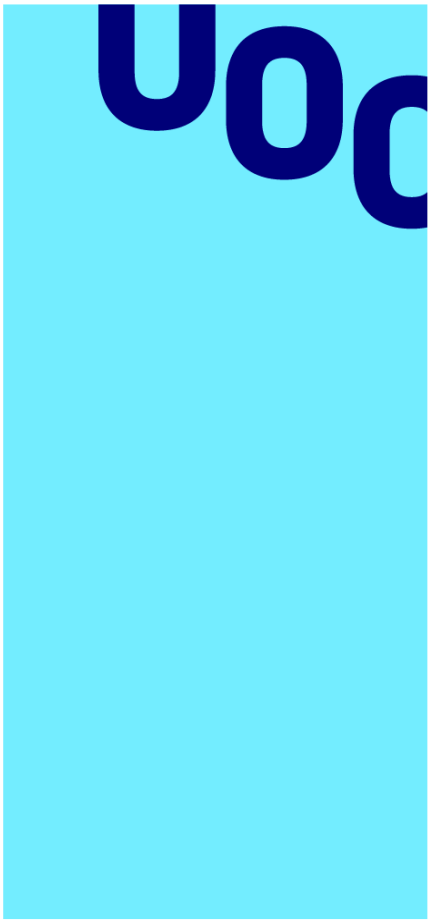
Implementación y administración de redes seguras basadas en perfiles de acceso



Universitat Oberta de Catalunya

Daniel Pérez Torres

Grado de Ingeniería Informática
TFG - Redes de computadores

Nombre Tutor/a de TF
Amadeu Albós Raya
Profesor/a responsable de la asignatura
Joan Manuel Marquès Puig

09/01/2024

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación y administración de redes seguras basadas en perfiles de acceso*
Nombre del autor:	*Daniel Pérez Torres*
Nombre del consultor/a:	*Amadeu Albós Raya*
Nombre del PRA:	*Joan Manuel Marquès Puig*
Fecha de entrega (mm/aaaa):	*01/2024*
Titulación o programa:	Grado de Ingeniería Informática
Área del Trabajo Final:	*TFG – Redes de computadores*
Idioma del trabajo:	*Castellano*
Palabras clave	*Redes, seguridad, NAC, 802.1x*

Resumen del Trabajo

Actualmente, los entornos de red corporativos se enfrentan a diferentes necesidades de conectividad que pueden derivar en brechas de seguridad. Dos ejemplos de ello podrían ser permitir a los trabajadores utilizar sus propios dispositivos (*BYOD*) o la movilidad de los empleados dentro de la propia organización. Estas y otras situaciones implican constantes gestiones administrativas a nivel de red, donde una mala decisión o configuración puede dar lugar a una amenaza que pondría en peligro su activo más valioso, la información.

La finalidad de este trabajo consiste en evitar esta situación mediante la implementación de una infraestructura que sea capaz de tomar decisiones, de manera automática, sobre los permisos y políticas de seguridad que se deben aplicar sobre cada usuario o equipo que intente conectarse a la red. Gracias a ello se logra un control de acceso basado en perfiles totalmente automatizado, donde cada uno de ellos dispondrá únicamente de los permisos que le otorgue el sistema.

Para lograrlo se definirá una estrategia de implementación en capa 2 del modelo OSI basada en el protocolo 802.1x, llevando a cabo su estudio en profundidad para así obtener, en primer lugar, una amplia visión sobre el mismo, y, en segundo lugar, los conocimientos necesarios para implementarlo sobre una infraestructura corporativa con garantías de éxito.

A su vez, este sistema de control de acceso estará apoyado por una herramienta de gestión centralizada que será desarrollada para tal propósito,

la cual permitirá a los/as administradores/as tener visibilidad sobre todo lo relacionado con el mismo, incluyendo tareas de configuración, monitorización y *troubleshooting*.

Abstract

Nowadays, corporate network environments face different connectivity needs that can lead to security breaches. Two examples of this could be allowing workers to use their own devices (BYOD) or employee mobility within the organization itself. These and other situations involve constant administrative management at the network level, where a bad decision or configuration can lead to a threat that could endanger your most valuable asset, the information.

The purpose of this work is to avoid this situation by implementing an infrastructure that is capable of automatically making decisions about the permissions and security policies that should be applied to each user or computer that tries to connect to the network. Thanks to this, an access control based on fully automated profiles is achieved, where each of them will only have the permissions granted by the system.

To achieve this, an implementation strategy will be defined in layer 2 of the OSI model based on the 802.1x protocol, carrying out an in-depth study to obtain, firstly, a broad vision of it, and secondly, the necessary knowledge to implement it on a corporate infrastructure with guarantees of success.

In turn, this access control system will be supported by a centralized management tool that will be developed for this purpose, which will allow administrators to have visibility over everything related to it, including configuration, monitoring and troubleshooting tasks.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

A medida que transcurren los años, las necesidades y características de las redes corporativas varían notablemente, debiendo adaptarse en todo momento a los objetivos del mercado y a las nuevas exigencias de clientes/as y empleados/as. Este hecho ha derivado en que en la actualidad surjan escenarios con diferentes modelos de conectividad dentro de una misma red corporativa, donde, por ejemplo, cada vez resulta más habitual proveer conectividad wifi a los clientes, facilitar la movilidad de los trabajadores dentro de la propia compañía o incluso permitir a los empleados conectar sus propios dispositivos a la red corporativa y trabajar con ellos, modelo este que se conoce como BYOD (*Bring your own device*).

Todo ello agrega enormes beneficios y facilidades, sin embargo, también implica importantes riesgos de seguridad si no se aplican las medidas adecuadas. En este sentido, la seguridad suele ser tratada en las capas de red y de enlace de datos del modelo OSI mediante la aplicación de políticas en firewalls y la gestión manual de segmentos de red, hecho que genera una importante carga administrativa para los/as administradores/as, y que a su vez se traduce en descontrol sobre los activos y posibles accesos no autorizados o con permisos no deseados.

Ante esta problemática, la medida más efectiva pasa por implementar algún sistema capaz de gestionar automáticamente las conexiones de red y que, además, aplique determinados permisos de acceso dependiendo del usuario o dispositivo que se conecte. Sin embargo, actualmente no es habitual que las organizaciones se decanten por este tipo de soluciones, ya que supone un importante desafío debido a su alta complejidad de implementación, gestión y coste. [1] [2]

Este proyecto nace con el objetivo de ofrecer una solución a la problemática recién expuesta, profundizando sobre los conocimientos, estrategias y configuraciones necesarias para implementar una infraestructura de red automatizada basada en perfiles de acceso, haciendo uso para ello de herramientas *open source.* A su vez, la solución estará acompañada por un software de gestión centralizada desarrollado para tal propósito, con el fin de facilitar la administración y monitorización del control de acceso a la red de la organización.

1.2. Objetivos del Trabajo

Este proyecto ha sido desarrollado con la finalidad de obtener el conocimiento necesario para diseñar e implementar una infraestructura de red basada en

perfiles de acceso, automatizada y segura. Con ello, el trabajo presenta los siguientes objetivos personales:

- Profundizar sobre los protocolos de autenticación de red 802.1x y MAB y su importancia en una infraestructura de red basada en NAC.
- Aprender a identificar los requisitos que una red de este tipo debe cumplir, así como ser capaz de seleccionar los componentes más adecuados para su implementación en base a las necesidades específicas de cada entorno.
- Ser capaz de diseñar e implementar una red robusta y segura basada en perfiles de acceso.
- Lograr dicha implementación al menor coste posible, utilizando para ello soluciones *open source*.
- Ser capaz de desarrollar una aplicación de gestión personalizada para facilitar la administración del sistema.
- Lograr que dicha aplicación resulte intuitiva y adaptable a cualquier entorno NAC.
- Conocer nuevas librerías y métodos de programación en Phyton, que puedan ser aplicables tanto al entorno NAC que se va a implementar como a otros ámbitos de redes y comunicaciones.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

En cuanto a las tres dimensiones asociadas a la competencia de compromiso ético y global (CCEG), este proyecto, debido a su naturaleza y resultado final, presenta un impacto positivo al alinearse con los siguientes ODS:

I. Sostenibilidad: El resultado de este trabajo es aplicable sobre cualquier infraestructura de red sin importar el tipo de organización ni propósito, aportando beneficios sobre su seguridad y gestión. Debido a ello está estrechamente relacionado de manera positiva con el "*ODS 9 - Industry, innovation and infrastructure".* Sin embargo, sobre el resto de objetivos de sostenibilidad (*"ODS 7 – Affordable and clean energy", ODS 11 –Sustainable cities and communities", "ODS 12 - Responsible consumption and production", "ODS 13 - Climate action", "ODS 14 - Life below wáter" y "ODS 15 - Life on land")*, su impacto es neutro, ya que su ámbito de aplicación y objetivos no influyen en los mismos.

II. Comportamiento ético y responsabilidad social (RS): Otro de los beneficios que aporta la implementación propuesta es el aumento de la calidad laboral de los/as administradores/as de red, así como la seguridad de la información de las organizaciones, la cual está estrechamente ligada al crecimiento económico de la misma y, con ello, a un crecimiento económico global. Es por ambos motivos que el proyecto también está alineado positivamente con los "*ODS 8 - Decent work and economic growth",* y "*ODS 16 – Peace, justice and strong instituons"*, este último en lo referente a la creación de instituciones sólidas. En cuanto al resto de objetivos de esta dimensión ("*ODS 1 – No poverty", "ODS2 – Cero Hunger"* y *"ODS 6 – Clean wáter and sanitation")* el impacto de este proyecto es neutro, ya que el ámbito de aplicación y resultado

no influye en dichos objetivos de comportamiento ético y responsabilidad social.

III. Diversidad y derechos humanos: Por último, este proyecto está dirigido y puede beneficiarse del mismo cualquier persona sin importar su género, raza, situación social, orientación sexual, creencias o cualquier otro aspecto que pueda suponer exclusión o discriminación. Por tanto, se puede vincular positivamente con los "*ODS 5 - Gender equality*" y "*ODS 10 – Reduced inequalities*".

1.4. Enfoque y método seguido

Debido a las características del producto que se desea obtener, y teniendo en cuenta que cada concepto o implementación llevada a cabo resulta necesaria para poder continuar con la siguiente, se ha considerado como opción ideal para el desarrollo del proyecto aplicar la metodología basada en **cascada**. Además, este modelo facilita la compresión de los conceptos y ayuda en gran medida a generar una documentación ordenada, completa y útil. [3]

Es por ello que cada fase del proyecto será desarrollada de manera independiente, debiendo ser finalizada para poder continuar con la siguiente. En este sentido, las fases de las que constará este trabajo son las que se listan a continuación, donde cada una complementa y genera valor añadido sobre la anterior:

- **Fase 0 - Plan de trabajo**: Donde se definen aspectos relevantes del proyecto, como la justificación del trabajo, objetivos, método para llevarlo a cabo o plazos de ejecución de cada una de las tareas de las que se compone.

- **Fase 1 - Estudio y análisis de la solución:** En la cual se recopilará toda la información necesaria para poder desarrollar el proyecto con garantías de éxito, se seleccionarán los métodos ideales a aplicar y se establecerán las propiedades que deberán cumplir los productos obtenidos.

- **Fase 2 – Diseño de los productos:** Donde, partiendo del estudio anterior, se definirán los requisitos, componentes necesarios y tecnologías ideales para lograr el objetivo. Con ello, se diseñarán los productos que se obtendrán al finalizar el proyecto.

- **Fase 3 - Implementación:** Como su nombre indica, en esta fase se llevarán a la práctica e implementarán los productos diseñados en la fase anterior.

- **Fase 4 - Pruebas**: En esta fase verificaremos el resultado de los productos mediante su aplicación en un entorno de laboratorio.

- **Fase 5 - Conclusiones y trabajos futuros**: Por último, en base a los resultados anteriores, se obtendrán conclusiones y se elaborarán una serie de propuestas para futuras implementaciones.

- **Fase 6 - Entrega del producto**: El proyecto se da por finalizado con la presentación y entrega de los productos desarrollados.

1.5. Planificación del Trabajo

Una vez decidido el enfoque y metodología a aplicar, la planificación del trabajo marcará las tareas e hitos a lograr durante el tiempo de ejecución del proyecto. Con las fases ya definidas, dichas tareas y el tiempo asignado a cada una de ellas serán las siguientes.

