

Laboratorio 8 Seguridad de sistemas

Nombre: Uño Astoraique Maria Fernanda

CI: 10468920

Propósito: El objetivo de este laboratorio es implementar un configurar un servidor RADIUS en un entorno de Linux para verificar la autenticidad de los usuarios mediante esquemas de autenticación, como la base de datos local, a partir de ello profundizar el conocimiento sobre la seguridad de información, además, les proporcionara experiencia práctica en la configuración y gestión de servicios de red críticos para la protección de sistemas empresariales.

DESCRIPCIÓN

Antes de iniciar las tres máquinas virtuales que se utilizarán, asegúrese de que el adaptador de red de cada una esté configurado en modo 'Bridged'.

Recursos: Sistema Operativo Kali Linux 2024.1 (<https://www.kali.org/get-kali/#kali-platforms>)

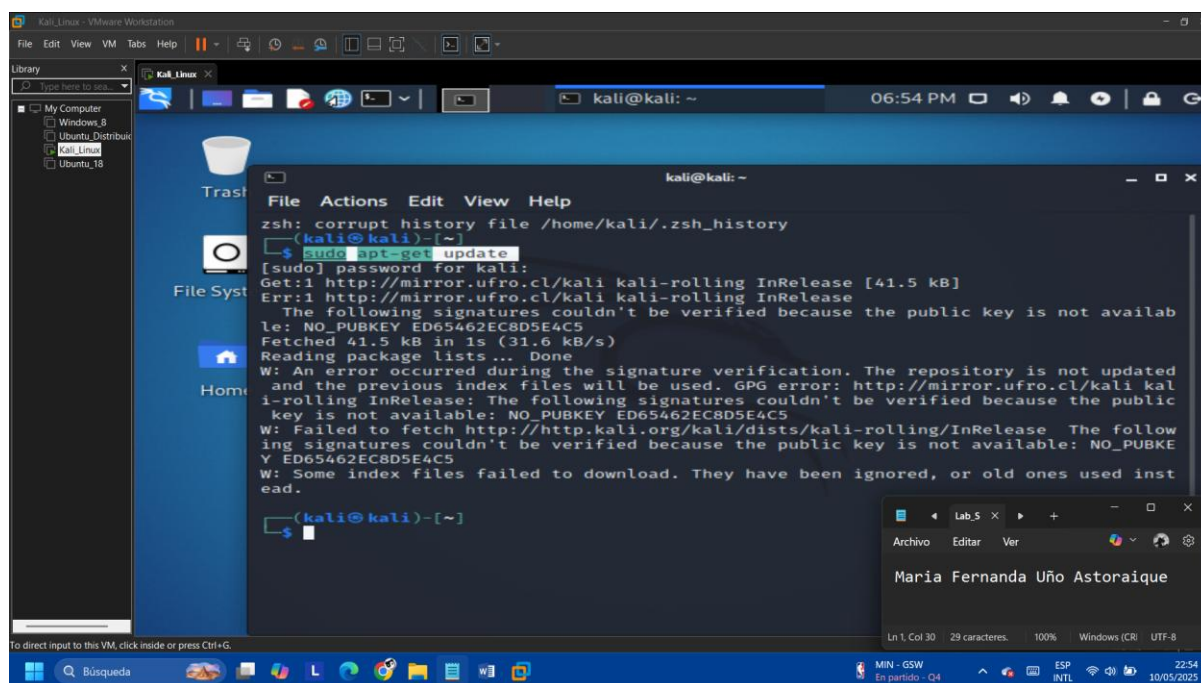
Sistema operativo Ubuntu 18 (<https://releases.ubuntu.com/18.04/>)

