ASICI

Asociación Internacional de Ciberseguridad

Guía de Comandos Linux para manejo de Logs

Los registros o Logs

El manejo de logs (registros) es una habilidad fundamental para cualquier administrador de sistemas, ingeniero de DevOps o profesional de ciberseguridad en entornos Linux. Los logs proporcionan información crucial sobre el estado del sistema, eventos de seguridad, errores de aplicaciones y actividad del usuario, siendo esenciales para la resolución de problemas, auditorías de seguridad y monitoreo del rendimiento.

1. Ubicación Común de Logs

La mayoría de los logs del sistema y de las aplicaciones se almacenan en el directorio `/var/log`. Dentro de este directorio, encontrarás subdirectorios y archivos de log específicos:

- `/var/log/syslog` o `/var/log/messages`: Registros generales del sistema (depende de la distribución).
- `/var/log/auth.log` o `/var/log/secure`: Registros de autenticación y seguridad.
- `/var/log/kern.log`: Registros del kernel.
- `/var/log/boot.log`: Mensajes de arranque del sistema.
- `/var/log/dmesg`: Buffer de mensajes del kernel.
- `/var/log/apt/history.log`: Historial de paquetes de APT (Debian/Ubuntu).
- `/var/log/yum.log`: Historial de paquetes de YUM/DNF (Red Hat/CentOS).
- `/var/log/apache2/access.log` y `/var/log/apache2/error.log`: Logs de Apache.
- `/var/log/nginx/access.log` y `/var/log/nginx/error.log`: Logs de Nginx.
- `/var/log/mysql/error.log`: Logs de MySQL/MariaDB.
- `/var/log/mail.log`: Logs del servidor de correo.
- `/var/log/lastlog`: Registra la última conexión de los usuarios (no es un archivo de texto plano).
- `/var/log/wtmp` y `/var/log/btmp`: Registros de inicios de sesión y fallidos (no son archivos de texto plano, se leen con `last` y `lastb`).

2. Visualización de Logs

Estos comandos te permiten ver el contenido de los archivos de log.

- **`cat <archivo_log>`:** Muestra el contenido completo de un archivo de log. Útil para archivos pequeños, pero no para archivos muy grandes.
    - Ejemplo: `cat /var/log/syslog`
- **`less <archivo_log>`:** Permite ver el contenido del archivo página por página. Puedes buscar (`/`), avanzar (`espacio`), retroceder (`b`) y salir (`q`). Es la herramienta más común para revisar logs.
    - Ejemplo: `less /var/log/auth.log`
- **`more <archivo_log>`:** Similar a `less`, pero con menos funcionalidades.

    - Ejemplo: `more /var/log/kern.log`
- **`head <archivo_log>`:** Muestra las primeras 10 líneas de un archivo (útil para ver el inicio).
    - Ejemplo: `head /var/log/boot.log`
    - `head -n 20 <archivo_log>`: Muestra las primeras 20 líneas.
- **`tail <archivo_log>`:** Muestra las últimas 10 líneas de un archivo (útil para ver los eventos más recientes).
    - Ejemplo: `tail /var/log/messages`
    - `tail -n 50 <archivo_log>`: Muestra las últimas 50 líneas.
    - **`tail -f <archivo_log>` (seguir en tiempo real):** Una de las opciones más usadas. Mantiene el archivo abierto y muestra las nuevas líneas a medida que se añaden. Esencial para monitorear logs en tiempo real.
        - Ejemplo: `tail -f /var/log/auth.log` (para ver intentos de login en vivo).
    - **`tail -F <archivo_log>`:** Similar a `-f`, pero también monitorea si el archivo es rotado (renombrado y recreado), y seguirá el nuevo archivo con el mismo nombre.
        - Ejemplo: `tail -F /var/log/apache2/access.log`
- **`grep <patrón> <archivo_log>`:** Busca líneas que coincidan con un patrón específico dentro de un archivo de log. Muy potente para filtrar información.
    - Ejemplo: `grep "ERROR" /var/log/syslog` (busca líneas que contengan "ERROR").
    - Ejemplo: `grep -i "failed password" /var/log/auth.log` (busca "failed password" ignorando mayúsculas/minúsculas).
    - Ejemplo: `grep -v "info" /var/log/syslog` (muestra líneas que *no* contienen "info").
    - Combinando con `tail`: `tail -f /var/log/apache2/error.log | grep "PHP Warning"` (monitorea errores PHP en tiempo real).
- **`zcat`, `zless`, `zmore`, `zgrep`:** Equivalentes a `cat`, `less`, `zmore`, `grep` pero para archivos comprimidos con `gzip` (que a menudo se usan para logs rotados, como `/var/log/syslog.1.gz`).
    - Ejemplo: `zless /var/log/syslog.1.gz`
    - Ejemplo: `zgrep "connection refused" /var/log/nginx/error.log.1.gz`

3. Filtrado y Procesamiento de Logs

Estos comandos te ayudan a refinar la información obtenida de los logs.

