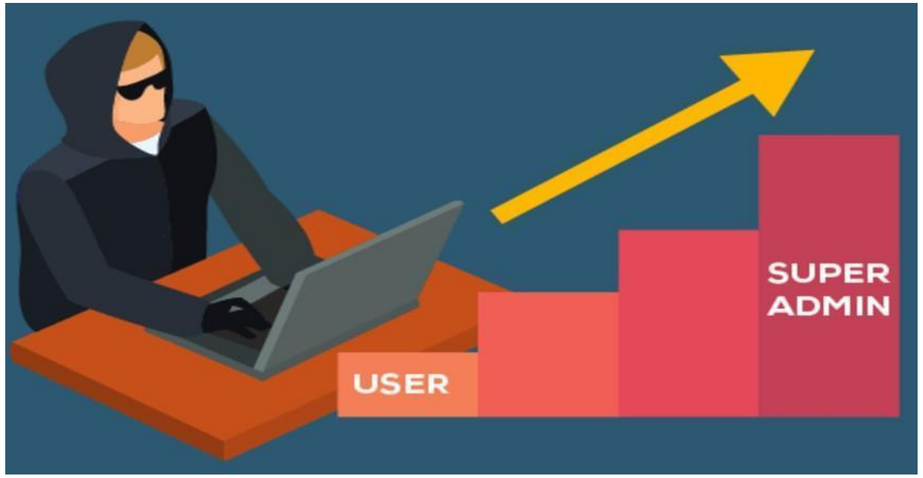
- La escalada de privilegios se produce de dos formas:
  - **Escalamiento Horizontal**
    - El usuario malicioso mantiene sus privilegios de bajo nivel actuales.
    - Consigue tener acceso a datos y funcionalidades que no deberían estar disponibles para este.
    - Los mismos pueden pertenecer a otros usuarios con mayores privilegios o procesos de sistema.
  - **Escalamiento Vertical**
    - el usuario malicioso con privilegios bajos pasa a tener mayor cantidad de privilegios.
    - La **Escalamiento de Privilegios** puede ocurrir en cualquiera de los otros sistemas operativos que conocemos.
    - La peor parte es que el **cibercriminal** puede borrar cualquier rastro del ataque que haya llevado a cabo mediante esta escalada de privilegios, de manera a que no sea posible localizar algún tipo de evidencia que permita el análisis de estos eventos.



https://blog.segu-info.com.ar/2020/03/escalamiento-de-privilegios-como.html?m=0

Elevación de Privilegios en Linux

- **Fases en Linux**
  - **Recopilar**: enumeración , más enumeración y algo más de enumeración.
  - **Proceso**: ordenar datos, analizar y priorizar.
  - **Buscar**: Buscar y encontrar el código de explotación.
  - **Adaptar**: personaliza el exploit para que encaje. No todos los exploits funcionan para todos los sistemas "listos para usar".
  - **Prueba**: Preparación para pruebas y errores .
- **Medidas de Protección**
  - Reforzar las políticas de contraseñas
  - Creación de usuarios especiales y grupos de usuarios con mínimos privilegios.
  - Evitar los típicos errores de programación de aplicaciones
  - Mantener la seguridad de las bases de datos y filtros de ingreso de datos de usuarios.
  - Mantenimiento de sistemas y aplicaciones al día
  - Permisos correctos y adecuados para todos los archivos y directorios
  - Cerrar puertos innecesarios y borrado de usuarios inactivos.
  - Limitar las posibilidades de transferir/descargar archivos
  - Modificar las credenciales por defecto en todos los dispositivos