Sistema Operativo Windows 8

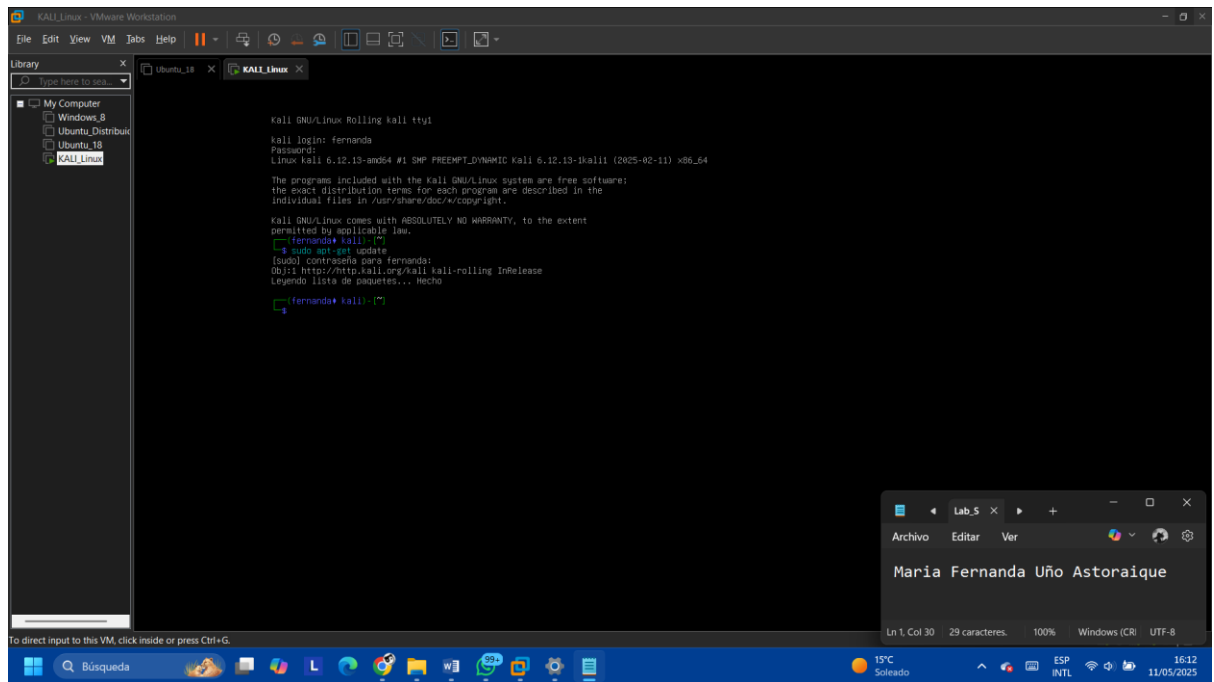
Freeradius V. 3.0

Instalación de Freeradius

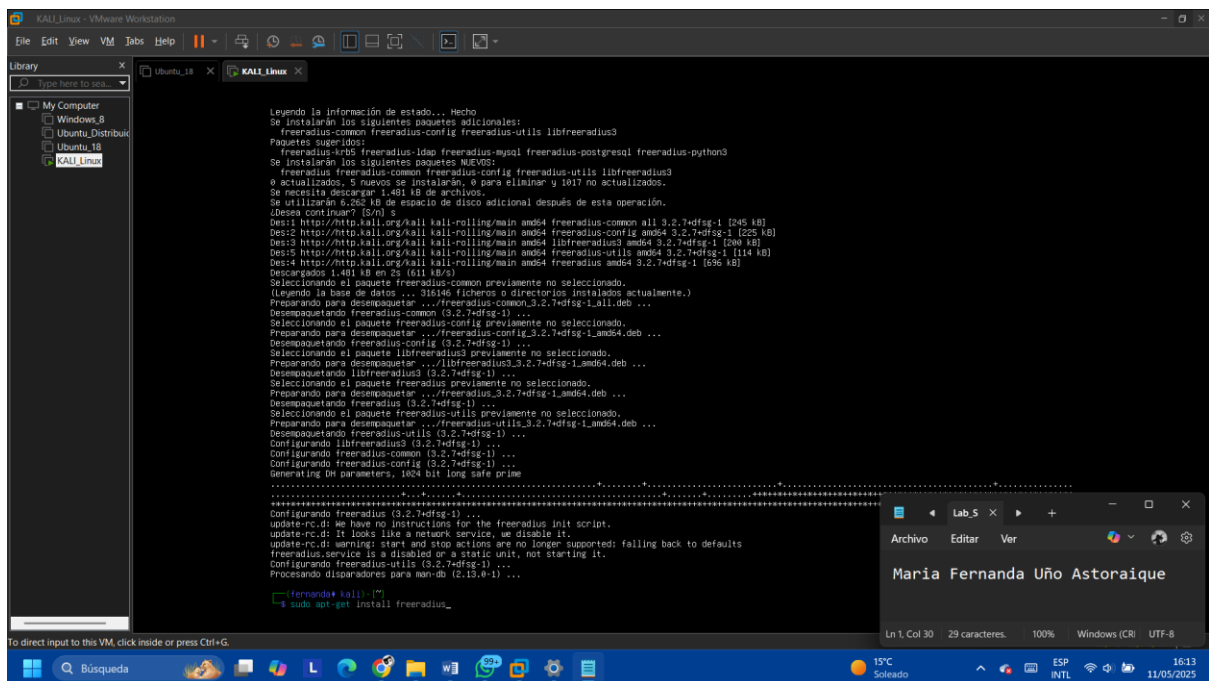
1. Inicialmente para la implementación de la Guía es imprescindible contar con la instalación de Freeradius. Por lo tanto, iniciamos con el usuario **kali** y contraseña **sistemas**, procederemos iniciando el proceso de actualización de los paquetes en Kali mediante el comando: **sudo apt-get update**.



```
kali@kali: ~  
zsh: corrupt history file /home/kali/.zsh_history  
[kali@kali]~  
[kali@kali]~$ sudo apt-get update  
[sudo] password for kali:  
Get:1 http://mirror.ufro.cl/kali kali-rolling InRelease [41.5 kB]  
Err:1 http://mirror.ufro.cl/kali kali-rolling InRelease  
The following signatures couldn't be verified because the public key is not available: NO_PUBKEY ED65462EC8D5E4C5  
Fetched 41.5 kB in 1s (31.6 kB/s)  
Reading package lists... Done  
W: An error occurred during the signature verification. The repository is not updated  
and the previous index files will be used. GPG error: http://mirror.ufro.cl/kali kal  
i-rolling InRelease: The following signatures couldn't be verified because the public  
key is not available: NO_PUBKEY ED65462EC8D5E4C5  
W: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease. The follow  
ing signatures couldn't be verified because the public key is not available: NO_PUBKE  
Y ED65462EC8D5E4C5  
W: Some index files failed to download. They have been ignored, or old ones used inst  
ead.  
[kali@kali]~$
```



