

THOMAS ESCAFFRE

REMOTE LOGGING SYSTEM

february 2025



SUMMARY

01

INTRODUCTION

02

SET UP RSYSLOG

03

TLS SECURITY

04

CONCLUSION

01

INTRODUCTION

Introduction

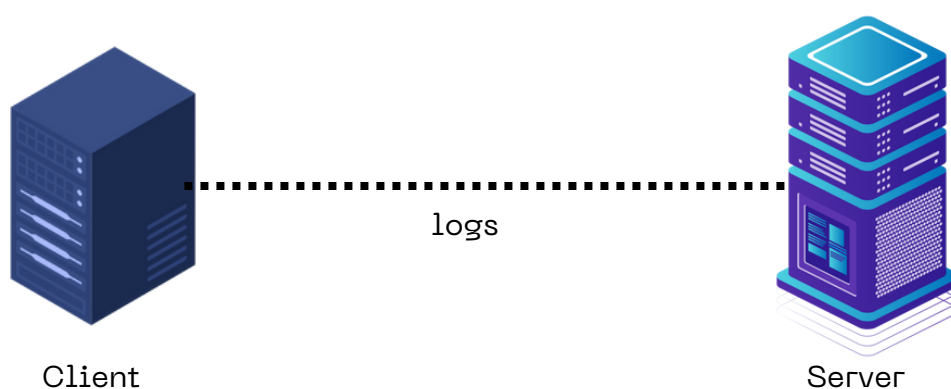
Le contexte

In the modern digital landscape, logging and monitoring are crucial components of cybersecurity and system administration. A well-implemented remote logging system allows organizations to centralize logs from multiple machines, ensuring efficient analysis, troubleshooting, and detection of security threats.

This project focuses on setting up a secure remote logging system where logs from a Linux virtual machine (VM) are transmitted to a centralized logging server over a secured channel using TLS encryption. The goal is to ensure data integrity, confidentiality, and availability while enabling efficient log management.

Such a system is essential for incident response, compliance with security standards (e.g., GDPR, ISO 27001), and proactive threat detection. By implementing this solution in a controlled virtualized environment, we gain practical insights into system hardening, network security, and log analysis techniques.

To achieve this, we will first focus on the installation and configuration of the remote logging system with rsyslog. Once this setup is complete, we will set up the TLS communication.



02

SET UP VMS



Set-Up VMs

How to install the configuration

In this project, you will configure rsyslog to securely transmit logs between two virtual machines (VMs). Each VM plays a crucial role in the log management system:

Linux Client



The client VM runs Ubuntu 2022 Linux, which is responsible for generating and sending system logs to the server. This machine will act as the source of log data, and through rsyslog, it will securely forward these logs to the Ubuntu server. The client VM will be configured to establish a TLS (Transport Layer Security) connection to ensure that the communication is encrypted and secure during the log transfer.

Linux Server



The server VM runs Ubuntu 2022 and is configured to receive logs from the client. This machine will act as the centralized log receiver, where all logs from the client will be stored and monitored. In this setup, Ubuntu will be configured with rsyslog to listen for incoming logs on a specific port and process them securely over the TLS connection. The server will also be responsible for storing the logs in dedicated log files for future analysis or troubleshooting.

Set-Up VMs

How to install the configuration

In this project, we will use rsyslog, a powerful and flexible logging system for Unix-based operating systems. rsyslog is an advanced version of the traditional syslog daemon, offering enhanced features such as high-performance log processing, remote log transmission, and secure communication using TLS encryption.

Why Use rsyslog?

- 1** Centralized Log Management – Instead of storing logs on multiple machines, rsyslog allows us to collect and centralize all logs on a single log server. This makes monitoring, troubleshooting, and auditing much easier.
- 2** Secure Log Transmission – Using TLS encryption, rsyslog ensures that log data is securely transmitted between the client and the server. This prevents unauthorized access or tampering.
- 3** Efficient and Scalable – rsyslog can handle a high volume of logs, filter log messages based on rules, and forward logs to various destinations (files, databases, or external monitoring tools like Elasticsearch).
- 4** Improved Security & Compliance – By collecting logs centrally, security teams can detect suspicious activity, enforce compliance regulations, and respond to incidents more effectively.

