Rsyslog

Bastionado de Redes y Sistemas

Autor: Eric Serrano Marín

Índice

1. Introducción

Este proyecto se centra en configurar Rsyslog para gestionar registros de eventos. Incluye la configuración del entorno, pruebas de funcionamiento, y la integración con Nginx para la gestión de logs web. Además, se analizarán los registros de Windows y se desarrollará un script para copiar temporalmente los archivos access y error, comparándolos con las copias de seguridad anteriores. El objetivo es asegurar una gestión eficiente y precisa de los archivos de registro.

2. Guía de Configuración

En nuestro entorno existen dos contenedores con Rsyslog preconfigurado para realizar logs remotos. Nuestro contenedor srv es quien hará de servidor centralizado de logs y el contenedor cli, será el cliente.

Configuración Rsyslog Servidor

```
/etc # tail -f rsyslog.conf
#$DefaultNetstreamDriverKeyFile /etc/ssl/rsyslog/rsyslog_SERVER.key.pem
#$InputTCPServerStreamDriverMode 1 # run driver in TLS-only mode
#$InputTCPServerStreamDriverAuthMode x509/name # enable peer authentication
#$InputTCPServerStreamDriverPermittedPeer bar # authorize client named bar (one line per client)
#$TCPServerRun 10514 # start a TCP syslog server at port 10514
# UDP Syslog Server:
$ModLoad imudp.so  # provides UDP syslog reception
$UDPServerRun 514 # start a UDP syslog server at standard port 514
```

Esta configuración establece Rsyslog para recibir registros tanto por TCP como por UDP, con configuraciones específicas para comunicación segura por TCP usando cifrado TLS y autenticación basada en certificados, así como definiendo dónde deben estar ubicadas las claves y certificados de cifrado en el sistema de archivos.

Configuración Rsyslog Cliente

```
/etc # tail -f rsyslog.conf
        action.resumeInterval="30"
)

# Receive messages from remote host via UDP
# for parameters see http://www.rsyslog.com/doc/imudp.html
#module(load="imudp")  # needs to be done just once
#input(
#       type="imudp"
#       port="514"
#)
```

Esta configuración del cliente Rsyslog está preparada para recibir mensajes de un host remoto a través de UDP

Después de esto, hay que comprobar que los logs del cliente se registran en el servidor de logs.

Para ello, desde el cliente usaremos logger para enviarnos un mensaje de prueba.

```
C:\Users\alumno\Downloads\rsyslog>docker exec -it rsyslog-cli-1 /bin/sh
/ # pwd
/
/ # logger "Mensaje de prueba desde el cliente"
/ #
```

Ahora desde el servidor miraremos si nos ha llegado el mensaje de prueba.

```
/var/log # ls
messages
/var/log # tail -f messages
2024-03-14T18:53:19.395193+00:00 srv rsyslogd: [origin software="rsyslogd" swVersion="8.9.0" x-pid="1" x-info="http://www.rsys
log.com"] start
2024-03-14T18:53:20+00:00 cliente : [origin software="rsyslogd" swVersion="8.2310.0" x-pid="116" x-info="https://www.rsyslog.c
om"] start
2024-03-14T19:13:19.476342+00:00 srv rsyslogd: -- MARK --
2024-03-14T19:13:19+00:00 cliente : -- MARK --
2024-03-14T19:33:19.507228+00:00 srv rsyslogd: -- MARK --
2024-03-14T19:33:19+00:00 cliente : -- MARK --
2024-03-14T19:38:03+00:00 cliente root: Mensaje de prueba desde el cliente
^C
/var/log #
```

Podemos observar que ha llegado el mensaje correctamente.