Hecho esto procedemos con la instalación de Freeradius con el siguiente comando: **sudo apt-get install freeradius**



Confirme la instalación presionando la tecla **Y**

Se requiere inicializar como **Super Usuario** dentro de kali con la contraseña **sistemas**

Nos dirigimos a la carpeta de Freeradius con el comando: **ls /etc/freeradius/3.0/**

Si la instalación se realizó correctamente tendrían que salir los siguientes archivos


```
GNU nano 2.9.3 /etc/freeradius/3.0/users
# RADIUS Client-Text-Password := "12345"
admin Client-Text-Password := "admin"
Reply-Message := "bienvenido"

#
# Configuration file for the radfiles module.
# Please see radfiles(5) manpage for more information.
#
# This file contains authentication security and configuration
# information for each user. Accounting requests are NOT processed
# through this file. Instead, see "accounting" in this directory.
#
# The first field is the user's name and can be up to
# 255 characters in length. This is followed (on the same line) with
# the list of authentication requirements for that user. This can
# include password, com server name, com server port number, protocol
# type (perhaps set by the "hints" file), and huntgroup name set by
# the "huntgroups" file).
#
# If you are not sure why a particular reply is being sent by the
# server, then run the server in debugging mode (radiusd -X), and
# you will see which entries in this file are matches.
#
# When an authentication request is received from the com server,
# these values are tested. Only the first match is used unless the
# "Fail-Through" variable is set to "yes".
#
# A special user named "DEFAULT" matches on all usernames.
# You can have several DEFAULT entries. All entries are processed
# in the order they appear in this file. The first entry that
# matches the login-request will stop processing unless you use
# the Fail-Through variable.
#
# Indented (with the tab character) lines following the first
# line indicate the configuration values to be passed back to the
# com server to allow the initiation of a user session.
# This can include things like the PPP configuration values
```

Una vez creados los usuarios guardamos y cerraremos el archivo.

A continuación, procederemos a configurar los clientes, estos son los equipos de red que tienen permiso para consultarnos por los usuarios.

Para ello ingresamos el comando: **nano /etc/freeradius/3.0/clients.conf**

```
[root@kali: /home/fernanda]
# nano /etc/freeradius/3.0/clients.conf
Completing file
freeradius/ freeradius/
```

Para comenzar con la configuración de este archivo nos vamos al final de este y escribimos los parámetros de nuestro cliente siguiendo el siguiente esquema.

Implementaremos los clientes que pueden realizar la autenticación, en este caso "0.0.0.0/0" nos indica que todas las ip pueden realizar la autenticación, si se quisiera solo en un segmento de red seria de la siguiente manera "192.168.100.0/0"

```
GNU nano 8.3 /etc/freeradius/3.0/clients.conf
# If there is no listener in a virtual server, SQL clients are added
# to the global list for that virtual server.
#
# If there is a listener, and the first listener does not have a
# "clients=..." configuration item, SQL clients are added to the
# global list.
#
# If there is a listener, and the first one does have a "clients=..."
# configuration item, SQL clients are added to that list. The clients
# [...] configured in that list are also added for that listener.
#
# The only issue is if you have multiple listeners in a virtual
# server, each with a different client list, then the SQL clients are
# added only to the first listener.

#clients_per_socket_clients {
#  client socket_client {
#    ipaddr = 192.0.2.4
#    secret = testing123
#  }
#}

client 192.168.28.159 {
  secret = sistemas
  shortname = router1
  nastype = cisco
}

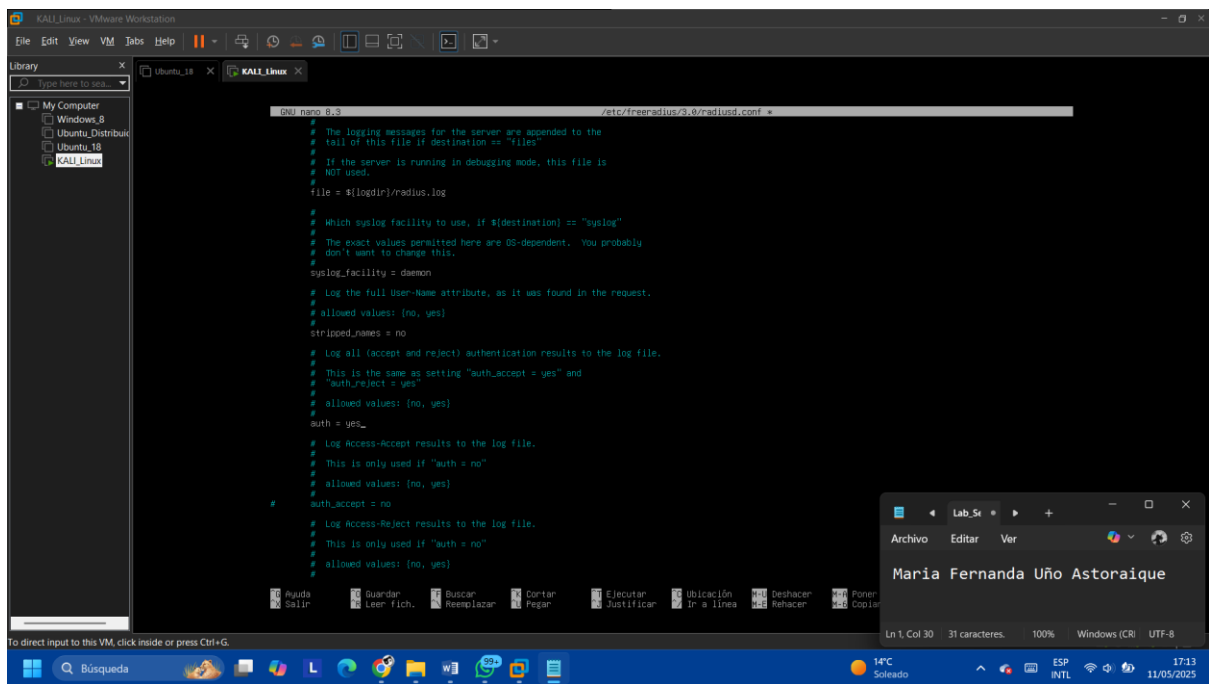
client 0.0.0.0/0 {
  ipaddr = 0.0.0.0/0
  secret = sistemas
  shortname = windows-client
}
```