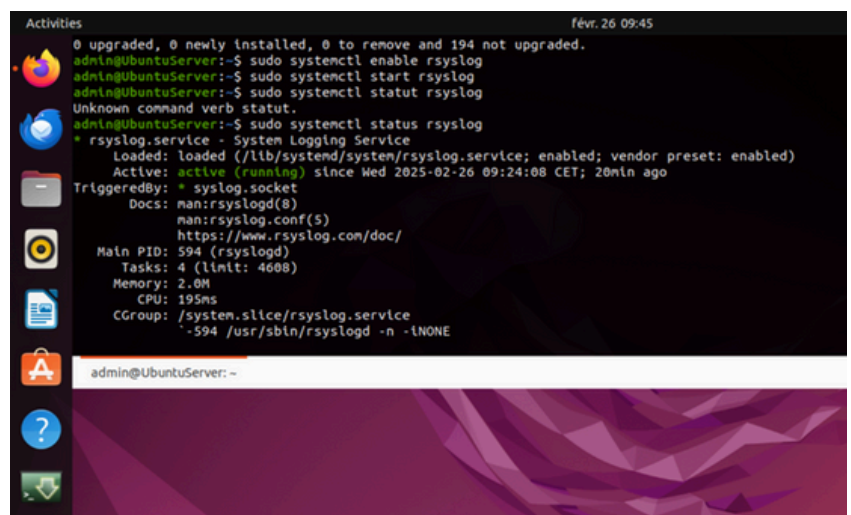
How Will We Use rsyslog?

- The Client VM will send system logs to the log server.
- The VM log server will receive, store, and manage logs securely.
- A TLS-encrypted connection will be established between both machines to ensure secure communication.

As you can see here, we installed rsyslog on our two VMs with the command :

`sudo apt update && sudo apt install rsyslog -y`

Then we test whether rsyslog is working.



```
Activities
févr. 26 09:45
0 upgraded, 0 newly installed, 0 to remove and 194 not upgraded.
admin@UbuntuServer:~$ sudo systemctl enable rsyslog
admin@UbuntuServer:~$ sudo systemctl start rsyslog
admin@UbuntuServer:~$ sudo systemctl status rsyslog
Unknown command verb status.
admin@UbuntuServer:~$ sudo systemctl status rsyslog
* rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2025-02-26 09:24:08 CET; 20min ago
     TriggeredBy: * syslog.socket
    Docs: man:rsyslogd(8)
          man:rsyslog.conf(5)
          https://www.rsyslog.com/doc/
   Main PID: 594 (rsyslogd)
     Tasks: 4 (limit: 4000)
    Memory: 2.0M
       CPU: 195ms
   CGroup: /system.slice/rsyslog.service
           └─594 /usr/sbin/rsyslogd -n -lNONE

admin@UbuntuServer:~$
```

Set-Up VMs

How to install the configuration

For this project, it is essential that both virtual machines can communicate with each other to allow secure log transmission. To achieve this, we have configured them to use an internal network in VirtualBox.