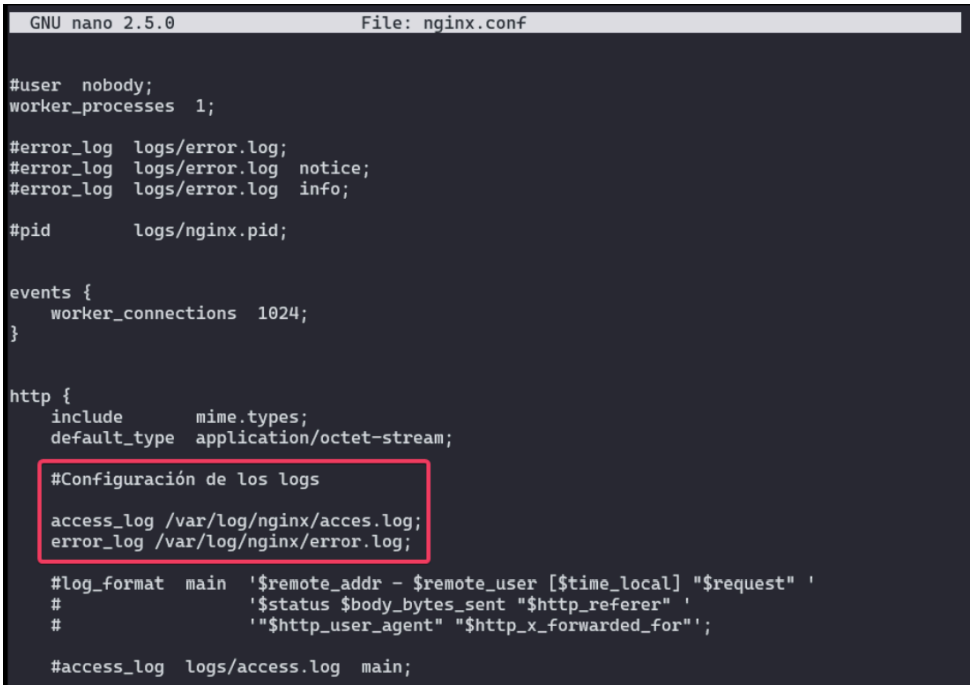
3. Configuración de Nginx

Configuraremos un servidor Nginx para que los logs se registren en el servidor de logs.

Instalación de Nginx

```
/ # apk add nginx
(1/3) Installing nginx-initscripts (1.8.0-r0)
Executing nginx-initscripts-1.8.0-r0.pre-install
(2/3) Installing pcre (8.38-r1)
(3/3) Installing nginx (1.8.1-r2)
Executing busybox-1.24.1-r7.trigger
OK: 9 MiB in 22 packages
```

Cambios en la configuración de Nginx



La línea **access_log /var/log/nginx/access.log;** habilita el registro de accesos. Nginx escribirá los registros de acceso en el archivo **/var/log/nginx/access.log**, esto incluye información sobre cada solicitud que el servidor web recibe, como la dirección IP del cliente, hora de la solicitud, recurso solicitado y más.

Por otro lado, la línea **error_log /var/log/nginx/error.log;** habilita el registro de errores. Se escribirán los errores en el archivo **/var/log/nginx/error.log**, esto incluye mensajes de error generados por el servidor web, lo cual es útil para la depuración y monitoreo del estado del servidor.

Comprobación del funcionamiento

Log de acceso

Vamos a hacer un curl al servidor web.

```
/ # curl 172.18.0.3
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

Comprobamos el archivo access.log.

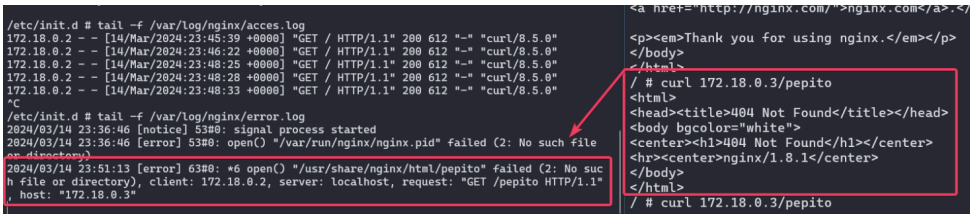
IP servidor: 172.18.0.3 IP cliente: 172.18.0.2

```
/etc/init.d # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:03
          inet addr:172.18.0.3  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3202 errors:0 dropped:0 overruns:0 frame:0
          TX packets:938 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4434609 (4.2 MiB)  TX bytes:67743 (66.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:56 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4396 (4.2 KiB)  TX bytes:4396 (4.2 KiB)