Nombre	Inicio	Fin	Hito
Fase 0: Plan de trabajo	5/10/23	9/10/23	
Definición	5/10/23	9/10/23	
Entrega del plan de trabajo	10/10/23	10/10/23	*
Fase 1: Estudio y análisis de la solución	11/10/23	6/11/23	
Estudio e informe sobre la necesidad	11/10/23	14/10/23	
Estudio de la seguridad en capa 2 del modelo OSI	15/10/23	21/10/23	
Análisis de métodos de autenticación en capa 2	22/10/23	27/10/23	
Análisis de NAC	28/10/23	1/11/23	
Estudio de ataques sobre NAC y cómo prevenirlos	2/11/23	6/11/23	
Entrega del primer seguimiento	7/11/23	7/11/23	*
Fase 2: Diseño de los productos	8/11/23	22/11/23	
Diseño de infraestructura de red basada en perfiles de acceso	8/11/23	17/11/23	
Diseño del software de gestión	18/11/23	22/11/23	
Fase 3: Implementación	23/11/23	19/12/23	
Implementación de infraestructura de red basada en perfiles de acceso	23/11/23	4/12/23	
Entrega del segundo seguimiento	5/12/23	5/12/23	*
Desarrollo del software de gestión	5/12/23	19/12/23	
Fase 4: Pruebas	20/12/23	3/1/24	
Definición de pruebas a realizar	20/12/23	24/12/23	
Ejecución de pruebas	25/12/23	3/1/24	
Fase 5: Conclusiones y trabajos futuros	4/1/24	8/1/24	
Conclusiones y trabajos futuros	4/1/24	8/1/24	
Fase 6: Entrega del producto	9/1/24	23/1/24	
Entrega de la memoria	9/1/24	9/1/24	*
Preparación de la defensa del trabajo	9/1/24	23/1/24	
Defensa del trabajo final	24/1/24	24/1/24	*

Figura 1: Planificación del trabajo

Lo cual genera el siguiente diagrama de Gantt:

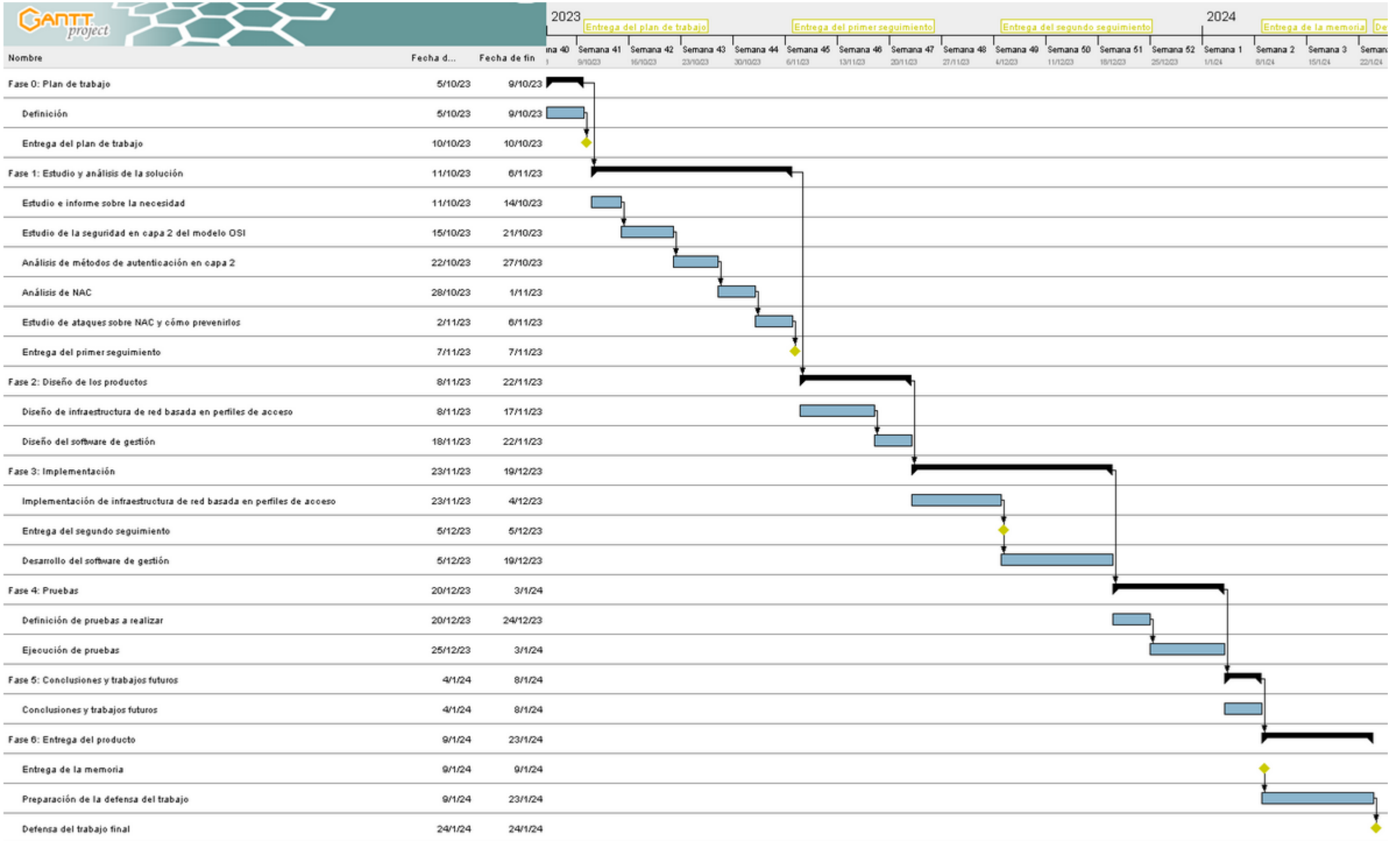


Figura 2: Diagrama de Gantt
Realizado con GanttProject

Asimismo, se puede observar que se han establecido los siguientes hitos:

Hito	Fecha de entrega
Entrega del plan de trabajo	10/10/2023
Entrega del primer seguimiento	07/11/23
Entrega del segundo seguimiento	05/12/23
Entrega de la memoria	09/01/24
Defensa del trabajo final	24/01/24

Por otro lado, para la realización del proyecto serán necesarios los siguientes recursos:

- Switch con soporte 802.1x y MAB.
- Punto de acceso con autenticación WPA2/WPA3 Enterprise.
- Servidor Radius y NAC – Se utilizará el software "*PacketFence*" (*open source*).
- Firewall – Se utilizará la solución "*PfSense*" (*open source*).
- Directorio Activo – Se utilizará un servidor "*Windows 2019 Server".*
- Conectividad con Internet.
- Entorno de desarrollo de aplicaciones:
  - Lenguaje: *HTML* y *Phyton* (versión 3.12.0)
  - Entorno de desarrollo: *Pycharm Community Edition (open source)*.
  - Servidor Web: *Flask* (*open source)*
  - Librerías *Phyton* para interactuar con elementos de red*: Paramiko, ldap3, mysql-connector…*
- Diferentes equipos que actuarán como dispositivos finales.

1.6. Breve sumario de productos obtenidos

Como resultado de este proyecto se generarán los siguientes entregables:

- El diseño e implementación de una infraestructura de red con control de acceso automatizado basado en perfiles de usuarios y/o dispositivos.
- El diseño y desarrollo de una herramienta de gestión que permita administrar y monitorizar de manera centralizada todos los elementos del control de acceso y los usuarios de la infraestructura.

1.7. Breve descripción de los otros capítulos de la memoria

Los capítulos que darán continuidad al actual, y que guiarán al proyecto hasta su finalización serán los siguientes:

- **Capítulo 2 - Estudio y análisis de la solución**: Como su nombre indica, este capítulo estará dedicado a obtener todos los conocimientos necesarios para poder ejecutar el proyecto. En primer lugar, se expondrá un enfoque teórico sobre la necesidad de su implementación, beneficios y aspectos relevantes a tener en cuenta, y, en segundo lugar, el análisis técnico de cada uno de los

protocolos y elementos necesarios que requiere la solución. Se finalizará el estudio analizando los posibles ataques sobre el sistema de control de acceso y la manera más adecuada para prevenirlos.

- **Capítulo 3 - Diseño**: En el tercer capítulo se llevará a cabo el diseño de los productos que se desean obtener. Se comenzará por la infraestructura de red basada en perfiles de acceso, ya que su resultado será necesario para posteriormente diseñar el software de gestión. En ambos casos se realizará el análisis de requisitos, componentes y tecnologías necesarias, para, con dichos datos, diseñar adecuadamente las soluciones.

- **Capítulo 4 - Implementación**: Diseñados los productos, se procederá a su implementación sobre un entorno de laboratorio. Nuevamente, en primer lugar se llevará a cabo la infraestructura de red basada en perfiles de acceso, donde se configurarán los dispositivos y se aplicarán las políticas necesarias. Tras ello, se desarrollará e implementará el software de gestión, el cual permitirá realizar labores de administración y monitorización sobre dicha infraestructura.

- **Capítulo 5 - Pruebas**: Con los productos ya operativos, se realizarán diferentes pruebas basadas en un caso práctico, con el objetivo de verificar que se cumple con los requisitos exigidos.

- **Capítulo 6 - Conclusiones y trabajos futuros**: Por último, se expondrán las conclusiones obtenidas tras la realización del trabajo y se propondrán futuras implementaciones, tanto de mejora del actual sistema como de nuevo desarrollo.

2. Estudio y análisis de la solución

2.1. Estudio sobre la necesidad del proyecto

Como se ha mencionado anteriormente, las redes corporativas cada vez resultan más complejas debido a la necesidad de adaptarse a las características y objetivos de negocio de cada organización. Por ejemplo, el simple hecho de prestar algún tipo de servicio a través de Internet supone una exposición permanente a un acceso potencialmente inseguro como lo es la red pública. Si a esto le sumamos multitud de necesidades más, como las ya mencionadas wifis para clientes, movilidad de empleados, o BYOD, entre otros, el riesgo de sufrir una brecha de seguridad crece exponencialmente.

Con ello, y teniendo en cuenta que el activo más valioso de cualquier organización es su información, y que la misma puede estar constantemente comprometida, el objetivo de este proyecto se centró en todo momento en proporcionar un método efectivo de securización y administración de la red. En base a dicha premisa, se realizó un estudio para determinar qué elementos eran más susceptibles a ataques y la manera más adecuada para protegerlos, así como un análisis de las medidas de seguridad más habituales adoptadas por las organizaciones, con el fin de detectar carencias y poder ofrecer alguna solución al respecto, evidenciando así la necesidad de este trabajo.

El primer paso del estudio consistió en **analizar los tipos de ataque** más comunes y eficaces utilizados por los ciberdelincuentes. La necesidad de este dato no es otra que obtener una visión adecuada sobre qué elementos son más vulnerables, y, por tanto, propensos a sufrir un ataque que pueda poner en riesgo la información corporativa. Con ello, se pretende identificar aquellos elementos de la red que requerirán políticas de protección robustas. El informe más reciente emitido por ENISA (*European Union Agency for Cybersecurity*), evidencia el siguiente resultado. [6, *página 9*]

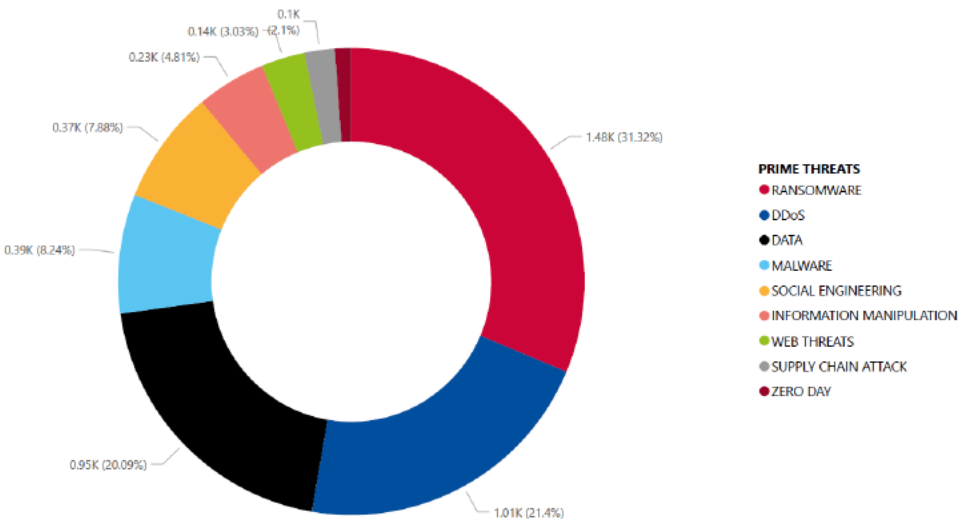


Figura 3: Estadísticas de ciberataques por tipo de amenaza (2023)

Analizando el modus operandi de cada una de estas amenazas, se puede llegar a la conclusión de que la gran mayoría de ellas requiere la intervención del usuario para poder materializarse, ya sea mediante la ejecución de un fichero o el acceso a enlaces indebidos, entre muchos otros [7]. Por tanto, una medida de seguridad eficiente podría ser aquella centrada en los usuarios de la red, tanto a nivel de concienciación en materia de ciberseguridad, como a nivel técnico mediante la aplicación de los sistemas adecuados para controlar el acceso y permisos de los dispositivos que utilizan.