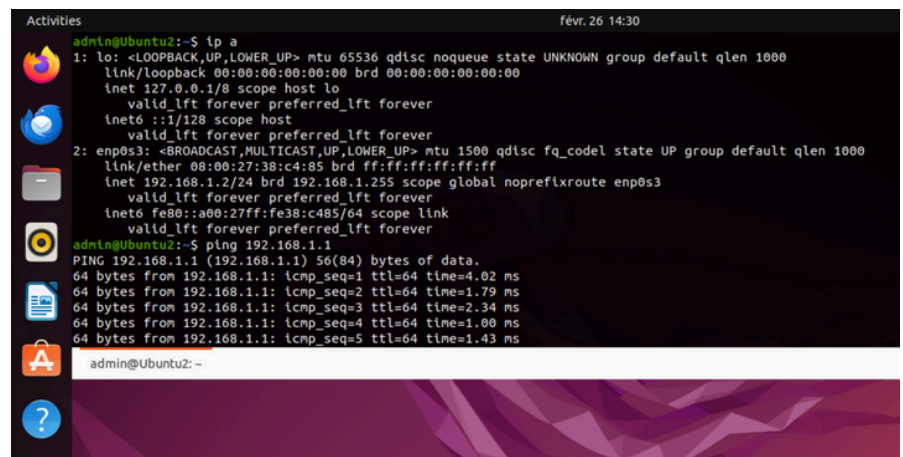
Why Use an Internal Network?

- 1** Direct Communication – By using an internal network, the VMs can communicate without external interference, improving security and isolation.
- 2** No Internet Dependency – The internal network allows communication even without internet access, ensuring stability and reducing external risks.
- 3** Controlled IP Addressing – Each VM is assigned a static or DHCP-based IP address within the internal network, ensuring they can reliably reach each other.

Assigning IP Addresses

- The Server is assigned a specific IP (e.g., 192.168.1.1).
- The Client is assigned another IP in the same range (e.g., 192.168.1.2).

As you can see here, the two VMs can communicate :



```
admin@Ubuntu2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:38:c4:85 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 brd 192.168.1.255 scope global noprefroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe38:c485/64 scope link
        valid_lft forever preferred_lft forever
admin@Ubuntu2:~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data:
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=4.02 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=1.79 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=2.34 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=1.00 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=1.43 ms
```



Client : 192.168.1.2



Server : 192.168.1.1

ping 

Set-Up VMs

How to install the configuration

To begin, the system needs to be configured so that one machine acts as a log receiver while another sends its logs to it. The Server is the machine that will collect logs, and the Client is the machine that will forward its logs to the Server. In this setup, the Server's hostname is Server, and the Client's hostname is Client. The Server has the IP address 192.168.1.2, and the Client has the IP address 192.168.1.1.

Configuring the Server (Log Receiver)

The first step is to enable the Server to accept logs from remote machines. This requires modifying the rsyslog configuration file. By opening the file `/etc/rsyslog.conf` with a text editor such as Vim, it is possible to enable both UDP and TCP log reception. Inside the file, there are two important sections that need to be modified. The first one loads the necessary modules. By default, these lines exist in the configuration file but are commented out. The comment marks should be removed from the following lines:

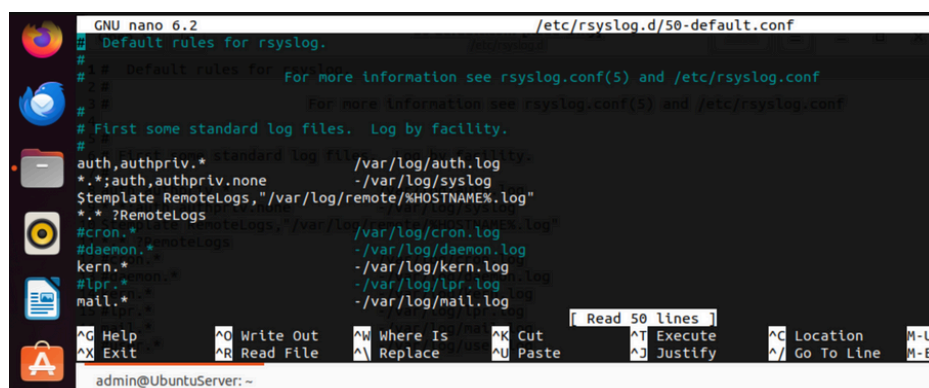


As usual after this modification : `sudo systemctl restart rsyslog`

And this commande to check if you have errors : `sudo systemctl status rsyslog`

if you don't want to have the logs of the Client in the same file as the server, you have to modificate `/etc/rsyslog.d/50-default.conf` and add :