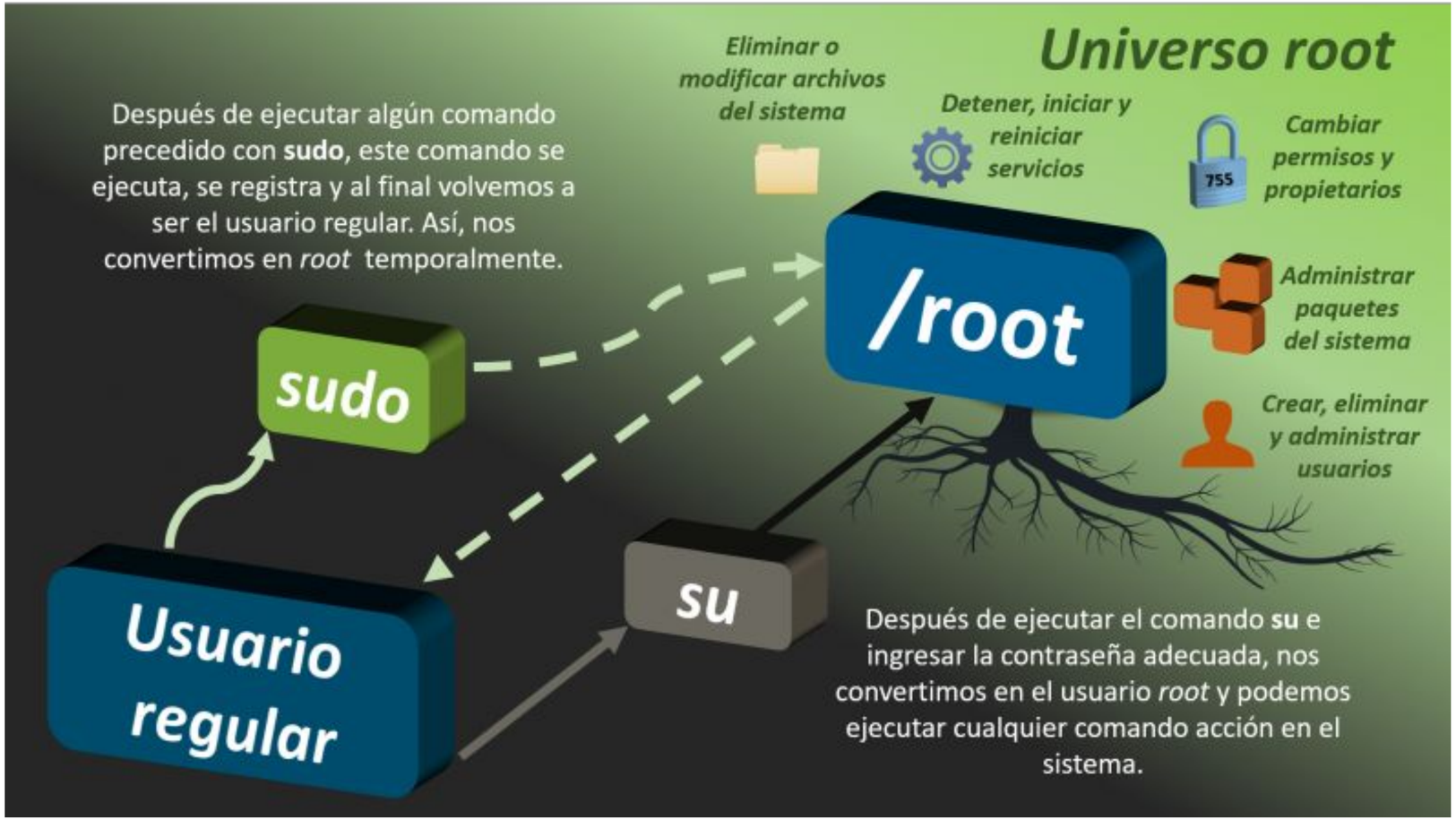
https://www.redeszone.net/tutoriales/seguridad/escalada-privilegios-que-es-funcionamiento/

Comandos de Elevación de Privilegios Linux

- **Se tienen 2 comandos que son:**

- **su**
  - Comando **su** (**switch user** o "cambiar de usuario).
  - Este comando simplemente ejecuta una nueva sesión en la consola shell como otro usuario, en general, el usuario **root**.
  - De esta manera, simplemente nos convertimos en el usuario **root** hasta cerrar esa sesión.