/etc/init.d # tail -f /var/log/nginx/acces.log
172.18.0.2 - - [14/Mar/2024:23:45:39 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/8.5.0"
172.18.0.2 - - [14/Mar/2024:23:46:22 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/8.5.0"
```

Prueba del funcionamiento en gif: **https://imgur.com/XE7GsKv**

Log de error

Para ello, vamos a hacer un curl al servidor con una ruta que no existe, con ello vamos a conseguir que nos de error.



Prueba del funcionamiento en gif: **https://imgur.com/Khk7Pxr**

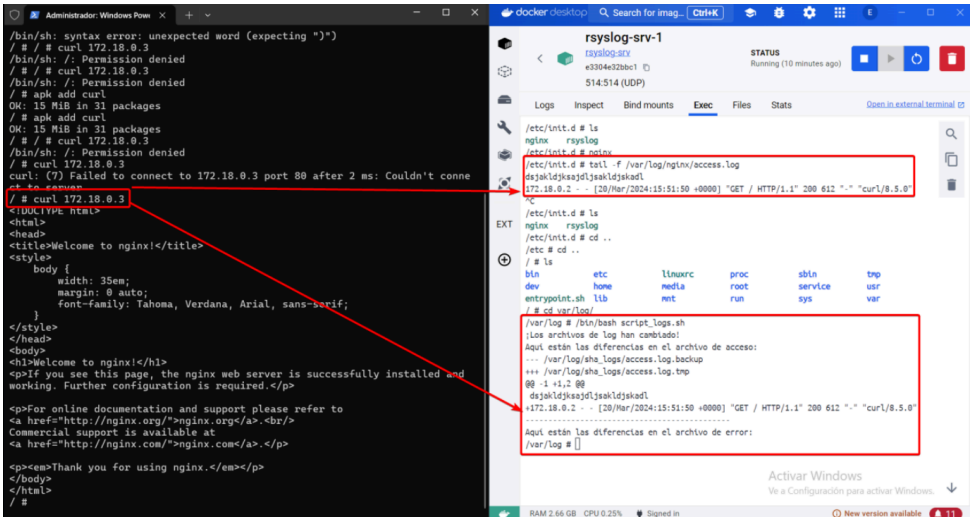
4. Script de Seguridad para la Gestión de Logs

```
#!/bin/sh
# Definir las rutas de los archivos de log y el directorio de
ACCESS_LOG="/var/log/nginx/access.log"
ERROR_LOG="/var/log/nginx/error.log"
BACKUP_DIR="/var/log/sha_logs"

# Crear el directorio de copias de seguridad si no existe
mkdir -p "$BACKUP_DIR"

# Crear copias de seguridad anteriores si no existen
if [ ! -f "$BACKUP_DIR/access.log.backup" ]; then
 cp "$ACCESS_LOG" "$BACKUP_DIR/access.log.backup"
fi
if [ ! -f "$BACKUP_DIR/error.log.backup" ]; then
 cp "$ERROR_LOG" "$BACKUP_DIR/error.log.backup"
fi

# Crear nuevas copias de seguridad de los archivos de log
cp "$ACCESS_LOG" "$BACKUP_DIR/access.log.tmp"
cp "$ERROR_LOG" "$BACKUP_DIR/error.log.tmp"

# Comparar las copias de seguridad anteriores con las actuale
if cmp -s "$BACKUP_DIR/access.log.backup" "$BACKUP_DIR/access
 cmp -s "$BACKUP_DIR/error.log.backup" "$BACKUP_DIR/error.log
 echo "Los archivos de log son iguales."
else
 echo "¡Los archivos de log han cambiado!"
 echo "Aquí están las diferencias en el archivo de acceso:"
 diff "$BACKUP_DIR/access.log.backup" "$BACKUP_DIR/access.log
 echo "----------------------------------------"
 echo "Aquí están las diferencias en el archivo de error:"
 diff "$BACKUP_DIR/error.log.backup" "$BACKUP_DIR/error.log.t
fi