Como proyecto centrado en seguridad de la red, me parece más adecuado enfocar el objetivo en controlar el acceso de los dispositivos, ya que la concienciación de los usuarios se basa en la formación, y, por tanto, deberá ser la organización quien se encargue de ello. Es por este motivo que, en primera instancia, el trabajo se centrará en controlar el acceso, permisos y vulnerabilidades de los dispositivos de los usuarios, con el objetivo prioritario de proteger los activos de la red, especialmente la información.

El segundo paso del estudio se enfocó en **identificar métodos y soluciones** eficientes para dar solución a la necesidad recién expuesta. En este sentido, existen multitud de opciones que podrían implementarse para este propósito, donde las más comunes podrían ser los firewalls, IPS, WAF, antivirus o EDR, entre otros [8]. Sin embargo, he descartado realizar un proyecto sobre estas soluciones por varios motivos; primero, porque algunas de ellas operan en capa 7, cuando el enfoque de este trabajo es a nivel de capas 2 y/o 3 del modelo OSI, segundo, porque otras solo se limitan al análisis de tráfico en capa 3, mientras que para lograr el objetivo también haría falta comprobar otros aspectos, como las características de los dispositivos, usuarios, etc, y tercero, porque para todas estas soluciones existe una amplísima documentación, por lo que se considera que esta necesidad está lo suficientemente cubierta.

Continuando con la búsqueda de una solución lo más adecuada y eficiente posible, una opción que sí cubriría todas las necesidades sería un **sistema de control de acceso basado en perfiles** (también denominado **NAC**), gracias al cual, cada usuario o dispositivo sería analizado antes de permitir su conexión a la red, logrando así evitar y/o aislar el acceso de aquellos que puedan suponer una amenaza. Basar el proyecto en este sistema se considera adecuado y necesario por los siguientes motivos:

- Aunque en una implementación de NAC intervengan diferentes elementos, el control de acceso se inicia en capa 2 del modelo OSI, por lo que encaja en un proyecto dedicado a redes de computadores.
- Con NAC se logran los objetivos indicados anteriormente, es decir, controlar el acceso a la red de los dispositivos de los usuarios, evitando aquellos que puedan significar una amenaza.
- La mayoría de soluciones NAC disponibles son propietarias que requieren una elevada inversión, suponiendo este un impedimento que evita su puesta en marcha. Debido a ello, se ha considerado necesario aportar una solución basada en *open source,* de la cual pueda beneficiarse cualquier persona u organización sin coste alguno (en lo que a *software* se refiere). [1] [2]

- La documentación existente sobre este tipo de implementaciones se podría dividir en dos tipos; por un lado, aquella realizada por los propios fabricantes, bastante completa pero que se centran en un determinado producto que además implica coste, y, por otro lado, aquella desarrollada por profesionales independientes o estudiantes, en cuyo caso muchas se basan en soluciones *open source,* pero centradas en un único elemento de NAC. Por ejemplo, en *packetfence* [9] o desarrollo de herramientas [10]. Es por ello que se considera necesario elaborar un proyecto que cubra esta necesidad abordando todos los elementos que este tipo de sistemas debe incorporar.
- Por último, la administración de NAC puede resultar compleja, por lo que también se ha detectado la necesidad de desarrollar una herramienta de gestión centralizada desde la cual se pueda administrar y monitorizar todos los elementos intervinientes en este tipo de sistemas.

Teniendo en cuenta todo ello, el primer paso consistirá en conocer la solución, para posteriormente diseñarla e implementarla.

2.2. Sistemas de control de acceso a la red (NAC): Conceptos básicos

De manera muy resumida, un sistema de control de acceso a la red podría definirse como un método de seguridad que permite **analizar** a los usuarios y/o dispositivos que traten de conectarse a una determinada infraestructura, para, en base a dicho análisis, **decidir**, de manera automatizada, qué políticas y medidas se aplicarán sobre cada uno de ellos con la finalidad de mantener la red lo más protegida posible.

Gracias a ello se mantiene un control exhaustivo sobre todos los accesos y elementos presentes en la red, y, lo que es más importante, se logra definir un nivel de seguridad adecuado y adaptado a las necesidades de conectividad de cada uno de ellos. Tanto es así, que estos sistemas pueden ser considerados como la primera línea de defensa en cuanto a ataques y brechas de seguridad provocadas por dispositivos internos se refiere. [4]

Dicha afirmación se evidencia gracias, por un lado, a que el análisis se inicia directamente en capa 2 del modelo OSI (primera línea), y, por otro lado, gracias a la cantidad de factores que pueden ser analizados para prevenir la conexión de dispositivos vulnerables que puedan poner en riesgo la seguridad de la red (ataques y brechas de seguridad).

En este aspecto, bien es cierto que los sistemas de control de acceso **no** tienen la capacidad de eliminar amenazas de seguridad de los dispositivos en cuestión, pero sí de prevenir la conexión de estos a la red. Para lograrlo, las políticas de acceso centran su operación en analizar todos aquellos aspectos que el/la administrador/a de red considere necesarios, como podrían ser las actualizaciones del antivirus o el software instalado, entre muchas otras.

Por tanto, la primera acción que llevan a cabo estos sistemas consiste en el ya mencionado análisis de dispositivos o usuarios mediante las políticas

establecidas. Tras ello, y en base al resultado obtenido, la segunda acción pasa por asignar determinados accesos a la red, definidos en lo que se conoce como **perfil de acceso**, el cual establecerá el nivel de permisos otorgados.

Al igual que ocurre con las políticas de acceso, los perfiles también disponen de multitud de opciones que el/la administrador/a deberá definir. Entre las acciones más comunes se encuentra la de denegar el acceso a la red o asignarle un determinado segmento. La primera de ellas resulta evidente, mientras que la segunda opción consiste en asignarle una determinada VLAN, es decir, el segmento de red al que pertenecerá el equipo.

Esta segunda opción resulta especialmente interesante sobre entornos que requieren diferentes tipos de conectividad. Por ejemplo, una red corporativa en la que cada departamento dispone de su propio segmento de red. En este caso se podría optar por crear una VLAN para cada uno de ellos, haciendo uso de NAC para identificar a cada tipo de usuario y asignarles automáticamente la VLAN a la que deben pertenecer. Al tratarse de un caso muy común, esta casuística se tendrá en cuenta a la hora de diseñar los productos de este trabajo.

Como se puede deducir, NAC ejerce un control exhaustivo sobre cualquier elemento que se conecte a la red, lo que también se traduce en una enorme visibilidad, permitiendo conocer en todo momento y en tiempo real los dispositivos conectados y las características de estos, facilitando así las labores de monitorización, administración y *troubleshooting*. [5]

Con todo ello, parecen numerosos los beneficios que un sistema de control de acceso puede ofrecer sobre una red corporativa. Sin embargo, también hay que tener en cuenta sus desventajas, para así poder valorar los posibles riesgos a los que nos enfrentamos y la manera más apropiada de implementarlo sobre una determinada organización en base a su infraestructura y necesidades.

2.2.1. Características, beneficios y desventajas de NAC

En general, las características básicas que cualquier solución NAC debe ofrecer son las siguientes [11][12]:

- **Identificación de dispositivos**: Es decir, obtener ciertos datos de cada equipo o usuario que intente conectarse a la red.
- **Diferentes métodos de autenticación**, que podrán basarse en la dirección MAC del dispositivo o mediante validación de usuario.
- Creación y asignación de **políticas de acceso y perfiles de seguridad**, gracias a los cuales se podrá definir qué dispositivos tendrán acceso a la red (los que cumplan determinados requisitos) y qué accesos se le otorgarán.

A su vez, dichas características y modo de operar dan como resultado los siguientes beneficios [11][12]:

- **Seguridad**: Gracias al control de dispositivos, usuarios y permisos.

- **Rendimiento de la red**: Únicamente podrán acceder los dispositivos o usuarios autenticados, y, además, con permisos bien definidos. Este hecho limita el tráfico de red, optimizando con ello el rendimiento.
- **Visibilidad y Monitorización**: NAC ofrece visibilidad e información de todos los dispositivos conectados a la red, facilitando así la resolución de incidencias.
- **Automatización**: Una vez configurado el sistema, todo el procedimiento se lleva a cabo de manera automática, hecho que limita la posibilidad de cometer errores de configuración.
- **Cumplimiento de normativas**: NAC se convierte en una herramienta ideal para cumplir con determinados reglamentos, como podría ser el RGPD.

Sin embargo, esta solución también presenta aspectos en contra que deben ser considerados, como podrían ser: [1][13]

- **Coste**: Aun optando por soluciones de software *open source*, el hardware (switchs, puntos de acceso, etc.) debe ser compatible con NAC, permitiendo la autenticación de dispositivos, por lo que, si la electrónica actual no permite esta opción, el coste de su implementación puede resultar muy elevado.
- **Compatibilidad**: NAC requiere diferentes elementos, por lo que hay que asegurarse de que todos ellos son compatibles y pueden operar entre sí.
- **Complejidad**: Son sistemas que pueden resultar complejos de implementar y administrar, y más aún en infraestructuras de gran tamaño.
- **Escalabilidad**: Si no se seleccionan bien los componentes, no se diseña correctamente la solución o no se realizan las labores de mantenimiento adecuadas, NAC presentará problemas de escalabilidad.

Evidentemente, este proyecto resulta inviable llevarlo a cabo sobre una red corporativa y en producción de gran tamaño, por lo que en un principio serán más ventajas que inconvenientes los que nos encontremos. Aun así, a nivel personal considero que la mejor manera de implementarlo es de forma progresiva (aún en infraestructuras grandes), primero, porque así se puede ir analizando el rendimiento de la solución a medida que vaya creciendo, permitiendo tomar las medidas necesarias para que la escalabilidad no presente problemas, y segundo, porque la complejidad se reduce enormemente si se opta por esta estrategia de implementación.

Analizado este aspecto, también se deberá tener en cuenta el modo en que operará el sistema y el tipo de instalación que más se adapte a las necesidades de la infraestructura, por lo que ambos aspectos también requieren su estudio… [5] [14]

2.2.2. Modos de operación

- **Pre-admission:** En este modo, el dispositivo que trata de conectarse a la red es analizado por NAC **antes** de permitirle el acceso.
- **Post-admission:** Sin embargo, en *post-admission*, por defecto se permite la conexión del dispositivo a la red, para posteriormente NAC supervisar su tráfico y comportamiento.

Como se puede observar, ambos modelos operan de manera muy diferente, sin embargo, no son excluyentes, pudiendo obtener los beneficios del uno y el otro. Su elección se podría considerar un aspecto clave a la hora de diseñar la solución, por lo que se tendrá muy en cuenta llegado el momento.

2.2.3. Tipos de implementación NAC

- **Inline:** Normalmente es una solución basada en hardware, que ubicada en un lugar estratégico de la red (donde pueda capturar todo el tráfico), combina la toma de decisiones de acceso de dispositivos con el análisis de tráfico.
- **Out-of-band:** Se trata de un servidor que alberga una solución basada en *software,* de tal manera que el resto de dispositivos de la infraestructura de red, como *switchs*, puntos de acceso o *routers*, consultarán a dicho servidor para permitir o denegar el acceso a equipos o usuarios.

Nuevamente, nos encontramos ante dos tipos de implementación totalmente diferentes, por lo que este también se convierte en otro aspecto clave a la hora de diseñar la solución.