Hasta este punto, Freeradius estaría configurado de forma local, sin embargo, para agregar una capa adicional de seguridad y poder rastrear quien consulta nuestro servicio, habilitaremos la generación de registros (logs).

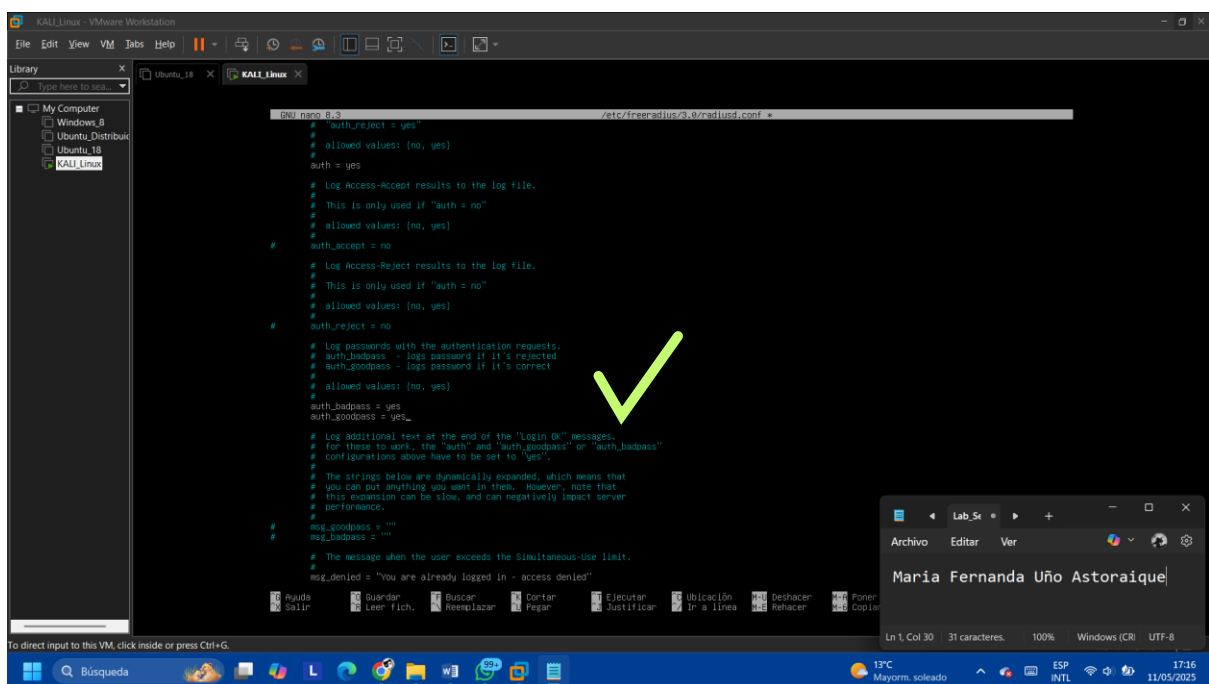
Para ello nos dirigimos al archivo `radiusd.conf` con el comando: **nano /etc/freeradius/3.0/radiusd.conf**

```
#
# allowed values: (no, yes)
#
auth = yes
#
# Log Access-Accept results to the log file.
# This is only used if "auth = no"
#
# allowed values: (no, yes)
#
auth_accept = no
#
# Log Access-Reject results to the log file.
# This is only used if "auth = no"
#
# allowed values: (no, yes)
#
auth_reject = no
#
# Log passwords with the authentication requests.
# auth_badpass - logs password if it's rejected
# auth_goodpass - logs password if it's correct
#
# allowed values: (no, yes)
#
auth_badpass = yes
auth_goodpass = yes
#
# Log additional text at the end of the "Login Ok" messages.
# For these to work, the "auth" and "auth_goodpass" or "auth_badpass"
# configurations above have to be set to "yes".
#
# The strings below are dynamically expanded, which means that
# you can put anything you want in them. However, note that
# this expansion can be slow, and can negatively impact server
# performance.
#
msg_goodpass = ""
msg_badpass = ""
#
# The message when the user exceeds the Simultaneous-Use limit.
msg_denied = "You are already logged in - access denied"
```