`$template RemoteLogs,"/var/log/remote/%HOSTNAME%.log"`
`*.* ?RemoteLogs`



then you have to change the permissions of the file :

`sudo chown syslog:adm /var/log/remote/`
`sudo chmod 755 /var/log/remote/`

Set-Up VMs

How to install the configuration

Configuring the Client (Log Sender)

The next step is to configure the Client to send its logs to the Server. This also requires modifying the rsyslog configuration file. On the Client machine, the file `/etc/rsyslog.d/50-default.conf` (the files in rsyslog.d are configurations files and are compiled in numerical order) should be opened and edited. There is a specific line in this file that dictates where logs are stored locally. This line should be commented out to prevent logs from being written only on the Client itself. The line that needs to be commented is:

```
# *.*;auth,authpriv.none -/var/log/syslog
```

After commenting out this line, a new one must be added at the end of the file to forward logs to the Server:

```
*.* @@192.168.1.1
```

This line means that all logs (*.*) will be sent to the Server at IP 192.168.1.1 using TCP (indicated by @@). If UDP is preferred, the line should be written as:

```
*.* @192.168.1.1
```

As usual after this modification : `sudo systemctl restart rsyslog`

And this commande to check if you have errors : `sudo systemctl status rsyslog`



Client : 192.168.1.2

Ready ✓



Server : 192.168.1.1

Ready ✓

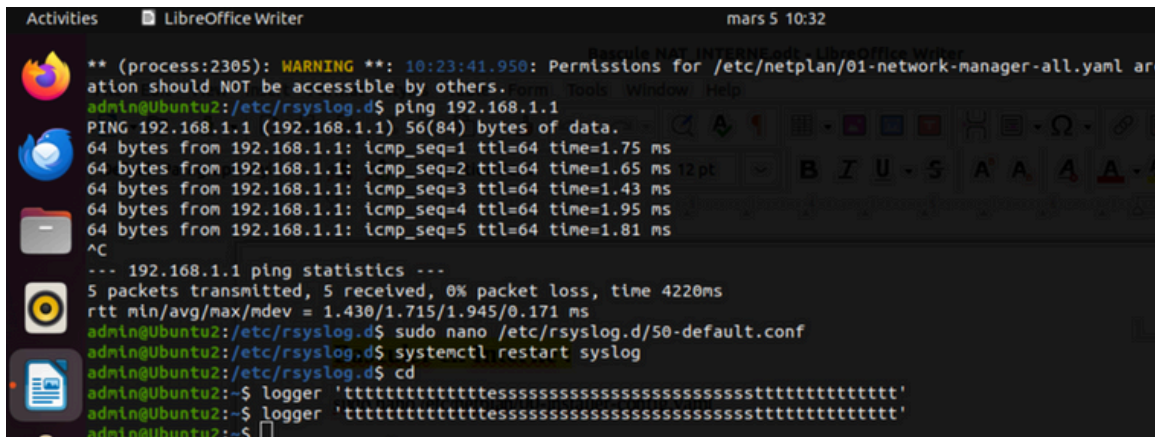
Set-Up VMs

How to install the configuration

Test Set-up

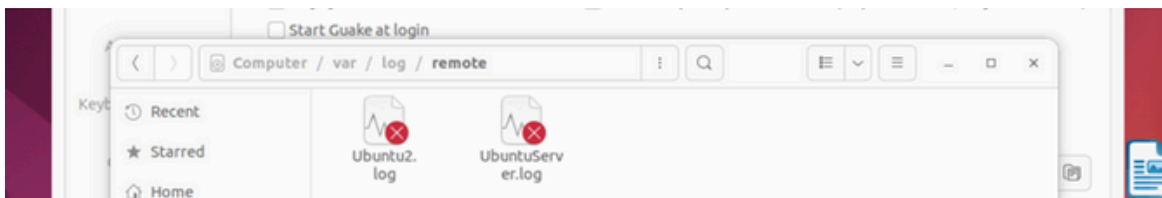
Once both the Server and Client are configured, it is important to verify that logs are being correctly transmitted. On the Client machine, a test log can be generated using the following command:

```
logger -p local0.info "tttttttttessssssstttttttt"
```