2.2.4. Casos de uso

Por un lado, los sistemas de control de acceso basados en perfiles resultan ideales sobre cualquier entorno que requiera diferentes tipos de conectividad, siendo especialmente adecuados para infraestructuras que presenten las siguientes necesidades; BYOD, dispositivos IoT, acceso para personal no corporativo o movilidad de empleados dentro de la organización. [12] [14].

Por otro lado, NAC también se presenta como la solución más adecuada sobre entornos que, aun no teniendo las necesidades anteriores, desean disponer de un alto grado de seguridad, visibilidad y monitorización de activos. En estos casos, si bien la parte de segmentación no resulta tan necesaria, sí que lo es todo lo relacionado con la política de seguridad definida.

Gracias al estudio realizado hasta el momento, ya se podría comenzar a definir determinados aspectos de su diseño y posterior implementación. Sin embargo, también conviene realizar el estudio sobre cómo operan estos sistemas a nivel más técnico, con el fin de obtener una visión más amplia sobre su funcionamiento, y así poder diseñar mejor la solución.

2.3. Autenticación en capa 2 del modelo OSI

Como se ha comentado anteriormente, los sistemas de control de acceso comienzan su función en capa 2 mediante la detección del dispositivo conectado y posterior decisión sobre su acceso a la red. Para que ello se pueda llevar a cabo, se debe implementar algún método de seguridad en el equipo encargado de facilitar la conectividad (normalmente un Switch o punto de acceso), propósito este que lograremos gracias al estándar **802.1x, MAB** (*MAC Authentication Bypass)* o ***WPA Enterprise***.

2.3.1. Estándar 802.1x

802.1x puede considerarse el método por excelencia y a la vez el más adecuado y seguro para un sistema de control de acceso basado en perfiles. Ello es debido a que cumple con todas las funcionalidades necesarias para autenticar a los dispositivos y/o usuarios antes de permitir su acceso a la red, por lo que se convierte en un protocolo imprescindible para lograr el objetivo de este trabajo.

De manera muy resumida, su modo de operar se basa en la siguiente lógica; en primer lugar, cuando un dispositivo se conecta a la red, bloquea su acceso, solicitándole credenciales para verificar su identidad. Tras ello, las envía a un servidor de autenticación, el cual se encargará de verificarlas. Si son válidas, se le permite el acceso a la red. De lo contrario, se bloqueará su acceso o se aplicará la medida adoptada para estos casos.

Este proceso, que se explicará en detalle en las próximas líneas, requiere de los siguientes elementos para poder llevarse a cabo… [15]

- ***Client / supplicant**:* Es el dispositivo que intenta acceder a la red.
- ***Access device / Authenticator**:* Es el elemento de la red al cual se conecta el cliente y sobre el cual se ejecuta 802.1x. Normalmente un Switch o AP.
- ***Radius Server**:* Es el servidor al que el *Access device* envía las credenciales de acceso del cliente, con el fin de que puedan ser verificadas.
- ***Identity provider / Directory**:* Es el almacén de credenciales donde el servidor Radius consultará para verificarlas.

Como se puede observar, el proceso de autenticación requiere la comunicación entre diferentes dispositivos para que se pueda llevar a cabo con éxito. Por tanto, resulta imprescindible hacer uso de algún método que se encargue de ello. En 802.1x, el protocolo utilizado para tal propósito es **EAP** (*Extensible Authentication Protocol),* el cual, además de aplicar el formato adecuado a la comunicación para que todos los dispositivos puedan interactuar entre sí, también facilita diferentes tipos de autenticación, siendo los más útiles y seguros para el propósito de este trabajo los siguientes [16]:

- ***EAP-TLS***: Se basa en la autenticación entre el cliente y el servidor mediante certificados digitales, lo que lo convierte en el método más robusto y seguro.
- ***EAP-PEAP**:* En este caso solo se requiere certificado en el lado del servidor, de tal manera que el cliente se autentica en el mismo haciendo uso de algún otro método, como puede ser el protocolo **MSCHAPv2** (usuario y contraseña).

Cabe destacar que EAP admite más opciones de autenticación, sin embargo, se han descartado para este proyecto porque, o bien son inseguras (*EAP-MD5*), o bien su propósito está orientado a entornos de telefonía móvil (*EAP-SIM, EAP-AKA*), o bien fueron desarrolladas por fabricantes (*EAP-TTLS, EAP-FAST*). [17]. Por tanto, la elección entre ***EAP-TLS*** o ***EAP-PEAP*** también se convierte en un aspecto clave a la hora de diseñar la solución.

Adquiridos los conocimientos básicos sobre *EAP*, y profundizando sobre la lógica aplicada y comentada anteriormente, nos encontramos con el siguiente modo de operar…

Primero, un cliente (o *supplicant*), compatible con 802.1x, se conecta a un *access device,* configurado también para operar en 802.1x. En este punto, el *access device* bloquea cualquier tipo de tráfico que atraviese el enlace, a excepción de aquel generado por el protocolo EAP, el cual aplicará un formato determinado para este tipo de comunicación, denominado ***EAPoL*** (*EAP over Lan).* Todo ello, visto de manera gráfica…

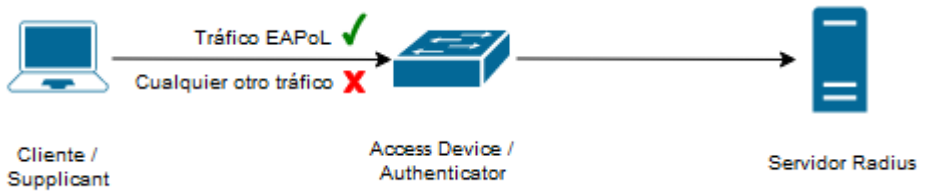


Figura 4: Autenticación 802.1x (Cliente - Switch)
Fuente propia (realizada con draw.io)

Tras ello, el cliente enviará sus credenciales mediante EAP, que serán recibidas y aceptadas por la interfaz del *access device* con la cual conecta. En este punto, el switch o punto de acceso actuará como *Authenticator*, enviando las credenciales al servidor *Radius* para que puedan ser verificadas. Para ello, EAP aplica un nuevo formato de paquete, denominado ***EAP over Radius***, el cual es enviado a través del puerto UDP 1812…

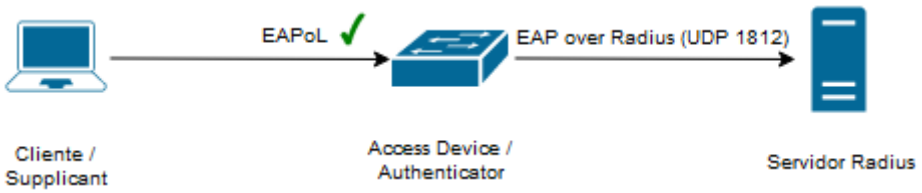


Figura 5: Autenticación 802.1x (Switch - Radius)
Fuente propia (realizada con draw.io)

Por último, el servidor Radius consultará las credenciales recibidas (en su propio servidor o en un tercero) y devolverá la respuesta al Switch, el cual aplicará la acción oportuna sobre la interfaz.

Con todo ello, si la autenticación finaliza con éxito se produce un intercambio de paquetes denominados "***handshake***" entre el cliente y el *access device*. El propósito de estos mensajes no es otro que verificar que el cliente sigue activo, con el fin de mantener la interfaz operativa o volver a bloquearla en caso de no obtener respuesta.

2.3.2. Mac Authentication Bypass (MAB)

802.1x se presenta como la solución ideal de autenticación para el propósito de este proyecto. Sin embargo, es imprescindible que el cliente sea compatible con

este método para poder implementarse. Este hecho se convierte en un inconveniente en determinados casos, ya que ciertos dispositivos, como impresoras o teléfonos IP, no suelen incorporar esta funcionalidad.

Para estas situaciones excepcionales podremos hacer uso de ***MAB***, gracias al cual podremos autenticar a los clientes en base a su dirección MAC. Entrando en detalle, su modo de operar es muy similar a 802.1x, pero el *access device*, en lugar de esperar un mensaje de autenticación EAPoL, lo que hará será aceptar la primera trama enviada por el dispositivo, bloqueando las restantes. Gracias a ello aprende su dirección MAC, la cual será enviada como usuario y contraseña al servidor Radius, que se encargará de verificar si está autorizada o no…

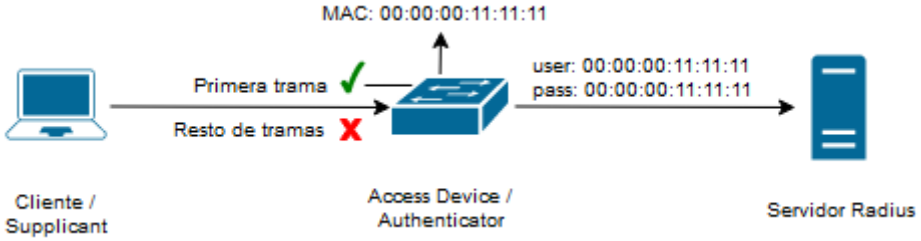


Figura 6: Autenticación MAB
Fuente propia (realizada con draw.io)

Evidentemente, este método resulta más inseguro que 802.1x, ya que una dirección MAC puede ser falseada con facilidad, por lo que lo ideal sería aplicarlo únicamente sobre dispositivos con accesos restringidos, como podrían serlo las impresoras. Además, dependiendo del modo aplicado, podría aceptar más de un dispositivo, por lo que este aspecto también hay que tenerlo presente en el diseño de la solución. Los modos de operación de MAB son los siguientes: [19]

- ***Single-host-mode****:* Donde únicamente se acepta un dispositivo (una MAC).
- ***Multi-Domain Authentication host mode****:* Donde se aceptará un dispositivo de voz (en la vlan de voz) y otro de datos (vlan de datos).
- ***Multi-Authentication host mode****:* Se permitirá la autenticación de multitud de dispositivos.
- ***Multi-host mode***: Donde se permitirán multitud de dispositivos, pero solo se autenticará al primero en conectarse.

Por último, MAB puede implementarse de manera independiente o como respaldo de 802.1x. En este último caso, el *access device* intentará como primera opción la autenticación por 802.1x, y si no obtiene respuesta del cliente (3 intentos de 30 segundos cada uno), aplicará MAB. [19]

2.3.3. WPA Enterprise

En cuanto a redes inalámbricas se refiere, el modo *enterprise* de WPA (en cualquiera de sus versiones) habilita la autenticación del cliente a través de un servidor Radius, lo que permite aplicar medidas y perfiles de seguridad personalizados sobre cada uno de ellos. Realmente, este modo hace uso de 802.1x para lograr su objetivo, por lo que se puede aplicar la teoría presente en

el apartado "*2.3.1"*, con la única diferencia de que la comunicación entre el cliente y el *access device* se llevará a cabo de manera inalámbrica. [20]

2.4. Sistema NAC

Analizados los métodos de autenticación que se deben tener en cuenta para el diseño de la solución, faltaría por realizar el estudio sobre los elementos necesarios para crear y asignar reglas y perfiles de acceso a cada usuario autenticado. Por un lado, las **reglas** definirán los requisitos que debe cumplir un dispositivo o usuario para acceder a la red, como, por ejemplo, pertenecer a un determinado grupo del directorio activo, mientras que los **perfiles** se encargarán de establecer los permisos de conectividad que se le otorgarán una vez concedido el acceso, como la asignación de una determinada VLAN.

Ambas funciones requieren diferentes sistemas, los cuales podrán ser implementados de manera independiente o formar parte de una solución unificada. El primero de los casos se suele dar al optar por *software* de código abierto, donde normalmente se deberá seleccionar uno que se encargará de analizar las características y/o vulnerabilidades de los dispositivos, como puede serlo *OpenVass*, y otro capaz de asignar un perfil de acceso a dicho dispositivo o usuario, como es el caso de *PacketFence*. Por su contra, una solución unificada incluye todas estas funciones en el mismo equipo, sin embargo, suelen ser sistemas propietarios, como *ClearPass* de Aruba HP.

Sea como fuere, al conjunto de todas estas características se le conoce como sistema NAC [23], servidor NAC [22] o *Network Policy Server* [21], dependiendo de la fuente consultada, y representa otro elemento clave a la hora de diseñar la solución. En este sentido, también hay que tener en cuenta que puede presentar dos modos de operar; con agente o sin agente. Con **agente** requiere la instalación de un software en los dispositivos, que contactará con los servidores y facilitará información detallada sobre el dispositivo en cuestión. Por su contra, un sistema NAC **sin agente** no requiere dicho software, por lo que se basará en 802.1x para obtener información de los equipos, la cual será menos detallada que aquella facilitada por el modelo con agente.