Una vez dentro del archivo, localizamos la línea correspondiente de la autenticación (auth) y modificamos el valor predeterminado “no” a “YES”



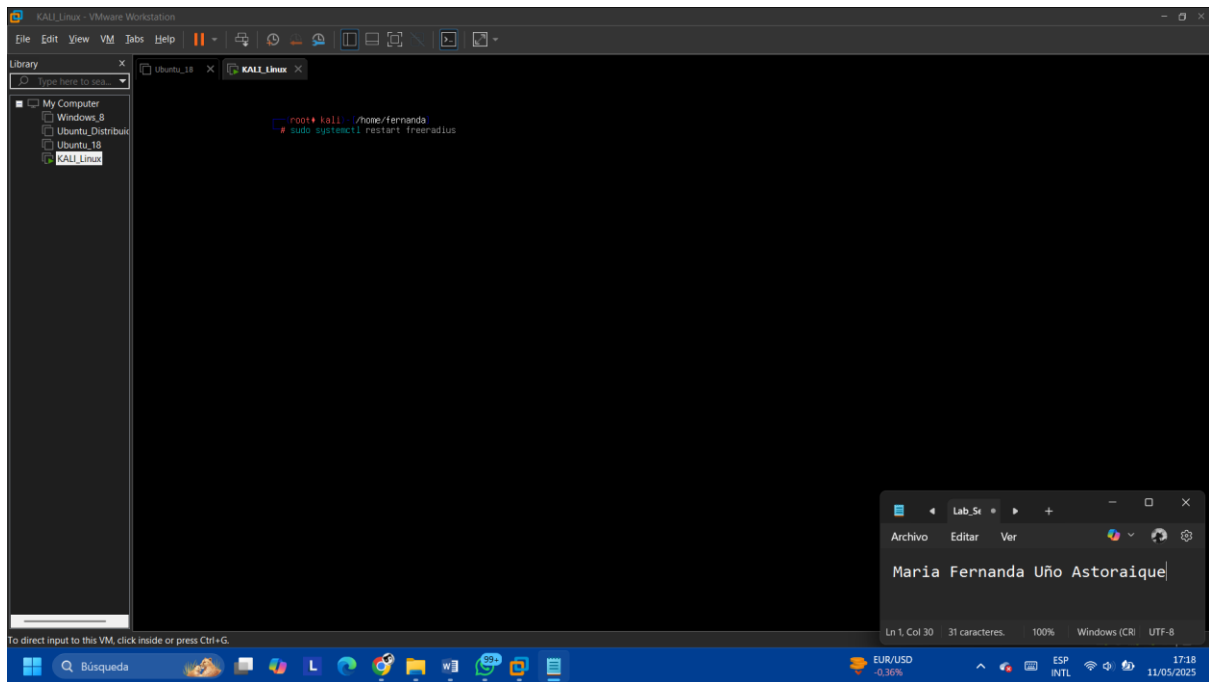
Realizaremos el mismo paso anterior para la generación de logs para eventos como **auth_badpass** y **auth_goodpass**



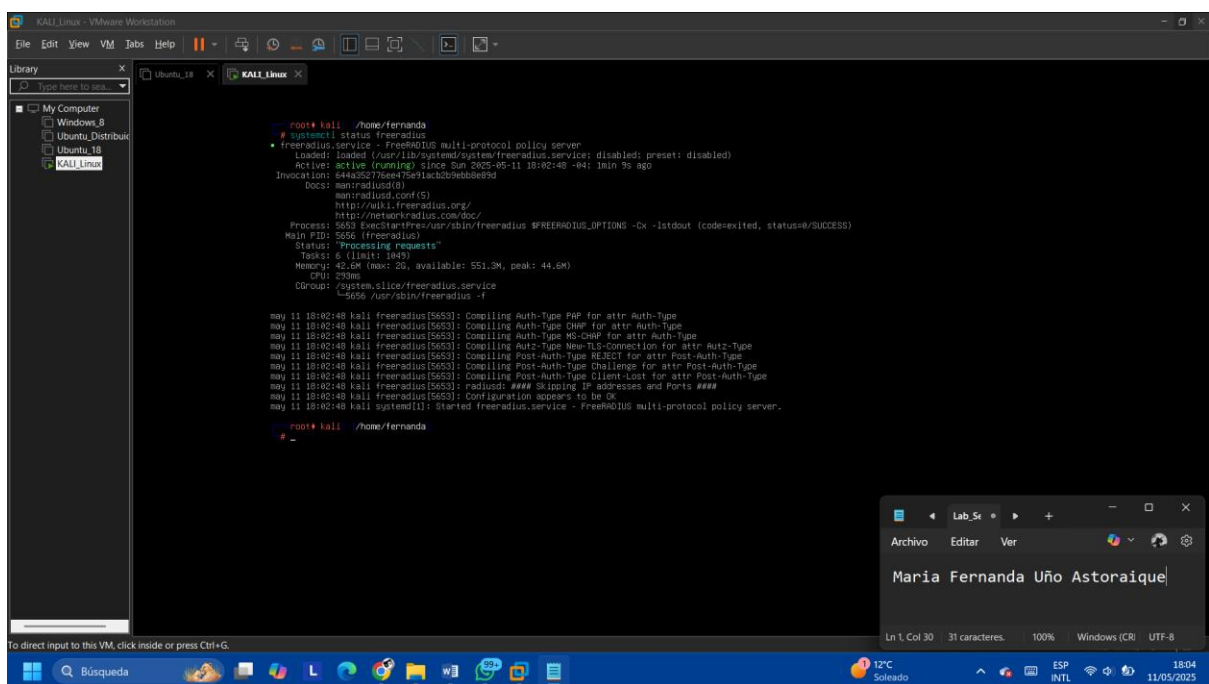
“auth_badpass = yes” Indicamos que se generen logs de cuando alguien intente autenticarse de manera incorrecta o falle en el proceso de inicio de sesión.