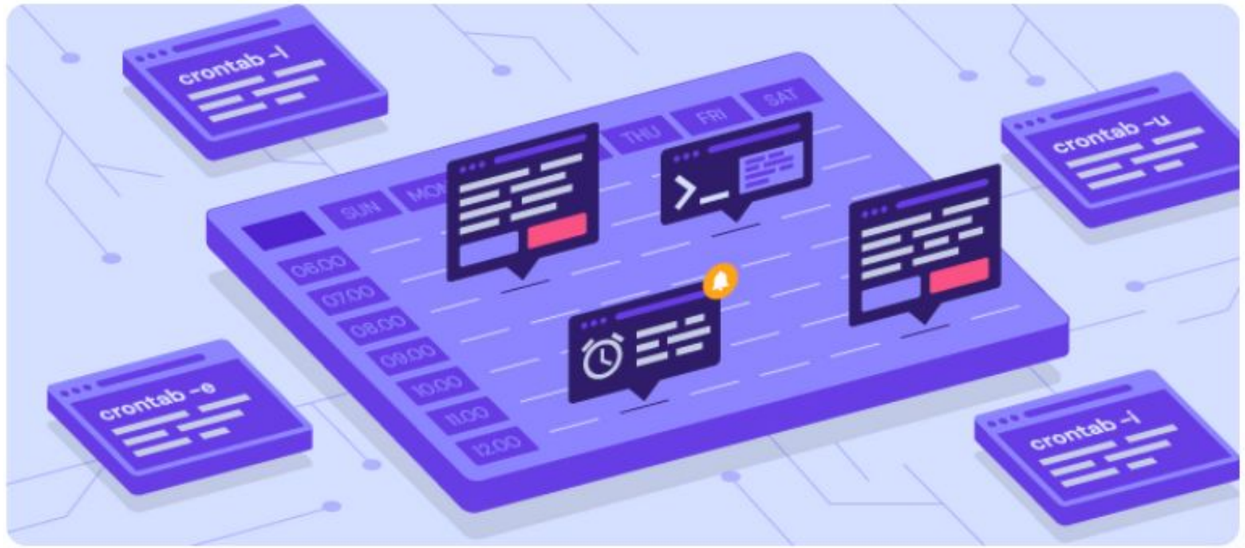
- **sudo**
  - El comando **sudo** (**super user do**, por sus siglas en inglés).
  - Esta es la forma recomendada y la mejora práctica para ejecutar acciones con privilegios elevados.
  - Se guarda en el archivo **/etc/sudoers**.
  - La gran diferencia, es que los comandos precedidos con **sudo** son ejecutados por el propio usuario, no por **root**



https://help.livehost.host/hc/linux/de-la-teoria-a-la-practica-sudo-sudoers-y-visudo?lang=es

Cron

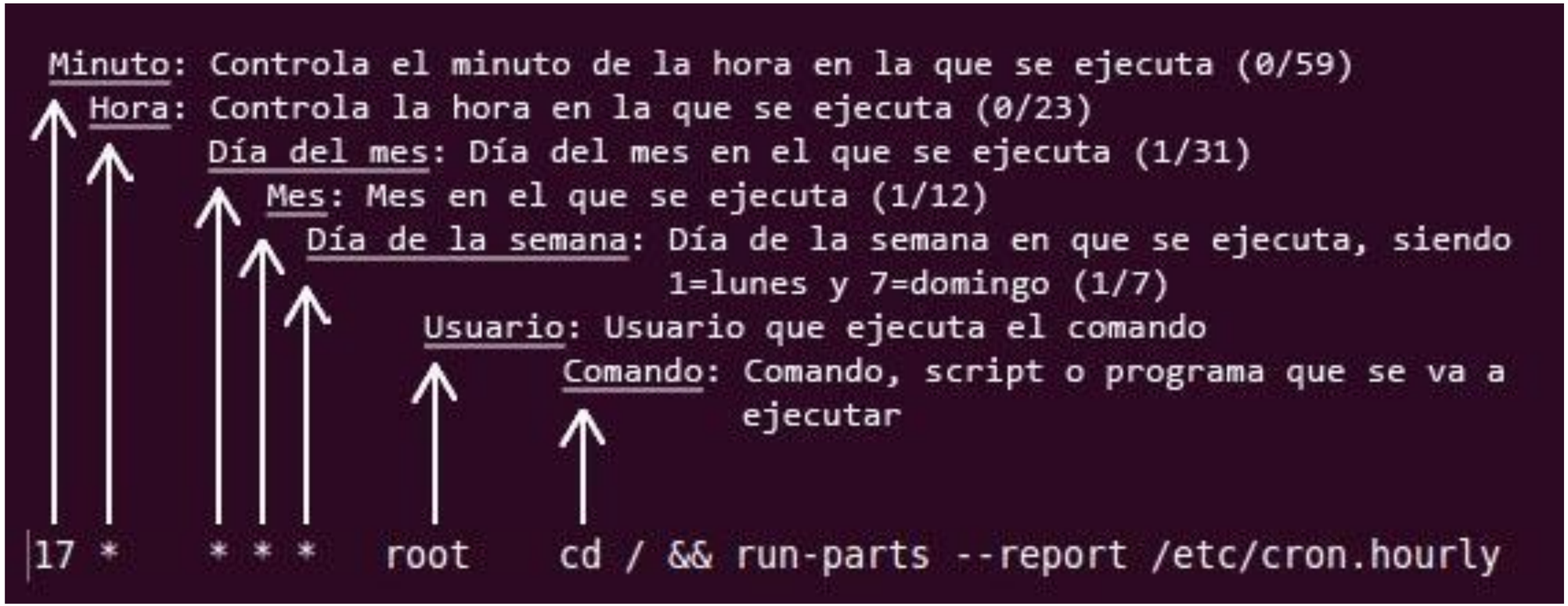
- El nombre **cron** viene del griego chronos que significa “tiempo”.
- En el sistema operativo **Unix**, **cron** es un administrador regular de procesos en segundo plano (**demonio**) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes).
- Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero **crontab**.
- El demonio cron inicia de **/etc/rc.d/** o **/etc/init.d** dependiendo de la distribución.
- El c**ron** se ejecuta en el **background**, revisa cada minuto la tabla de tareas **crontab** de **/etc/crontab** o en **/var/spool/cron** en búsqueda de tareas que se deban cumplir.
- Como usuario podemos agregar comandos o scripts con tareas a **cron** para automatizar algunos procesos.
- Esto es útil por ejemplo para automatizar la actualización de un sistema o un buen sistema de respaldos.