Con todo ello, el objetivo de este trabajo consiste en implementar un sistema de control de acceso basado en perfiles haciendo uso de soluciones *open source,* por lo que conviene analizar las posibles alternativas que tenemos para ello. Históricamente, tres de las opciones más robustas para este propósito han sido *PacketFence*, *OpenNAC* y *FreeNAC*. Sin embargo, el proyecto de *FreeNAC* ha sido descontinuado, mientras que *OpenNAC* ha pasado a ser una solución de pago. Por su contra, *PacketFence,* además de no requerir licencia, incluye en un solo sistema el servidor *Radius* y la creación de perfiles de acceso, entre otras muchas características, por lo que contempla todo lo necesario para este proyecto, convirtiéndose así en la solución ideal.

2.5. Ataques sobre NAC: Medidas de prevención

Al igual que cualquier sistema o protocolo, NAC no está exento de sufrir ataques, por lo que conviene conocer las técnicas más habituales utilizadas por los ciberdelincuentes con el fin de poder aplicar las medidas adecuadas y tratar de evitarlos. Normalmente, el objetivo principal consiste en obtener las credenciales de usuarios legítimos, para así poder acceder a la red con los privilegios de los mismos. Para lograrlo, los métodos más comunes, y sobre los que se deberán tomar medidas de protección, son los siguientes:

Por un lado, ya sea en una red cableada o inalámbrica, la mayoría de ataques se basan en el método **MITM** (*Men In The Middle*), donde el atacante intentará interceptar el proceso de autenticación del usuario para descifrar sus credenciales y obtenerlas en texto claro. Contra ello, la mejor medida de protección se basa en aplicar métodos de autenticación robustos, donde la comunicación se lleve a cabo de manera cifrada. Por tanto, se recomienda hacer uso de ***EAP-TLS**,* el cual se basa en certificados digitales tanto en el lado del cliente como del servidor, imposibilitando así el éxito del ya mencionado ataque. En el caso de no ser posible aplicarlo, resulta importante asegurar la autenticación mediante algún protocolo que cree un túnel TLS entre cliente y servidor, como podría serlo ***EAP-PEAP***. [24] [25]

Continuando con los ataques MITM, estos resultan especialmente peligrosos y sencillos de llevar a cabo en redes inalámbricas, por lo que habrá que prestar especial atención ante estos escenarios. Cabe recordar que con *WPA2/3 Enterprise* realmente se aplica 802.1x como método de autenticación, por lo que nuevamente *EAP-TLS* se convierte en la opción ideal. Sin embargo, *EAP-PEAP* puede ser fácilmente atacado mediante la creación de un punto de acceso fraudulento. Para evitarlo, resulta importante habilitar la autenticación del punto de acceso mediante la aplicación de un certificado. [24] [25]

Por último, para los dispositivos que no compatibles con 802.1x, como impresoras o IoT, normalmente se habilitará *MAB* como método de autenticación. Para estos casos, llevar a cabo el ataque no supone ninguna complejidad, ya que falsificar la MAC de un equipo no resulta nada complicado. Es por ello que los permisos concedidos a estos dispositivos se deben restringir lo máximo posible, de tal manera que, si un ataque a este método tiene éxito, el atacante no disponga de acceso a recursos corporativos. [26]

El objetivo de realizar un estudio tan detallado sobre las características de este tipo de sistemas no ha sido otro que obtener los conocimientos necesarios para tomar las mejores decisiones durante el diseño e implementación de los productos. Además, esta información también facilita la detección y resolución de incidencias en caso de que sucedieran. Con todo ello, comienzo el diseño de la solución…

3. Diseño de los productos

3.1. Diseño de una red segura basada en perfiles de acceso

3.1.1. Diseño lógico de la red y selección del hardware

NAC es una solución cuyo diseño depende por completo de las características de la organización en la que se desee implementar. Es por ello que para este proyecto se ha creado un entorno de laboratorio que intenta reflejar las necesidades más comunes de la mayoría de redes corporativas, para, sobre el mismo, realizar el diseño e implementación del acceso basado en perfiles, asemejándolo lo más posible a cualquier entorno real.

De esta manera, el laboratorio escenifica una compañía que presenta las siguientes características:

- Está compuesta por diferentes departamentos, donde, por seguridad, cada uno de ellos deberá pertenecer a una **VLAN** diferente, que a su vez dispondrá de permisos específicos.
- Se permite la movilidad de los empleados dentro de la organización, pudiendo conectarse a través de cualquier punto de red disponible.
- Dispone de servidores e impresoras, que también deben estar ubicados en subredes independientes, con accesos muy restringidos.
- Por último, se ha habilitado conectividad Wifi para que los empleados puedan acceder a la red a través de sus portátiles corporativos. Es importante recalcar que esta Wifi solo es para empleados, no para invitados, por tanto, deberán autenticarse antes de permitir su acceso.

Con ello, la red presenta el siguiente diseño lógico a alto nivel:

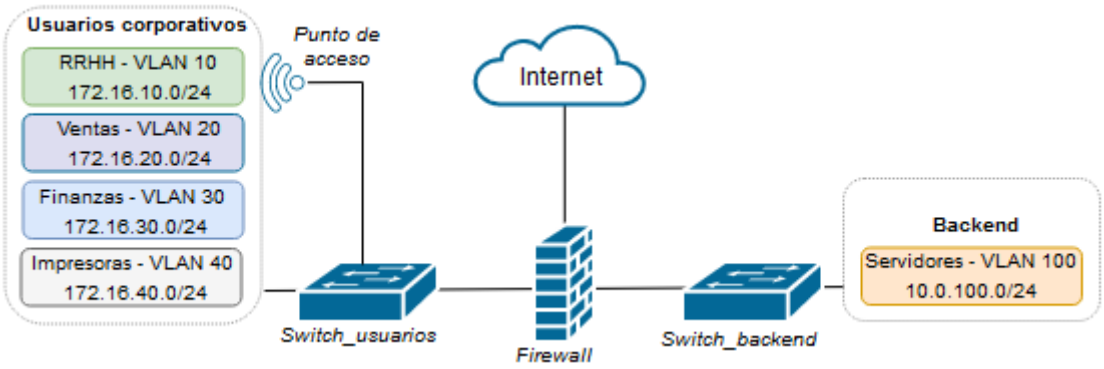


Figura 7: Esquema de red de laboratorio
Fuente propia (realizada con draw.io)

Como se ha mencionado en apartados anteriores, para implementar el control de acceso basado en perfiles resulta imprescindible que el *hardware* sea compatible con 802.1x y/o *MAB*. De ambos, *MAB* resulta inseguro, por lo que tan solo se implementará sobre aquellos equipos incompatibles con 802.1x, como pueden serlo las impresoras (red cableada). Por tanto, para este proyecto, el

switch de usuarios deberá poder operar con ambos métodos, mientras que el punto de acceso bastará con que permita el tipo de validación *WPA2-Enterprise* (802.1x). Teniendo en cuenta todo ello, la elección de ambos dispositivos resulta crucial para poder implementar NAC con garantías.

En este sentido, existen muchas soluciones en el mercado que cumplen con estas características, donde la gran mayoría de ellas están orientadas a entornos corporativos. Para el laboratorio se ha optado por los siguientes elementos de red para implementar NAC:

- 2 *switchs* del fabricante **Enterasys**, modelo **B5G124-24** (*datasheet*)**.** El motivo de su elección es que, además de permitir operar con ambos métodos, presenta todas las características que debe tener un switch corporativo.
- 1 punto de acceso **ASUS**, modelo **RT-AX59U** (*datasheet*). El motivo de su elección es que, entre otros muchos métodos de seguridad, permite la autenticación WPA2-Enterprise, además de presentar una buena relación entre precio y funcionalidades.

Por otro lado, aunque el ***firewall*** no intervenga directamente en el proceso de autenticación de NAC, sí que gestiona los permisos en capa 3 que tendrá cada uno de los perfiles. Además, actúa como servidor DHCP y lleva a cabo el enrutamiento de todas ellas, por lo que también resulta crucial en la implementación. Para este caso, se ha optado por la solución de software **PfSense**. El motivo de ello es que se trata de un producto *open source* que incluye todas las características que se necesitan, además de ser una solución robusta y respaldada por una amplia comunidad.

Por último, una vez presentada la red de laboratorio, así como analizados y seleccionados los elementos de hardware necesarios para permitir el proceso de autenticación, se puede comenzar a diseñar la solución NAC, donde la primera acción consistirá en identificar las características que debe cumplir el producto que se implementará…

3.1.2. Características necesarias y selección del servidor NAC

En el estudio realizado en el capítulo 2 se evidenciaron numerosos aspectos que resultan clave a la hora de diseñar un sistema NAC. Como ya se mencionó, estas características dependerán de la infraestructura de red y necesidades de la organización, las cuales ya conocemos.

Cabe recordar que uno de los objetivos de este proyecto consiste en lograr que la implementación no resulte muy costosa en términos económicos. Por tanto, este aspecto también se tendrá en cuenta en las siguientes decisiones. Con ello, el primer paso que se ha llevado a cabo ha sido identificar los requisitos que deberá cumplir el sistema, para posteriormente decidir qué característica hará cumplir cada uno de ellos, así como el motivo de su elección. Todo ello queda reflejado en la siguiente tabla…

Requisito	Característica necesaria para implementarlo	Justificación
Aspectos generales		
Los dispositivos solo podrán acceder a la red una vez autorizado su acceso por el sistema.	NAC en modo de operación **pre-admission.**	De los dos modos de operación analizados en el capítulo 2, *pre-admission* es el que cumple con esta característica.
Con el fin de ahorrar costes, la solución debe estar basada en *software.*	El tipo de implementación debe ser **out-of-band.**	La implementación *in-line* está basada en hardware y son soluciones propietarias, por lo que su coste se encarece considerablemente. Por su contra, una solución *out-of-band* está basada en *software* y puede ser *open source*.
En los equipos de los usuarios no se debe instalar ningún *software* relacionado con el control de acceso.	- Tipo de operación: **sin agente**. - **802.1x**	Las soluciones con agente suelen incluir mayores características y beneficios, sin embargo, son sistemas propietarios que requieren una gran inversión en términos de licenciamiento. Es por ello que, con el fin de ahorrar costes, para este proyecto se optará por una solución sin agente. Sin embargo, los dispositivos sí que deben ser compatibles con 802.1x, para permitir la autenticación.
La autenticación se basará en usuario/contraseña de directorio activo y se transmitirá cifrada mediante túnel TLS.	- Método de autenticación: **EAP-PEAP**	Aunque EAP-TLS (autenticación con certificado de usuario) se considere más seguro, requiere la implementación de un sistema que actúe como autoridad certificadora, así como la generación de un certificado por usuario. Debido a la limitación de tiempo de ejecución de este proyecto, esta característica se propondrá para futuras mejoras. Por su contra, como la organización dispone de un AD, se puede hacer uso de EAP-PEAP como método de autenticación de manera segura, ya que las credenciales se transmitirán a través de un túnel TLS.

Permitir movilidad dentro de la organización: se debe identificar a los usuarios por departamento y asignarles permisos específicos de manera automática.	**Radius y AD**.	El servidor Radius es necesario para la autenticación en 802.1x, mientras que, gracias al directorio activo, podremos comprobar a qué departamento pertenece el usuario. De esta manera, Radius contactará con el AD en el proceso de autenticación, y en base al resultado asignará al usuario un determinado perfil de acceso. Por tanto, la solución deberá permitir la opción de que Radius base la autenticación en AD.
Requisitos para el acceso a la red cableada		
Autenticar a los usuarios antes de acceder a la red.	**802.1x**	La solución debe ser compatible con 802.1x para permitir la autenticación de los usuarios y la asignación de perfiles de seguridad. Cabe mencionar que los usuarios únicamente se podrán validar en la red mediante este protocolo, nunca mediante MAB, ya que resulta inseguro.
Autenticar dispositivos incompatibles con 802.1x.	**MAB**	Solo se aplicará para las impresoras de red incompatibles con 802.1x. Por tanto, la solución seleccionada también debe permitir la autenticación *MAB*.
Requisitos para el acceso inalámbrico		
Autenticar a los usuarios antes de acceder a la red	**WPA2-Enterprise**	*WPA2-Enterprise* se basa en 802.1x, por lo que este es otro motivo para que la solución deba ser compatible con dicho método.