“auth_goodpass = yes” Indicamos que se generen logs para todos usuarios que inicien sesión correctamente.

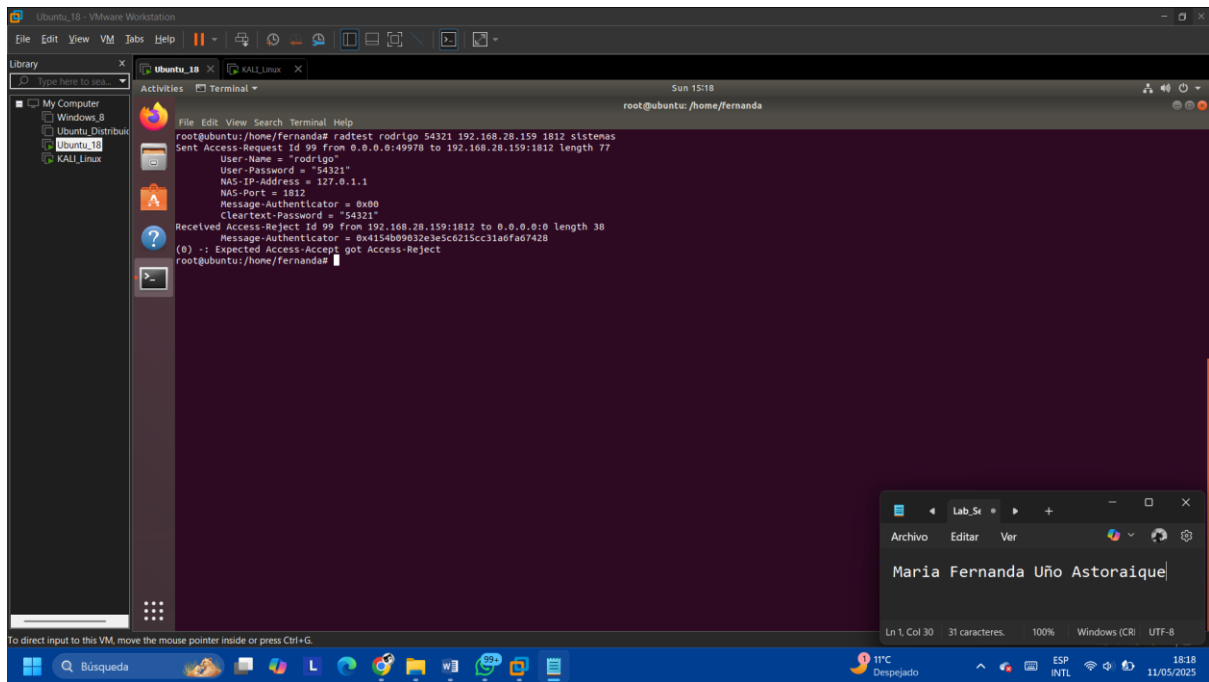
Procedemos a reiniciar el servicio de Freeradius para aplicar todos los cambios realizados con el siguiente comando: **sudo systemctl restart freeradius**



Una vez hecho esto, verificamos que el servicio este activo con el comando `sudo systemctl status freeradius`