# Mover las copias de seguridad temporales a las anteriores
mv "$BACKUP_DIR/access.log.tmp" "$BACKUP_DIR/access.log.backu
mv "$BACKUP_DIR/error.log.tmp" "$BACKUP_DIR/error.log.backup"
```

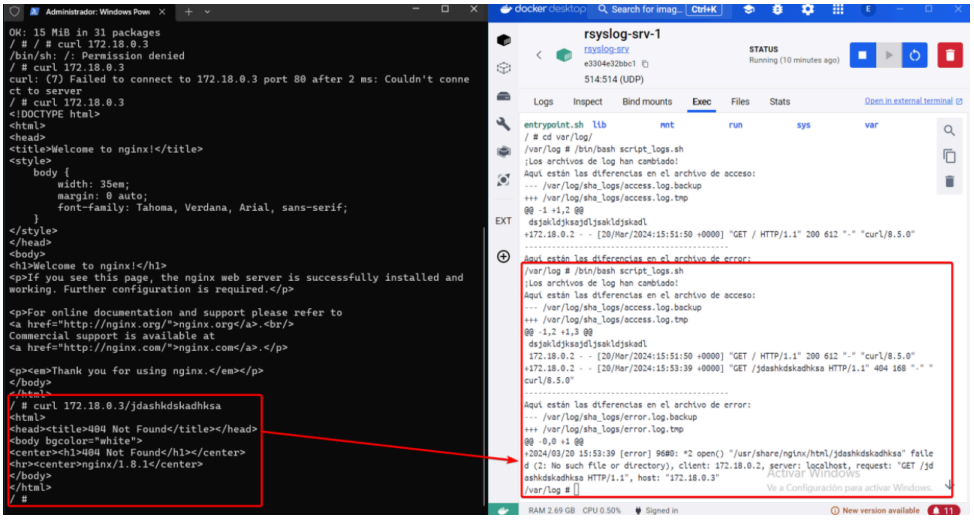
Este script hace copias temporales de los archivos de registro **(access.log.tmp y error.log.tmp)** y las compara con las copias de seguridad anteriores **(access.log.backup y error.log.backup)**. Si ve algún cambio, muestra las diferencias y luego actualiza las copias de seguridad anteriores con las nuevas copias temporales.

Si no hay cambios, simplemente dice que los archivos de registro son iguales.

En la siguiente captura podemos observar cómo desde el cliente hago un curl y el script me detecta que ha cambiado.



En la siguiente captura lo mismo, pero esta vez con errors, lanzando un curl a una dirección no existente.



Ahora, vamos a borrar manualmente alguna línea de nuestros archivos **access.log** y **errors.log**, y así comprobar que nuestro script esta funcionando correctamente.

Voy a borrar la línea del medio.