Tal vez llame la atención el hecho de que no se ha definido ningún requisito que afecte a los servidores de la organización. El motivo de ello es que la función de estos equipos consiste en prestar servicios, por lo que conviene que su configuración y conectividad sea lo más estática posible. Además, siempre estarán ubicados en el mismo lugar físico y conectados a la misma toma de red, mientras que su seguridad y accesos estarán total y constantemente controlados por los administradores del sistema. Es por todo ello que **no** se considera necesario ni adecuado aplicar NAC sobre la red de servidores.

Resumiendo, se contemplan dos escenarios de conectividad para usuarios corporativos; cableado e inalámbrico. En ambos casos se establecerá el sistema de control de acceso basado en perfiles, de tal manera que a los usuarios de

cada departamento se les asignará siempre el mismo perfil de conectividad de manera automática, sin importar el dispositivo ni tipo de conexión utilizada para acceder a la red, logrando así un elevado grado de seguridad en cuanto a acceso se refiere. Para poder llevarlo a cabo, la solución NAC que se implemente deberá contemplar todas las características indicadas en la tabla anterior.

En este sentido, analizando las diferentes soluciones *open source* (ya mencionados en el apartado 2.4), no cabe duda de que la mejor opción a implementar es **PacketFence**, ya que, además de contar con todas las características necesarias, está ampliamente respaldada y en constante evolución.

3.1.3. Diseño de una estrategia de autenticación

En cuanto a la autenticación, se han mencionado diferentes aspectos que se deben tener en cuenta para el diseño de la solución. Por un lado, el proceso comienza en el puerto del Switch o punto de acceso. Por otro lado, los usuarios solo podrán autenticarse mediante 802.1x, mientras que MAB solo se permitirá para impresoras. Además, la infraestructura presenta dos tipos de acceso, de manera cableada e inalámbrica. Todo ello implica tomar una serie de decisiones a implementar sobre el Switch, el punto de acceso y el servidor NAC. En el primero y segundo de los casos, para forzar la autenticación, mientras que, en el tercero, para validarla y asignar un perfil de conectividad.

En este punto hay que tener en cuenta que las impresoras únicamente se conectarán de manera cableada, por lo que el método de autenticación MAB tan solo se deberá aplicar en el Switch de usuarios. Con ello, las medidas que se han considerado adecuadas para cumplir con los requisitos expuestos anteriormente son las siguientes:

Medidas a adoptar en el Switch
Permitir dos métodos de autenticación en cada puerto, que se ejecutarán de manera secuencial en el siguiente orden: 1.- **802.1x** (opción prioritaria). 2.- **MAB** (opción secundaria). 3.- **Denegar acceso.** De tal manera que, cuando un dispositivo no sea compatible con 802.1x, se ejecutará MAB. Sin embargo, los usuarios no podrán validarse utilizando el segundo método, ya que en la configuración del servidor NAC se filtrarán los dispositivos que puedan autenticarse mediante el mismo (solo impresoras). Gracias a ello, cubrimos las necesidades de movilidad y autenticación.

Medidas a adoptar en el punto de acceso
Como al punto de acceso tan solo se conectarán usuarios corporativos, únicamente se permitirá el acceso mediante 802.1x, que en entonos inalámbricos se lleva a cabo mediante el tipo de autenticación *WPA2-Enterprise.* Por tanto, en este caso la secuencia será la siguiente: 1.- **WPA2-Enterprise** 2.- **Denegar acceso.**
Medidas a adoptar en el servidor NAC
Crear diferentes políticas de acceso, que se ejecutarán de manera secuencial y que deberán cumplir los siguientes requisitos… ・Para el caso de los usuarios, se comprobará a qué grupo del AD pertenece, y en base al resultado, se le asignará un perfil de conectividad específico. ・Para el caso de las impresoras, se creará una política que únicamente permita el acceso a sus direcciones MAC. Para el proyecto, supongamos que tan solo disponemos de una impresora con MAC 00-00-00-11-11-11. Con ello, las políticas que se consideran necesarias son: 1.- Si el usuario pertenece al grupo del AD "**RRHH**", se le asigna la **VLAN 10**. 2.- Si el usuario pertenece al grupo del AD "**Ventas**", se le asigna la **VLAN 20**. 3.- Si el usuario pertenece al grupo del AD "**Finanzas**", se le asigna la **VLAN 30**. 4.- Si el dispositivo tiene la MAC **00-00-00-11-11-11**, se le asigna la **VLAN 40**. De tal manera que, si por ejemplo, un usuario corporativo del grupo finanzas se conecta a la red (en cualquier ubicación), al autenticarse con el servidor NAC coincidirá con la política 3, y se le asignará el perfil de conectividad asociado, el cual lo agrega automáticamente en la VLAN 30. Si por el contrario el dispositivo no responde a 802.1x, será analizado por la política 4, donde, si su MAC es la 00-00-00-11-11-11, se le asignará el perfil de conectividad que lo asocia con la VLAN 40. Con ello, cumplimos la necesidad de asignar perfiles de acceso a cada tipo de usuario.

Por tanto, el comportamiento que deberá presentar la red cuando un usuario se conecte a la misma deberá ser el siguiente…

Para el acceso cableado se debe respetar la siguiente lógica…

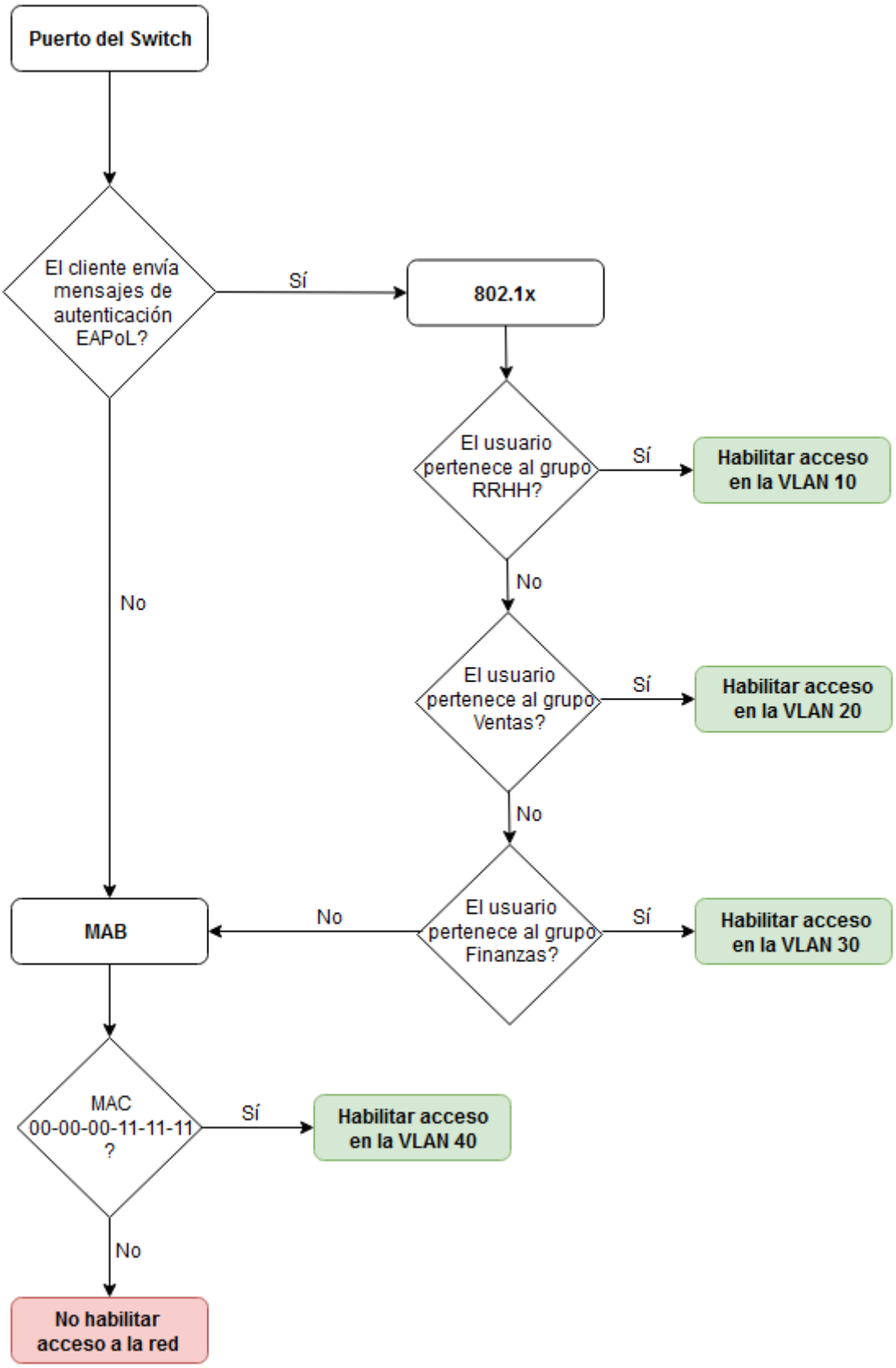


Figura 8: Diagrama de decisiones - Acceso cableado
Fuente propia (realizada con draw.io)

Mientras que para el acceso inalámbrico se llevaría a cabo de la siguiente manera...

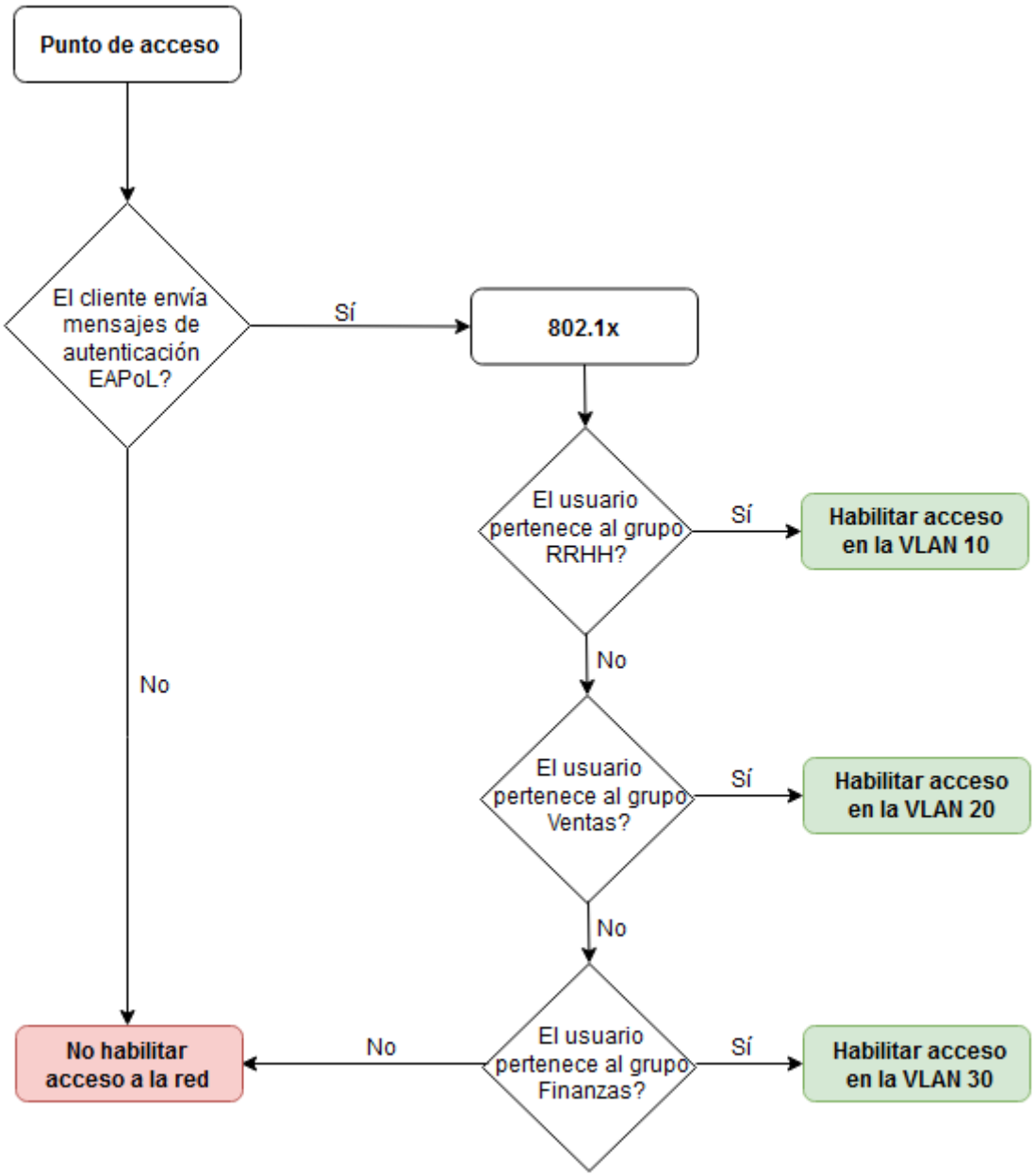


Figura 9: Diagrama de decisiones - Acceso inalámbrico
Fuente propia (realizada con draw.io)

3.1.4. Definición de perfiles de acceso NAC

Gracias a las decisiones de diseño tomadas anteriormente se ha logrado definir la manera en la que serán identificados los usuarios y asociados a un determinado perfil de acceso. Como se ha podido comprobar, NAC asignará a cada tipo de usuario una VLAN diferente, es decir, segmentos de red independientes, de tal manera que no existirá comunicación entre cada uno de ellos, logrando así mayor seguridad en la red.