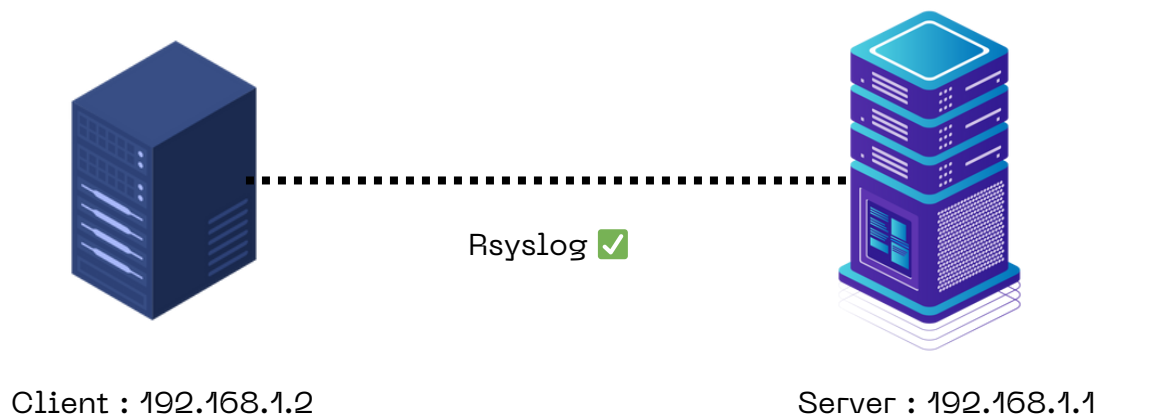


```
admin@Ubuntu2:/etc/rsyslog.d$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data:
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.75 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=1.65 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=1.43 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=1.95 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=1.81 ms
^C
--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4220ms
rtt min/avg/max/mdev = 1.430/1.715/1.945/0.171 ms
admin@Ubuntu2:/etc/rsyslog.d$ sudo nano /etc/rsyslog.d/50-default.conf
admin@Ubuntu2:/etc/rsyslog.d$ systemctl restart rsyslog
admin@Ubuntu2:/etc/rsyslog.d$ cd
admin@Ubuntu2:~$ logger 'tttttttttttessssssssssssssssssssssssssssstttttttttttt'
admin@Ubuntu2:~$ logger 'tttttttttttessssssssssssssssssssssssssssstttttttttttt'
admin@Ubuntu2:~$
```

then, check on your server the file where the logs should appeared in the server :



```
'builtin:omfwd') [v8.2112.0 try https://www.rsyslog.com/e/2359 ]
18947 Mar  5 10:31:26 Ubuntu2 systemd-resolved[397]: Using degraded feature set TCP instead of UDP
for DNS server 8.8.4.4.
18948 Mar  5 10:31:26 Ubuntu2 admin: ttttttttttttessssssssssssssssssssssssssssstttttttttttt
18949 Mar  5 10:31:27 UbuntuServer systemd-resolved[380]: Using degraded feature set TCP instead of
UDP for DNS server 8.8.4.4.
18950 Mar  5 10:31:32 UbuntuServer dbus-daemon[1199]: [session uid=1001 pid=1199] Activating
service name='org.gnome.gedit' requested by ':1.112' (uid=1001 pid=2414 comm="/usr/bin/
nautilus --gapplication-service" label="unconfined")
18951 Mar  5 10:31:32 UbuntuServer dbus-daemon[1199]: [session uid=1001 pid=1199] Successfully
```



03

TLS SECURITY



TLS Security

Securing Rsyslog with TLS

Introduction to TLS in Rsyslog

Transport Layer Security (TLS) is a cryptographic protocol designed to provide secure communication over a network. It ensures data integrity, confidentiality, and authentication between systems. In our Rsyslog project, implementing TLS will allow us to encrypt log messages exchanged between the client and the server, preventing unauthorized access and data tampering.