https://www.hostinger.es/tutoriales/cron-job

Archivo Crontab

- Es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario.
- Verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en el background.
- Cada usuario puede tener su propio archivo **crontab**, de hecho, el **/etc/crontab** se asume que es el archivo **crontab** del usuario **root**, cuando los usuarios normales (e incluso **root**) desean generar su propio archivo de **crontab**, entonces utilizaremos el **comando crontab**.
- Es la manera más sencilla de administrar tareas de **cron** en sistemas multiusuario, ya sea como simple usuario de sistema o usuario **root**.



http://dis.um.es/~lopezquesada/documentos/IES_1213/SAD/curso/UT3/ActividadesAlumnos/13/cron.html

Variable de entorno PATH.

- Una variable de entorno en Linux es un valor nominal dinámico que se almacena en el sistema y que se utiliza por las aplicaciones lanzadas en **shells**.
- Las variables de entorno son marcadores de posición para la información del sistema que pasa datos a los programas.
- Se pueden enumerar, configurar y hacer persistentes con diferentes comandos y archivos
- **PATH**
  - Cuando invocamos un comando en Linux como **tree** o **un shell**, estamos ejecutando un programa almacenado en alguna parte.
  - Este programa es localizado por **Linux** en diferentes posibles ubicaciones de forma automática.
  - Para saber la ubicación de un comando utilizamos otro comando llamado **which**
  - La forma en la que **Linux** encuentra estos comandos es a través de la variable de entorno **PATH**.
  - Esta variable almacena una lista de directorios en los cuales **Linux** debe buscar un comando para poder ejecutarlo.
  - Uno de los ataques más comunes en sistemas **Linux** son usando esta variable.