Sin embargo, está política de acceso, que corresponde a **capa 2** (VLANs), también debe estar acompañada por políticas de capa 3, con el fin de controlar y limitar el tráfico permitido por cada subred, logrando con ello un mayor grado de seguridad. Por tanto, para completar el perfil de conectividad se deben definir los accesos que se permitirán a cada tipo de usuario, los cuales serán gestionados por el firewall. Para este proyecto, se han diseñado y considerado adecuados los siguientes accesos.

Perfil de acceso	Accesos asignados		
	VLAN	Segmento de red	Accesos permitidos en capa 3
RRHH	10	172.16.10.0/24	- Acceso a Internet. - Acceso a Herramienta de gestión de RRHH. - Acceso a Impresoras.
Ventas	20	172.16.20.0/24	- Acceso a Internet. - Acceso a base de datos de Inventario.
Finanzas	30	172.16.30.0/24	- Acceso a Herramienta de gestión de Finanzas. - Acceso a Impresoras.
Impresoras	40	172.16.40.0/24	- Sin accesos permitidos (solo se permitirán conexiones entrantes desde las VLAN de RRHH y Finanzas).

Cabe mencionar que, en una red corporativa en producción, los accesos en capa 3 serán más numerosos y estarán ajustados por protocolo. Sin embargo, para la red de laboratorio de este proyecto se ha considerado adecuado definir una pequeña muestra de todos ellos, ya que, ni se trata de un entorno en producción, ni el objetivo del trabajo se centra en el firewall.

3.1.5. Ubicación del servidor NAC

Por último, faltaría por definir la ubicación del servidor NAC dentro de la red y la dirección IP que se le asignará. La decisión tomada en este sentido será ubicar el NAC en la VLAN de servidores, asignándole la IP estática **10.0.100.10/24** y nombre **nac.TFGdanielperez.es.**

Una vez decidido el diseño y modo de operar, un ejemplo gráfico del proceso que se deberá llevar a cabo durante el acceso de un usuario autorizado a la red, sería el siguiente. (se tomará como ejemplo un usuario del departamento de ventas).

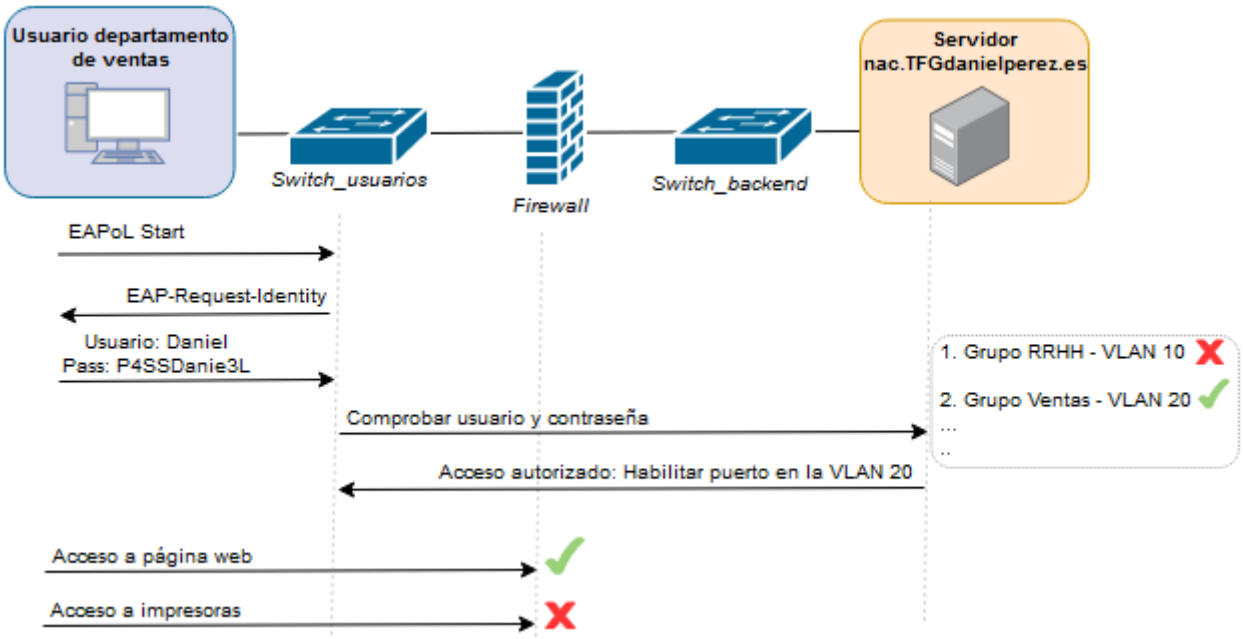


Figura 10: Ejemplo de acceso autorizado a la red
Fuente propia (realizada con draw.io)

Una vez concluido el diseño de la solución NAC, se puede comenzar a diseñar el software de gestión del mismo, el cual brindará a los administradores de red diferentes herramientas de administración y monitorización del sistema.

3.2. Diseño de herramienta de gestión

Como se ha podido comprobar, el sistema NAC que se ha implementado está compuesto por diferentes elementos, donde cabe destacar el switch, el servidor NAC, el firewall y el directorio activo (el punto de acceso no se menciona ya que su configuración es muy básica y estática). Esto significa que cualquier cambio, consulta o resolución de problemas requerirá analizar o realizar modificaciones en cada uno de ellos de manera individual, hecho que dificulta las labores administrativas a las personas encargadas de ello.

Con el fin de facilitar la gestión del sistema, se diseñará e implementará una herramienta, dirigida a los administradores de red, que centralice y automatice las labores de administración, monitorización y *troubleshooting*, de tal manera que, desde una única aplicación, se puedan llevar a cabo las tareas más relevantes del sistema. Su diseño se ha llevado a cabo de la siguiente manera…

3.2.1. Consideraciones generales

Antes de comenzar con el diseño de la herramienta y sus funciones, la primera decisión que se ha tomado ha sido seleccionar el tipo de aplicación que se implementará y el lenguaje que se utilizará para su desarrollo, ya que dichas características influyen directamente en el ya mencionado diseño.

Por un lado, se ha marcado como objetivo desarrollar una aplicación intuitiva, compatible con la totalidad de sistemas operativos, que sea poco exigente en el consumo de recursos del equipo cliente, y que a su vez no requiera instalación. Realizando un análisis de las diferentes posibilidades, se ha considerado como opción más adecuada el desarrollo de una **aplicación web**, ya que cumple con todos los requisitos recién expuestos.

Por otro lado, el hecho de que la aplicación sea web implica hacer uso de código **HTTP**. Sin embargo, también resultará necesario ejecutar operaciones incompatibles con este lenguaje, las cuales se llevarán a cabo mediante programación en **Phyton**. La elección de este lenguaje sobre otros es debido a que, a nivel de red, se trata de la opción más habitual para el desarrollo de *scripts,* ya que dispone de librerías con multitud de funciones adecuadas para ello, y, por tanto, ideales para el sistema NAC que se implementará.

Por último, teniendo en cuenta que la aplicación será HTTP y basada en Phyton, se ha buscado alguna solución que actúe como servidor web para alojar la herramienta, y que a su vez sea capaz de operar con ambos lenguajes. Realizando un análisis comparativo entre soluciones de código abierto, las dos mejores opciones son Django y Flask, ya que, aunque existen más alternativas, estas están respaldadas y mantenidas por una amplia comunidad. De ambas, **Flask** es más ligero y sencillo de implementar, por lo que se ajusta más a los objetivos de la aplicación que se desea diseñar. [28]

3.2.2. Análisis de requisitos funcionales

Otro aspecto relevante a tener en cuenta para el diseño de la herramienta será decidir qué funciones se incorporarán y cuál es la finalidad de cada una de ellas, objetivo este que logramos gracias al análisis de requisitos funcionales. Sin embargo, son tantas las operaciones que se podrían llevar a cabo sobre un sistema NAC que no resultaría, ni viable ni adecuado, desarrollarlas todas en una primera versión de la aplicación. Es por ello que el resultado del análisis de requisitos para esta primera versión es el siguiente:

ID	Descripción	Ámbito	Funcionalidad
Req00	**Autenticación del usuario**	Sistema	El sistema debe permitir a los usuarios iniciar sesión.
Req01	**Crear nuevo perfil de acceso**	Administración	El sistema debe permitir al usuario crear un nuevo perfil de acceso y configurarlo automáticamente en todos los elementos de NAC.
Req02	**Agregar nuevo Switch**	Administración	El sistema debe permitir configurar automáticamente un Switch y agregarlo al sistema NAC.
Req03	**Ver usuarios conectados**	Monitorización	La aplicación debe permitir ver información de usuarios actualmente conectados.

Req04	**Ver usuarios con acceso denegado**	Monitorización	El sistema debe permitir ver accesos de usuarios que han sido denegados.
Req05	**Ver accesos de red bloqueados**	Monitorización	La herramienta debe permitir ver las infracciones de accesos de red.
Req06	**Capturar paquetes**	Troubleshooting	La aplicación debe permitir al usuario capturar y almacenar paquetes de autenticación del servidor NAC en formato *pcap,* que posteriormente podrá descargar para su análisis.
Req07	**Visualización de logs**	Troubleshooting	El sistema debe permitir al usuario ver los logs de todos los dispositivos del sistema NAC.

3.2.3. Diseño de mapa de navegación (arquitectura web)

Con las funciones ya definidas se puede continuar el diseño de la aplicación con el mapa de navegación, el cual expone de manera gráfica y a alto nivel la disposición y ubicación de cada una de las opciones que se presentarán al usuario. En este aspecto, como se facilitarán tres ámbitos, y uno de los objetivos de la aplicación es ser intuitiva y amigable, se creará un menú para cada uno de ellos. De esta manera, la interfaz web quedará estructurada en tres niveles de la siguiente manera:

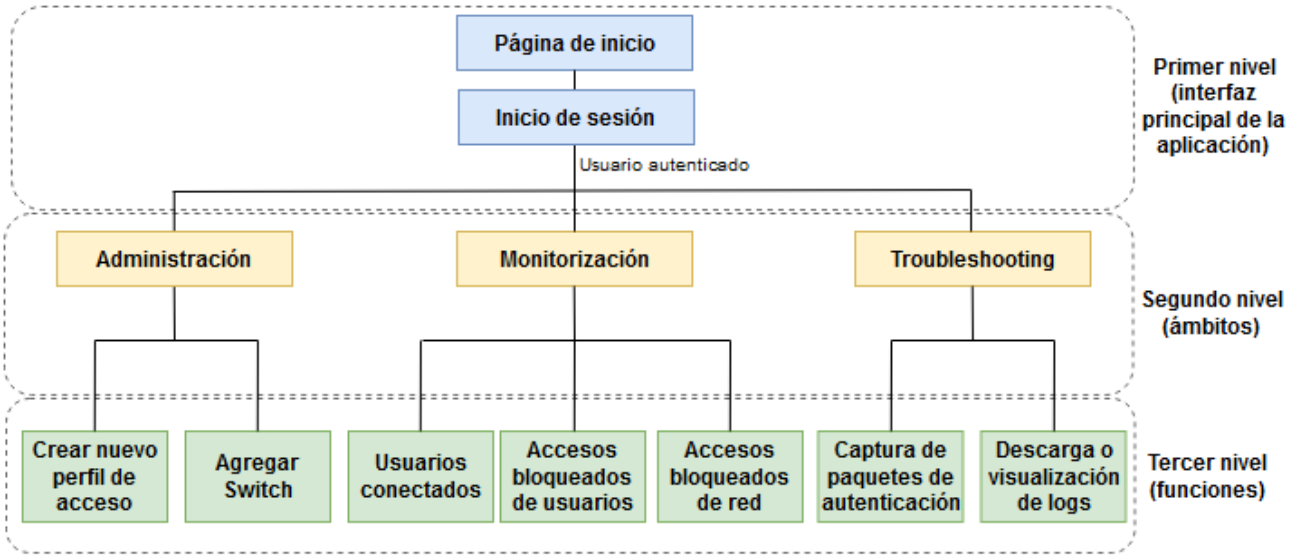


Figura 11: Mapa de navegación web
Fuente propia (realizada con draw.io)

El objetivo de los niveles, aparte de lograr que la aplicación sea intuitiva, consiste en facilitar la escalabilidad a la hora de agregar nuevas funciones, de tal manera que se mantendrá una estructura en la que los ámbitos siempre serán representados en segundo nivel, mientras que las funcionalidades en el tercero.