```
GNU nano 2.5.0                                File: access.log

dsjakldjksajdljsakldjskadl
172.18.0.2 - - [20/Mar/2024:15:51:50 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/8.5.0"
172.18.0.2 - - [20/Mar/2024:15:53:39 +0000] "GET /jdashkdskadhksa HTTP/1.1" 404 168 "-" "curl/8.5.0"
```

Podemos observar que nos avisa de los que archivos de log han cambiado y nos dice que ha cambiado. Nos aparece el símbolo "**-**" que significa que se ha borrado.

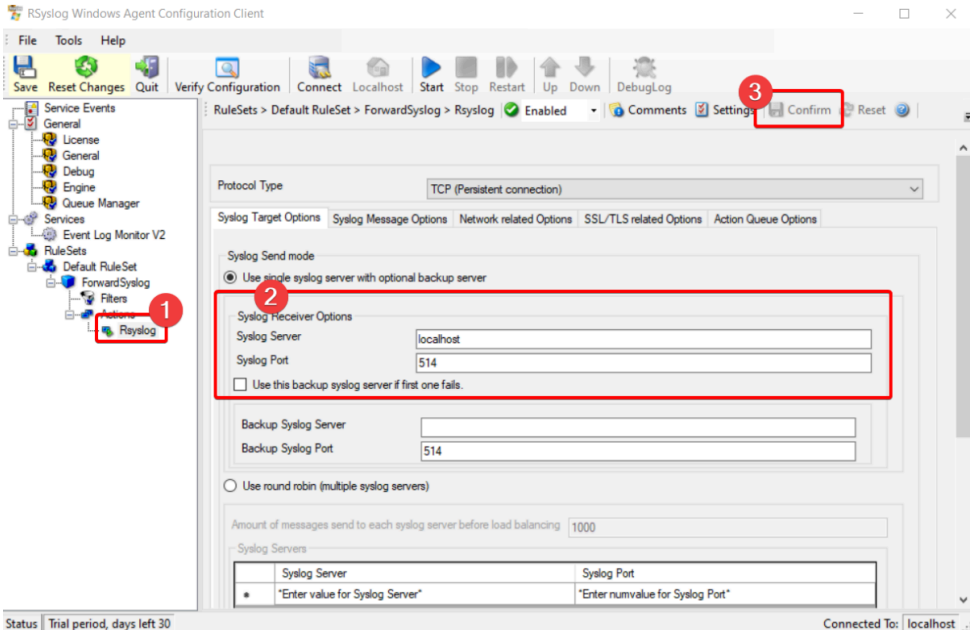
```
/var/log # /bin/bash script_logs.sh
¡Los archivos de log han cambiado!
Aquí están las diferencias en el archivo de acceso:
--- /var/log/sha_logs/access.log.backup
+++ /var/log/sha_logs/access.log.tmp
@@ -1,3 +1,2 @@
 dsjakldjksajdljsakldjskadl
-172.18.0.2 - - [20/Mar/2024:15:51:50 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/8.5.0"
 172.18.0.2 - - [20/Mar/2024:15:53:39 +0000] "GET /jdashkdskadhksa HTTP/1.1" 404 168 "-" "curl/8.5.0"
```

Registro de logs de Windows

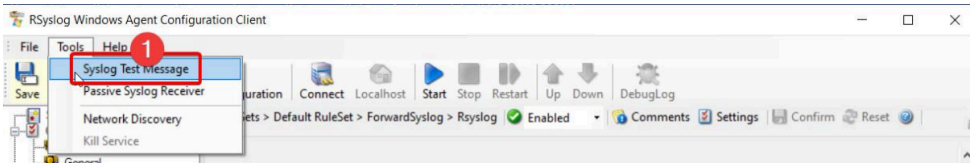
Para esta parte vamos a instalar en un cliente Windows un agente de log remoto. En este caso usaremos Rsyslog.

Configuraremos el receiver, que va a ser nuestra máquina rsyslog que tenemos en Docker.

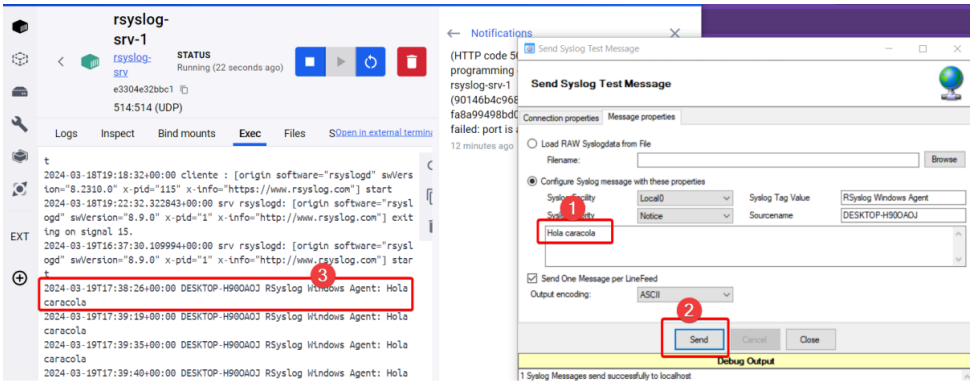
Rule Sets > Default Rule Set > ForwardSyslog > Rsyslog.



Después nos dirigiremos a **Tools > Syslog Test Message** para enviar un mensaje para ver si está funcionando correctamente.



Con el comando **tail -f messages** podemos observar como al enviar el mensaje lo recibimos correctamente.



Funcionamiento en Gif: **https://imgur.com/LhGcjnd**

Conclusiones

1. **Automatización de la Seguridad**: La creación de scripts automatiza el proceso de copia de seguridad y comparación de logs, reduciendo la posibilidad de errores humanos y asegurando que los logs estén siempre protegidos.
2. **Detección de Cambios**: Utilizar un script en shell es una solución sencilla y eficiente que puede ser implementada rápidamente en cualquier sistema basado en Unix. Esto hace que la práctica sea accesible incluso para aquellos con conocimientos básicos de scripting.
3. **Mejora la Gestión de Logs**: Al tener copias de seguridad regulares y un sistema de comparación, se mejora la gestión de los logs, facilitando la auditoría y el análisis de eventos pasados en caso de incidentes de seguridad.
4. **Escalabilidad**: Este enfoque puede ser fácilmente escalado para incluir más archivos de log o para integrarse con sistemas de monitoreo y alertas más avanzados, proporcionando una base sólida para una estrategia de seguridad más amplia.