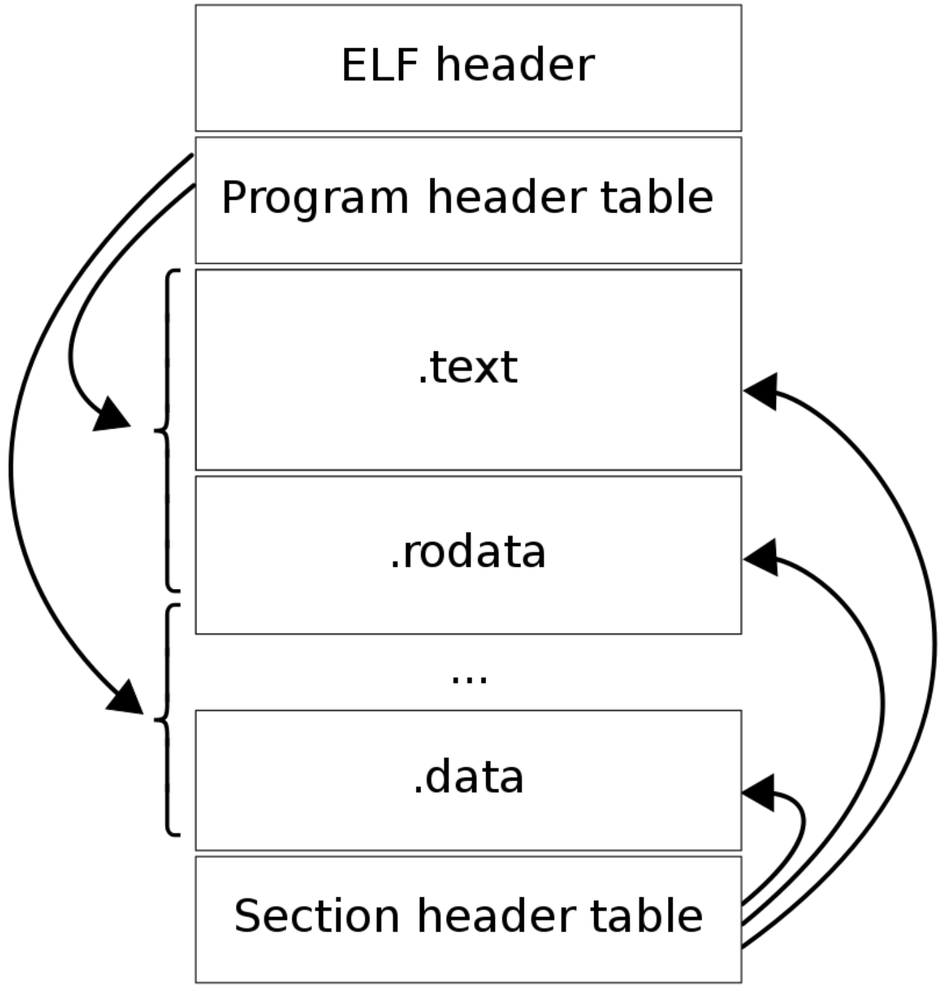
Compilación de Binarios

- **gcc**
  - El **GNU Compiler Collection** (colección de compiladores GNU) es un conjunto de compiladores creados por el proyecto GNU.
  - GCC es software libre y lo distribuye la Free Software Foundation (FSF) bajo la licencia general pública GPL.

- **ELF**
  - El formato de archivo **ELF** (**Extensible Linking Format**) es un archivo ejecutable con sistemas **Unix** y **Linux**.
  - Hay **tres tipos** de archivos ELF diferentes
    - **Archivo de objeto reubicable (reubicable)**: normalmente con extensión **.O**. Al compilar un programa en C, los compiladores primero transforman todos los archivos de código fuente del programa en archivos de objetos compilados. Luego, el compilador vincula los archivos de objetos compilados en un archivo ejecutable.

    - **Ejecutable (ejecutable)**: Archivos ejecutables para aplicaciones y herramientas, normalmente sin extensión o con extensión **BIN**.

    - **Biblioteca compartida (objeto compartido)**: archivos con extensión **.SO**, contienen información que puede ser utilizada por uno o más programas para descargar recursos de modo que las aplicaciones que llaman al archivo SO no tengan que proporcionar el archivo.