- **`awk`:** Un potente lenguaje de procesamiento de texto. Útil para extraer campos específicos o realizar operaciones más complejas.
    - Ejemplo: `cat /var/log/apache2/access.log | awk '{print $1, $7}'` (muestra la IP de origen y la URL solicitada).

- **`sed`:** Otro potente editor de flujos de texto, útil para reemplazar o manipular texto en los logs.
    - Ejemplo: `cat /var/log/syslog | sed 's/kernel/núcleo/g'` (reemplaza "kernel" por "núcleo").
- **`cut`:** Extrae secciones o campos de cada línea de un archivo.
    - Ejemplo: `cat /var/log/auth.log | grep "sshd" | cut -d' ' -f1,2,3,9` (extrae fecha, hora y el usuario en los logs de sshd).
- **`sort`:** Ordena las líneas de un archivo.
    - Ejemplo: `cat /var/log/apache2/access.log | awk '{print $1}' | sort | uniq -c | sort -nr` (cuenta las IPs únicas que accedieron al servidor y las ordena de mayor a menor).
- **`uniq`:** Reporta o filtra líneas repetidas adyacentes.
    - `uniq -c`: Cuenta las ocurrencias.
- **`wc -l`:** Cuenta el número de líneas en un archivo (útil para saber cuántos errores, por ejemplo).
    - Ejemplo: `grep "Failed password" /var/log/auth.log | wc -l` (cuenta los intentos fallidos de contraseña).

4. Herramientas Específicas para Logs

Algunas herramientas están diseñadas específicamente para interactuar con tipos de logs o sistemas de registro.

- **`journalctl`:** Comando para consultar el `systemd journal`, que es el sistema de registro moderno en muchas distribuciones Linux (como Ubuntu 16.04+, CentOS 7+, Fedora). Centraliza los logs de kernel, servicios, aplicaciones, etc.
    - `journalctl`: Muestra todos los logs desde el arranque.
    - `journalctl -f`: Muestra logs en tiempo real (similar a `tail -f`).
    - `journalctl -u <servicio>`: Muestra logs de un servicio específico (ej. `journalctl -u nginx`).
    - `journalctl -p err`: Muestra solo mensajes de error.
    - `journalctl --since "2023-01-01 10:00:00" --until "2023-01-01 11:00:00"`: Filtra por rango de tiempo.
    - `journalctl _PID=<PID>`: Muestra logs de un proceso específico.
    - `journalctl --disk-usage`: Muestra el espacio en disco usado por el journal.
- **`last`:** Muestra la lista de los últimos usuarios que iniciaron sesión. Lee el archivo `/var/log/wtmp`.
    - Ejemplo: `last`
    - `last -a`: Muestra el hostname en la última columna.
- **`lastb`:** Muestra los intentos de inicio de sesión fallidos. Lee el archivo `/var/log/btmp`. Requiere privilegios de superusuario.
    - Ejemplo: `sudo lastb`

- **`dmesg`:** Muestra el buffer de mensajes del kernel. Útil para depurar problemas de hardware o de controladores.
    - Ejemplo: `dmesg | less`
    - `dmesg | grep "eth0"` (muestra mensajes relacionados con la interfaz de red `eth0`).
- **`logrotate`:** Es una utilidad que se encarga de la rotación, compresión y eliminación de los archivos de log. Evita que los archivos de log crezcan indefinidamente y consuman todo el espacio en disco. Su configuración se encuentra en `/etc/logrotate.conf` y en `/etc/logrotate.d/`. No es un comando de visualización, sino de gestión.

5. Manejo de Logs de Servidores Web (Apache/Nginx)

Los servidores web generan logs muy detallados sobre las solicitudes HTTP. Son cruciales para monitorear el tráfico, detectar ataques, analizar el rendimiento y depurar errores.

- **Ubicaciones comunes:**
    - **Apache:** `/var/log/apache2/access.log` (peticiones exitosas) y `/var/log/apache2/error.log` (errores del servidor). En CentOS/RHEL, suelen estar en `/var/log/httpd/`.
    - **Nginx:** `/var/log/nginx/access.log` y `/var/log/nginx/error.log`.
- **Análisis de `access.log` (ejemplos):**
    - Ver tráfico en tiempo real: `tail -f /var/log/apache2/access.log`
    - Contar el número de solicitudes por IP: `awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr`
    - Encontrar todas las peticiones a una URL específica: `grep "/admin/login.php" /var/log/nginx/access.log`
    - Contar los códigos de estado HTTP más comunes: `awk '{print $9}' /var/log/apache2/access.log | sort | uniq -c | sort -nr`
    - Mostrar solo las solicitudes que resultaron en un error 404: `grep " 404 " /var/log/nginx/access.log`
    - Obtener los 10 agentes de usuario más comunes: `awk -F'"' '{print $6}' /var/log/apache2/access.log | sort | uniq -c | sort -nr | head -n 10`
- **Análisis de `error.log` (ejemplos):**
    - Monitorear errores PHP en tiempo real: `tail -f /var/log/apache2/error.log | grep "PHP Warning\|PHP Parse error"`
    - Buscar errores específicos de permisos: `grep "Permission denied" /var/log/nginx/error.log`

6. Códigos de Estado HTTP Comunes en Logs Web

Los logs de acceso web suelen incluir un código de estado HTTP para cada solicitud, indicando el resultado de la petición.