By default, Rsyslog sends logs in plaintext, which makes them vulnerable to interception. With TLS, we can encrypt these communications and authenticate both the client and the server using digital certificates. This guarantees that logs are transmitted securely, mitigating the risk of data leaks and ensuring compliance with security best practices.

To achieve this, we will:

1. Generate and configure a Certificate Authority (CA) to sign certificates.
2. Create secure private and public key pairs for both the Rsyslog server and clients.
3. Enable TLS encryption in Rsyslog configuration files to enforce secure log transmission.

By following this approach, we ensure that only authorized systems can send and receive logs, strengthening the security of our logging infrastructure.





TLS Security

Securing Rsyslog with TLS

Installation

To secure our Rsyslog installation, we first need to install some tools as we will use GnuTLS. Run the following command as Root on both the server and client:

```
apt install -y rsyslog-gnutls gnutls-bin gnutls-doc
```

Generating Secure Keys

Rsyslog can use the TLS protocol to authenticate both the client and server and to encrypt their communications. The principle is as follows:

Each machine has its own unique private/public key pair and a certificate signed by a common Certificate Authority (CA).

In this example, the CA will be generated by our server.

On the Server

- Generating the Certificate Authority (CA) Certificate

Run the following command to generate the private key for the Certificate Authority:

```
certtool --generate-privkey --outfile ca-key.pem --sec-param High
```

This key will be used by the CA to sign the certificates for the server and client.

- Next, generate a self-signed certificate associated with the private key we just created:

```
certtool --generate-self-signed --load-privkey ca-key.pem --outfile ca.pem
```

The program will ask for various details. You can leave all sections blank except the Common Name that must be the full name of the machine.

What you need to full :

Common name: rsyslog-server

The certificate will expire in (days): 3650

Does the certificate belong to an authority? (y/N): y

Enter a dnsName of the subject of the certificate: rsyslog-server

Is the above information ok? (y/N): y

The CA certificate is now generated.



TLS Security

Securing Rsyslog with TLS

- Generating the Server Certificate

Generate the server's private key:

```
certtool --generate-privkey --outfile rsyslog-server-key.pem --sec-param High
```

Now create a certificate signing request (CSR):

```
certtool --generate-request --load-privkey rsyslog-server-key.pem --outfile rsyslog-server-certificat-request.pem
```

The lines you have to full :

Country name (2 chars): FR

Common name: rsyslog-server

Enter a dnsName of the subject of the certificate: rsyslog-server

Is this a TLS web client certificate? (y/N): y

Is this a TLS web server certificate? (y/N): y

Now generate the signed server certificate using the CA certificate, CA private key, and the CSR:

```
certtool --generate-certificate --load-request rsyslog-server-certificat-request.pem --outfile rsyslog-server-certificat.pem --load-ca-certificate ca.pem --load-ca-privkey ca-key.pem
```

Again, provide responses to the prompts:

The certificate will expire in (days): 365

Is this a TLS web client certificate? (y/N): y

Is this a TLS web server certificate? (y/N): y

Enter a dnsName of the subject of the certificate: rsyslog-server



TLS Security

Securing Rsyslog with TLS

On the Client

The steps are mostly the same, but with some file exchanges. The CSR needs to be sent to the server for signing, then the signed certificate and the CA certificate are sent back to the client.