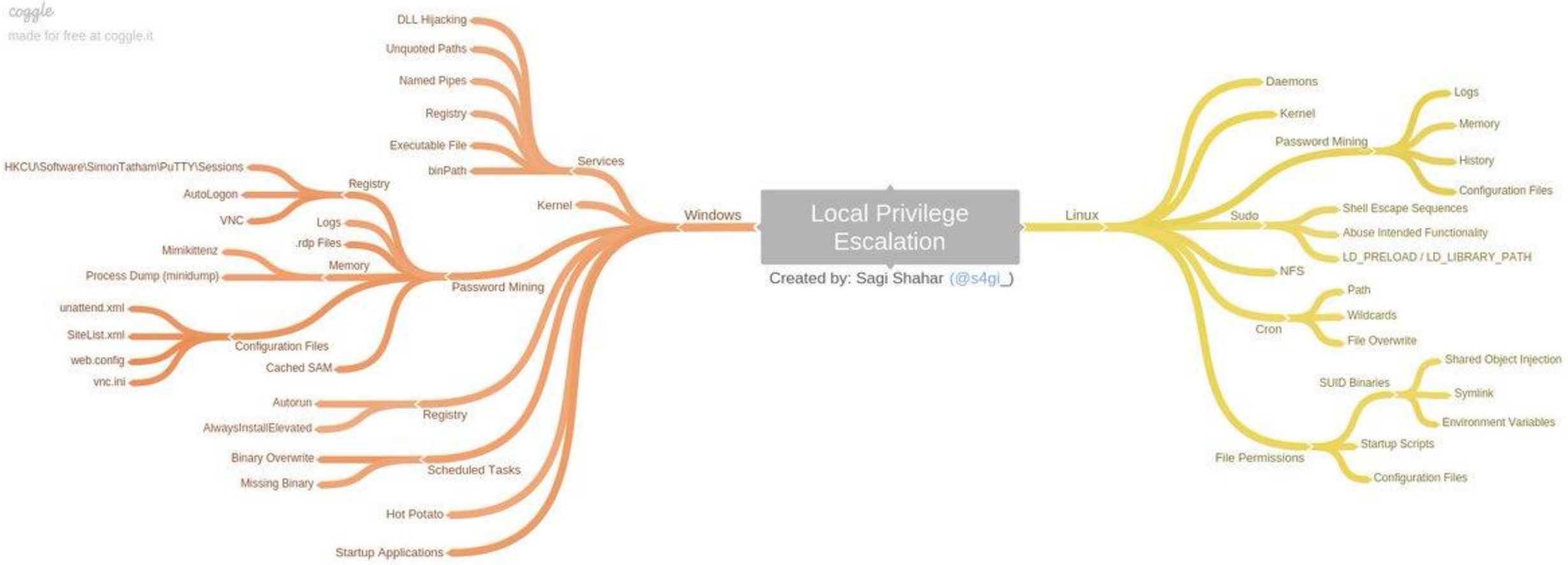


https://es.wikipedia.org/wiki/Executable_and_Linkable_Format

Permiso SUID

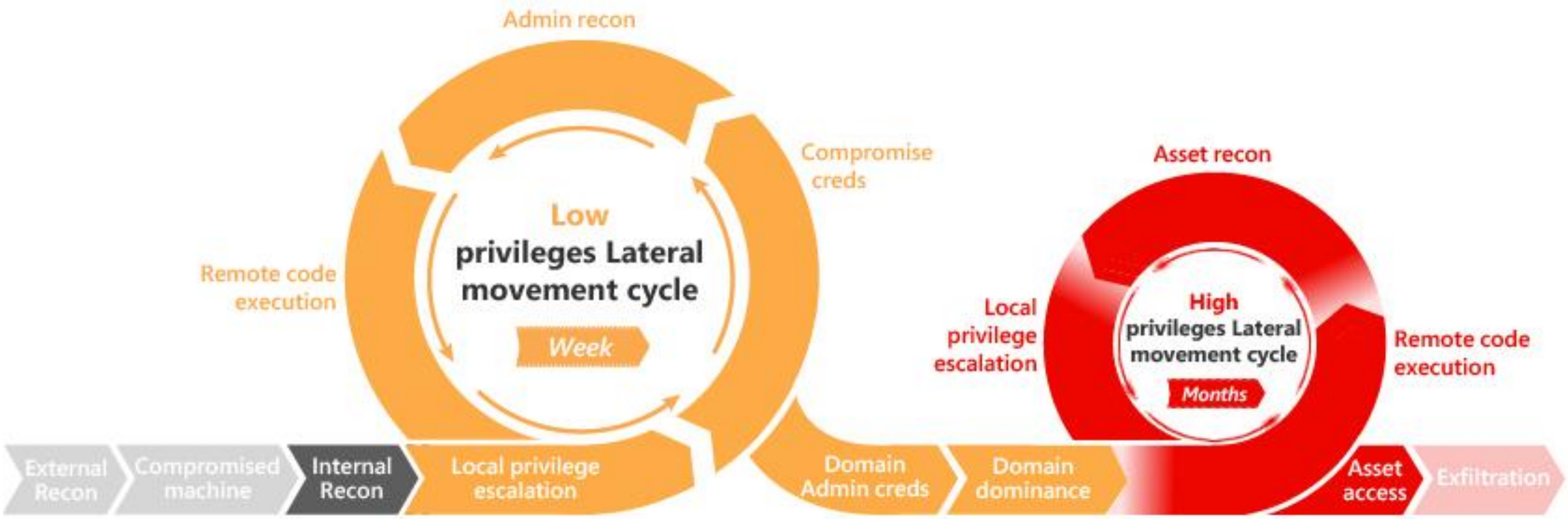
- **Setuid** y **Setgid** son términos de **Unix**, abreviaturas para "**Set User ID**" y "**Set Group ID**", respectivamente.
- **Setuid**, también llamado a veces "**suid**", y "**setgid**" son permisos de acceso que pueden asignarse a archivos o directorios en un sistema operativo basado en Unix.
- Se utilizan principalmente para permitir a los usuarios del sistema ejecutar binarios con privilegios elevados temporalmente para realizar una tarea específica.
- Si un fichero tiene activado el bit "**Setuid**" se identifica con una "**s**" en un listado de la siguiente forma:
  - -rw**s**r-xr–x 1 root shadow 27920 ago 15 22:45 /usr/bin/passwd
- Esta propiedad es necesaria para que los usuarios normales puedan realizar tareas que requieran privilegios más altos que los que dispone un usuario común.
- Algunas tareas que requieren privilegios elevados pueden no ser obvias, por ejemplo,
  - El comando **ping**, que debe enviar y escuchar paquetes de control por una interfaz de red.
  - El ejemplo más claro de fichero ejecutable y con el bit **setuid** es el **su**.
- El bit **SUID** no debe configurarse especialmente en ningún editor de archivos, ya que un atacante puede sobrescribir cualquier archivo presente en el sistema.