Código	Clase	Descripción	Significado en Logs
1xx	**Informational**	Respuesta provisional.	Indica que la solicitud ha sido recibida y el proceso continúa. Raros en logs de acceso típicos.
100	Informational	Continue	El cliente debe continuar con su solicitud.
2xx	**Success**	La acción fue recibida, entendida y aceptada.	**Indica que la solicitud se completó con éxito.** Son los códigos más deseables y frecuentes en logs de sitios web funcionando correctamente.
200	Success	OK	**La solicitud ha tenido éxito.** La respuesta estándar para solicitudes HTTP exitosas. Muy común.
201	Success	Created	La solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado. Común en APIs REST (ej. creación de un usuario).
204	Success	No Content	El servidor procesó la solicitud con éxito, pero no devuelve ningún contenido. Común en PUT/DELETE donde no hay contenido para retornar.
3xx	**Redirection**	Se necesita una acción adicional para completar la solicitud.	Indican que el cliente necesita realizar alguna acción adicional para completar su solicitud, generalmente una redirección.
301	Redirection	Moved Permanently	El recurso solicitado ha sido asignado permanentemente a una nueva URI. El cliente debe redirigir a la nueva URL y usarla en el futuro. Muy importante para SEO.
302	Redirection	Found (Previously "Moved Temporarily")	El recurso solicitado se encuentra temporalmente en una URI diferente. El cliente debe continuar usando la URI original para futuras solicitudes.

304	Redirection	Not Modified	El recurso no ha sido modificado desde la última vez que el cliente lo solicitó (basado en encabezados condicionales como `If-Modified-Since`). Reduce el uso de ancho de banda.
4xx	**Client Error**	La solicitud contiene una sintaxis incorrecta o no puede completarse.	**Indican errores causados por el cliente (ej. URL mal escrita, falta de autenticación).** Muy importantes para identificar problemas de usuario o posibles ataques (escaneo de vulnerabilidades).
400	Client Error	Bad Request	El servidor no pudo entender la solicitud debido a una sintaxis incorrecta.
401	Client Error	Unauthorized	La solicitud requiere autenticación del usuario.
403	Client Error	Forbidden	El servidor se niega a autorizar la solicitud. A diferencia del 401, la autenticación no ayudará.
404	Client Error	Not Found	**El recurso solicitado no se encontró en el servidor.** Muy común si el usuario introduce una URL incorrecta o un enlace roto.
405	Client Error	Method Not Allowed	El método de solicitud (ej. GET, POST) no está permitido para el recurso indicado.
429	Client Error	Too Many Requests	El usuario ha enviado demasiadas solicitudes en un período de tiempo determinado (rate limiting). Útil para detectar ataques de fuerza bruta o escaneo.
5xx	**Server Error**	El servidor falló en cumplir una solicitud aparentemente válida.	**Indican que el servidor encontró un error al intentar procesar la solicitud.** Estos son críticos y requieren atención inmediata del administrador, ya que señalan problemas con el servidor o la aplicación.
500	Server Error	Internal Server Error	**Un mensaje de error genérico del servidor.** Indica que algo salió mal en el servidor, pero no se puede especificar el tipo de error. Se debe revisar el `error.log` del servidor o logs de la aplicación.

502	Server Error	Bad Gateway	El servidor, mientras actuaba como puerta de enlace o proxy, recibió una respuesta no válida de un servidor ascendente. Común en configuraciones de proxy inverso (Nginx con backend Apache/PHP-FPM).
503	Server Error	Service Unavailable	El servidor no está listo para manejar la solicitud (ej. sobrecargado o en mantenimiento).
504	Server Error	Gateway Timeout	El servidor, mientras actuaba como puerta de enlace o proxy, no recibió una respuesta a tiempo de un servidor ascendente. También común en configuraciones de proxy inverso.

7. Conceptos Importantes sobre Logs

- **Daemon de Syslog (`rsyslogd`, `syslog-ng`):** Son los programas que recogen y distribuyen los mensajes de log de diferentes fuentes a los archivos apropiados.
- **Niveles de Severidad:** Los mensajes de log tienen niveles de severidad (prioridad), como:
  - `emerg` (emergencia, sistema inutilizable)
  - `alert` (alerta, acción inmediata necesaria)
  - `crit` (crítico)
  - `err` (error)
  - `warning` (advertencia)
  - `notice` (normal pero significativo)
  - `info` (informativo)
  - `debug` (mensajes de depuración)
- **Facilidades:** Los mensajes de log también se clasifican por "facilidades" (origen), como `auth`, `kern`, `mail`, `daemon`, `syslog`, etc.
- **Rotación de Logs:** Es el proceso de archivar y, opcionalmente, comprimir y eliminar periódicamente los archivos de log antiguos para liberar espacio en disco.