- Generate the client's private key:

```
certtool --generate-privkey --outfile rsyslog-client-key.pem --sec-param High
```

- Generate the CSR:

```
certtool --generate-request --load-privkey rsyslog-client-key.pem --outfile rsyslog-client-certificat-request.pem
```

What you have to full :

Common name: rsyslog-client

Enter a dnsName of the subject of the certificate: rsyslog-client

Is this a TLS web client certificate? (y/N): y

Is this a TLS web server certificate? (y/N): y

- Send the CSR to the server:

```
scp ./rsyslog-client-certificat-request.pem root@192.168.1.100:/etc/ssl/certs/rsyslog/
```

- On the server, generate the signed client certificate:

```
certtool --generate-certificate --load-request rsyslog-client-certificat-request.pem --outfile rsyslog-client-certificat.pem --load-ca-certificate ca.pem --load-ca-privkey ca-key.pem
```

(full the same lines as usual)

- Send the signed client certificate and CA certificate back to the client :

```
scp ./rsyslog-client-certificat.pem root@192.168.1.101:/etc/ssl/certs/rsyslog/
```

```
scp ./ca.pem root@192.168.1.101:/etc/ssl/certs/rsyslog/
```


TLS Security

Securing Rsyslog with TLS

Files Obtained

On the server:

- ca-key.pem: CA private key
- ca.pem: CA certificate
- rsyslog-server-key.pem: Server private key
- rsyslog-server-certificat-request.pem: Server CSR
- rsyslog-server-certificat.pem: Server signed certificate



On the client:

- ca.pem: CA certificate
- rsyslog-client-key.pem: Client private key
- rsyslog-client-certificat-request.pem: Client CSR
- rsyslog-client-certificat.pem: Client signed certificate

Configuring TLS

On the Server

- Edit `/etc/rsyslog.conf` and modify the module entry:

```
module(  
    load="imtcp"  
    StreamDriver.Name="gtls"  
    StreamDriver.Mode="1"  
    StreamDriver.Authmode="x509/name"  
    PermittedPeer="192.168.1.101"  
)
```

Modify the TCP input port:

```
input(type="imtcp" port="6514")
```

Add global TLS settings:

```
global(  
    DefaultNetstreamDriver="gtls"  
    DefaultNetstreamDriverCAFile="/etc/ssl/certs/rsyslog/ca.pem"  
    DefaultNetstreamDriverCertFile="/etc/ssl/certs/rsyslog/rsyslog-server-  
certificat.pem"  
    DefaultNetstreamDriverKeyFile="/etc/ssl/certs/rsyslog/rsyslog-server-  
key.pem"  
)
```

- Restart Rsyslog.



TLS Security

Securing Rsyslog with TLS

On the Client

- Add TLS configuration in `/etc/rsyslog.conf`:

```
global(  
    DefaultNetstreamDriver="gtls"  
    DefaultNetstreamDriverCAFile="/etc/ssl/certs/rsyslog/ca.pem"  
    DefaultNetstreamDriverCertFile="/etc/ssl/certs/rsyslog/rsyslog-client-  
certificat.pem"  
    DefaultNetstreamDriverKeyFile="/etc/ssl/certs/rsyslog/rsyslog-client-  
key.pem"  
)
```

- Restart Rsyslog.
- Check communication using:

```
tcpdump -i <network-interface> tcp port 6514 -X -s 0 -nn
```

04

CONCLUSION



Conclusion

What we did

Through this project, we successfully implemented a secure remote logging system using Rsyslog with TLS encryption. By configuring a client-server architecture, we ensured that logs are transmitted securely, protecting their integrity, confidentiality, and authenticity.

This setup provides several benefits:

- Enhanced security – Logs are encrypted, preventing unauthorized access.
- Centralized log management – Facilitates monitoring, troubleshooting, and compliance.
- Scalability – The system can be extended to multiple clients with minimal configuration.

By following industry best practices, we gained practical insights into network security, system administration, and cryptographic protocols. This project lays a solid foundation for further enhancements, such as log analysis integration with SIEM tools or automated alerting mechanisms.