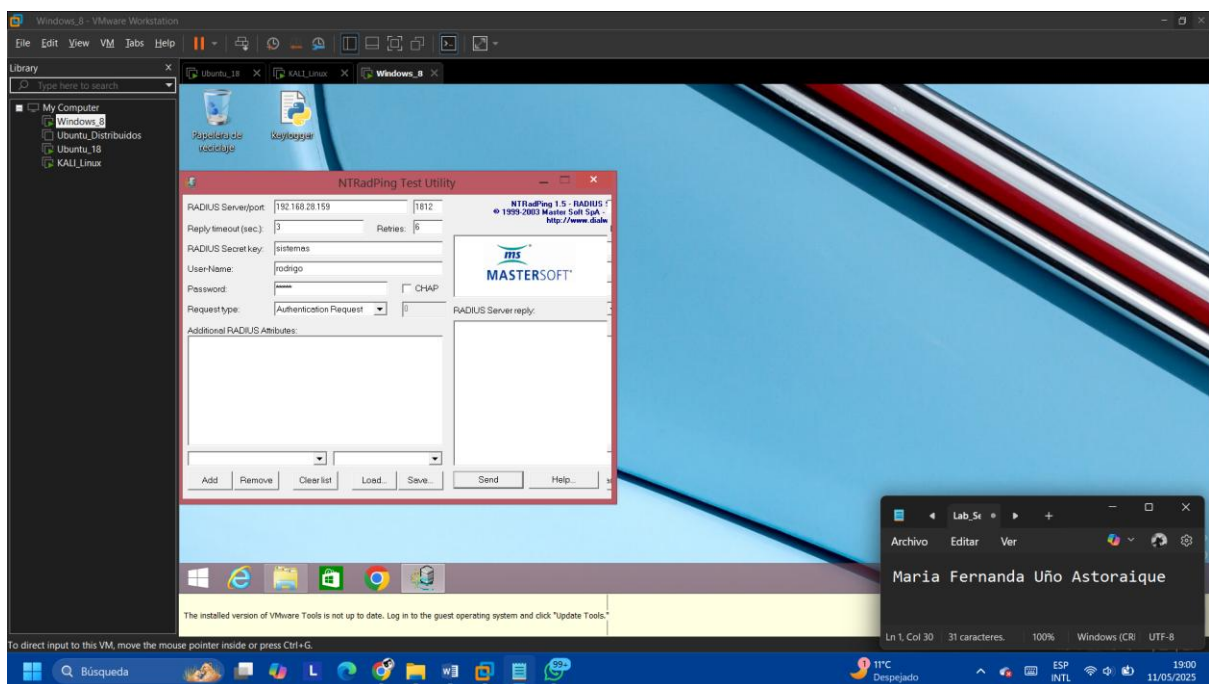


Ya está listo para usarse para ello nos vamos a otro equipo en este caso utilizamos Ubuntu eh instamos la siguiente herramienta

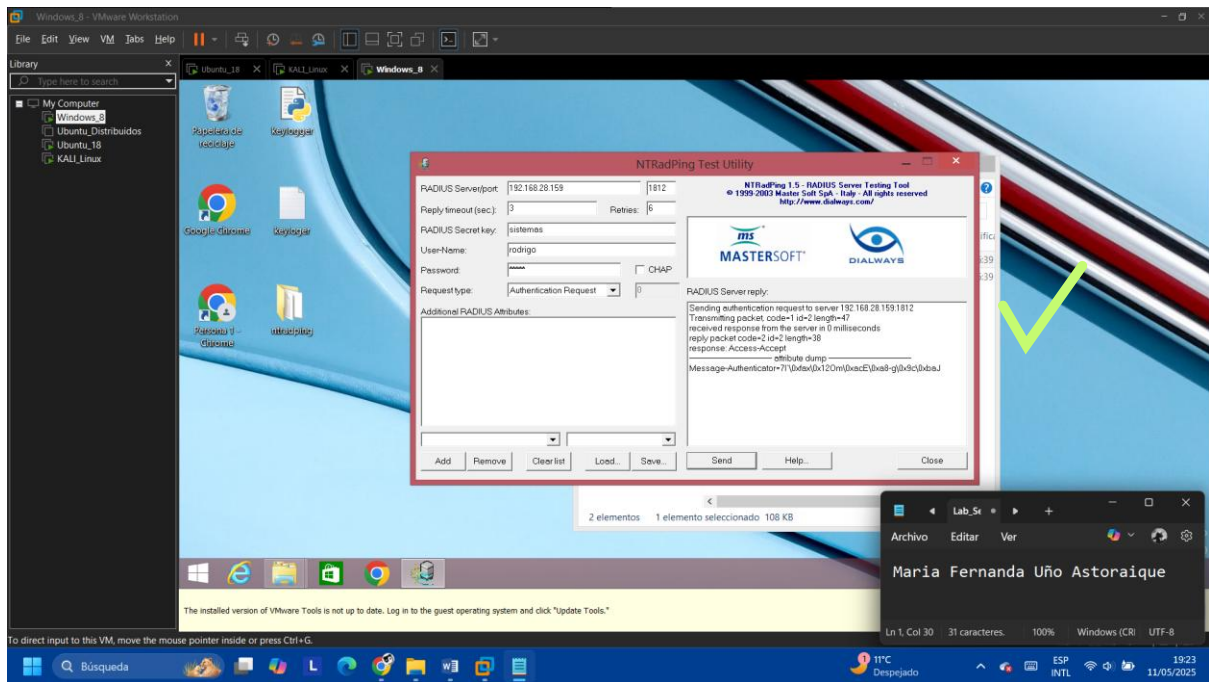


Nos dirigiremos a Windows 8 y abriremos la aplicación de NTRadPing, esta se encuentra dentro de la carpeta con el mismo nombre de la aplicación.

Dentro de la aplicación pondremos la ip del servidor radius, puerto lógico, contraseña compartida, usuario, contraseña de usuario, una vez llenado todas las casillas presiona **Send**.