Escalada de Privilegios



https://www.hackplayers.com/2019/11/taller-para-escalar-privilegios-en-windows-linux.html

Fases de la elevación de Privilegios en Windows



https://www.linkedin.com/pulse/tty-interactiva-como-hacer-una-terminal-jos%C3%A9-francisco-flores-zxnee/

Linux Privilege Escalation

- **Técnicas utilizadas para la escalada de privilegios**
  - Como ya hemos la explotación y que ahora tenemos **shell** en el sistema remoto.
  - Dependiendo de cómo llegamos allí, probablemente no tengamos privilegios de **'root'**.
  - Las técnicas mencionadas a continuación se pueden utilizar para obtener acceso a **'root'** en el sistema.
    - **1. Explosiones del núcleo**
      - Los exploits del **kernel** son programas que aprovechan las vulnerabilidades del **kernel** para ejecutar código arbitrario con permisos elevados
      - El infame exploit DirtyCow – Linux Kernel <= 3.19.0-73.8
    - **2. Explotar servicios que se ejecutan como root**
      - Siempre debe verificar si los servidores web, los servidores de correo, los servidores de bases de datos, etc.
      - Se están ejecutando como **root**.
      - Muchas veces, los administradores web ejecutan estos servicios como **root** y se olvidan de los problemas de seguridad que podría causar
      - Ejemplo del servicio EXIM de correo electrónico.
    - **3. Explotación de los ejecutables SUID**
      - Al marcar el programa ping como SUID con el propietario como **root**, el **ping** se ejecuta con privilegios de **root** cada vez que un usuario con privilegios bajos ejecuta el programa

Linux Privilege Escalation

- **4. Explotación de los derechos/usuario con SUDO**
  - Si el atacante no puede obtener acceso raíz directamente a través de otras técnicas, podría intentar comprometer a cualquiera de los usuarios que tienen acceso **SUDO**

- **5. Explotar trabajos cron mal configurados**
  - Los trabajos cron generalmente se ejecutan con privilegios de **root**. Si podemos manipular con éxito cualquier script o binario que esté definido en los trabajos cron, entonces podemos ejecutar código arbitrario con privilegios de **root**.

- **6. Explotación de usuarios con '.' en su RUTA**
  - Tener '.' en el PATH significa que el usuario puede ejecutar binarios/scripts desde el directorio actual.
  - Para evitar tener que ingresar esos dos caracteres adicionales cada vez, el usuario agrega '.' a su PATH.
  - Este puede ser un método excelente para que un atacante aumente sus privilegios.


THE
BRIDGE