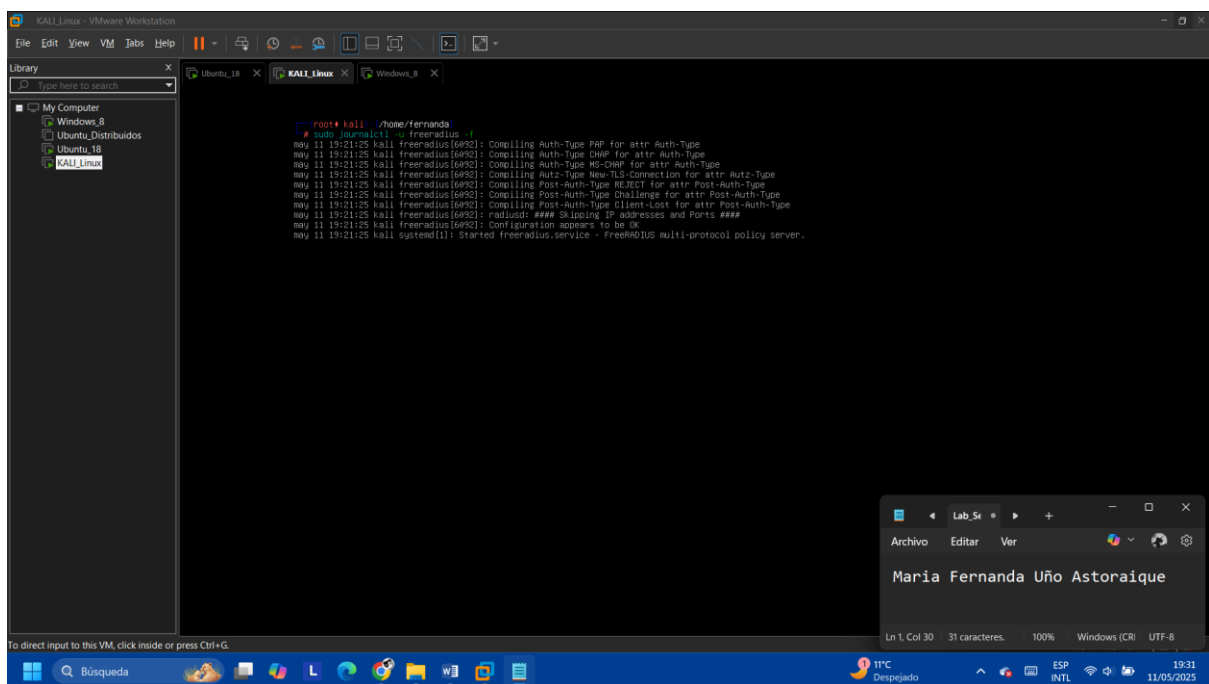
Y este nos mostrara el siguiente mensaje



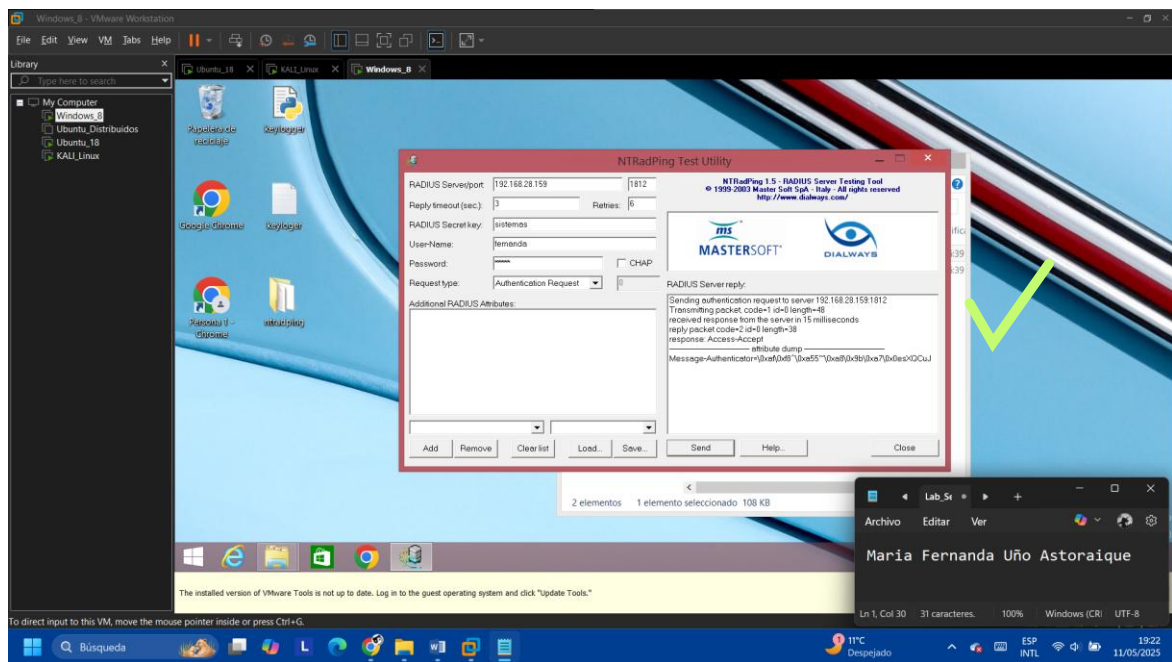
Evaluación

1.- Con que comando se puede ver los logs en tiempo real en el servidor radius.

R.- `sudo tail -f /var/log/freeradius/radius.log`, otra opción: `sudo journalctl -u freeradius -f`



2.- Crear un nuevo usuario e logearse tanto desde Linux como desde Windows e indique que datos puede observar en estos logs. Al crear el usuario ponga su nombre como usuario y su apellido como contraseña.



Se puede observar:

Fecha y hora del intento de autenticación.

Usuario que intenta autenticarse.

Resultado de la autenticación (aceptado o rechazado).