Del mismo modo, el primer y segundo nivel siempre se podrán tratar como capas de interfaz, ya que su función únicamente consiste en representar de manera gráfica las opciones disponibles. Por su contra, el tercer nivel deberá tratarse, tanto en capa de interfaz como de aplicación, ya que, además de presentar opciones al usuario, también interactuará directamente con los elementos de la infraestructura de red mediante la ejecución de *scripts*.

3.2.4. Modelado de la aplicación web

Entrando en el diseño interno de la aplicación, en primer lugar se definirá el modo de operar que deberá presentar cada función, en segundo lugar, las relaciones y propiedades de cada uno de los elementos presentes en la misma, es decir, su diagrama de clases UML, y en tercer lugar, la lógica de decisiones que se debe cumplir. Llegados a este punto resulta necesario mencionar que dicho diseño se basará en la metodología **OOWS** (*Object Oriented Web Solution*) que es una extensión de UML diseñada para la representación de aplicaciones web, donde una página puede ser tratada como un objeto. [29] [30]

Retomando el modo de operar, para que cada función pueda llevarse a cabo se deberá cumplir el siguiente comportamiento:

- **Inicio de sesión**: Mostrar formulario solicitando nombre de usuario y contraseña → Validar los datos → Habilitar o denegar acceso.
- **Crear nuevo perfil de acceso**: Mostrar formulario solicitando nombre del nuevo perfil, id de VLAN y rango de red → Verificar que el perfil no existe → Crear la VLAN en el Switch y propagarla por los puertos necesarios → Crear una interfaz en el firewall para enrutar la nueva subred → Crear rango DHCP → Crear grupo en el AD para incluir los usuarios del nuevo perfil → Crear nuevo rol en el servidor NAC.
- **Agregar nuevo Switch**: Mostrar formulario solicitando IP del Switch → Configurar las VLAN de los perfiles en el Switch → Configurar la autenticación 802.1x y MAB en los puertos → Configurar el servidor NAC como fuente de autenticación (Radius).
- **Ver accesos bloqueados**: Capturar del firewall los accesos de red que han sido bloqueados y mostrarlos al usuario.
- **Usuarios conectados**: Capturar del servidor NAC la información de conexiones validadas con éxito y mostrarlas al usuario.
- **Usuarios con acceso denegado**: Capturar del servidor NAC la información de conexiones que han sido denegadas y mostrarlas al usuario.
- **Captura de paquetes**: Capturar todo el tráfico de autenticación de usuarios contra el servidor NAC que se está generando en tiempo real → Almacenar el resultado en un fichero *pcap* → Descargar fichero *pcap* en el equipo del usuario.
- **Descarga de logs**: Mostrar al usuario un listado de logs disponibles relacionados con el sistema NAC → Mostrar al usuario el contenido del fichero log seleccionado.

Con ello, y también teniendo en cuenta el mapa de navegación, el diseño UML de relaciones estáticas entre clases y el diagrama de actividades de la aplicación quedan representados de la siguiente manera:

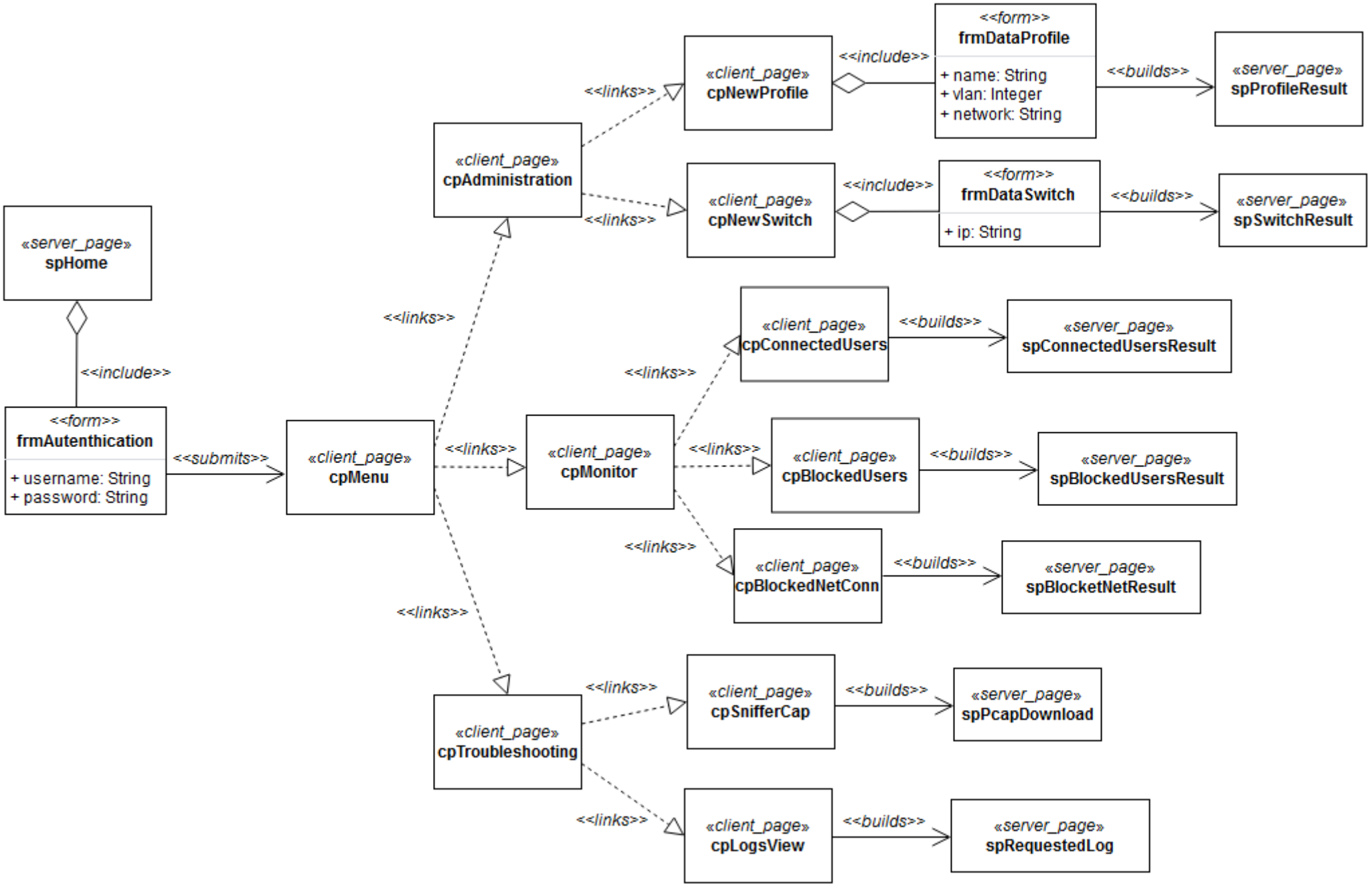


Figura 12: Diagrama de clases estáticas UML

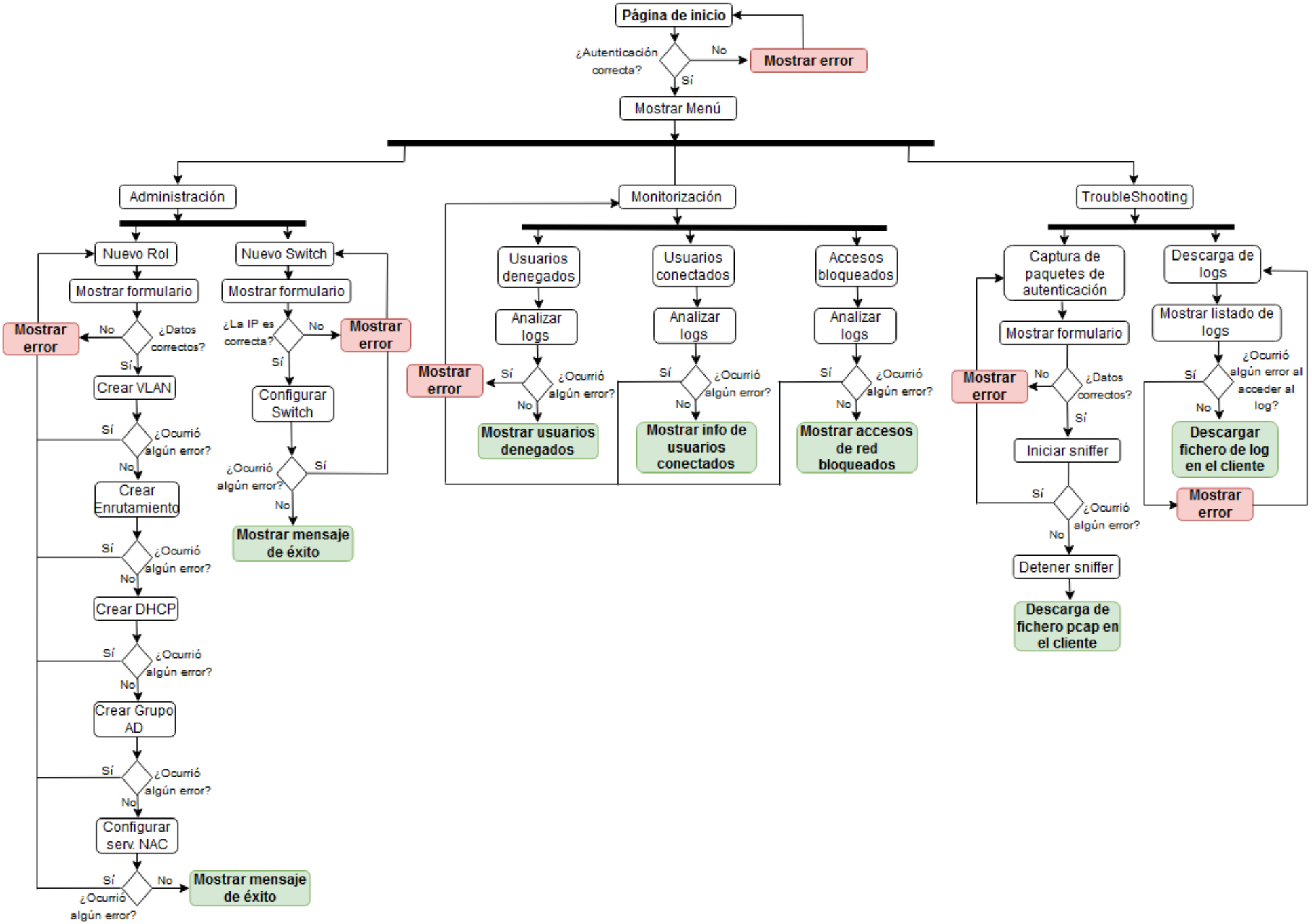


Figura 13: Diagrama de actividades del software de gestión

3.2.5 Actores y casos de uso

Por último, desde el punto de vista del usuario y la interacción de este con las funcionalidades de la aplicación, se diseñan los siguientes casos de uso. Cabe recordar que esta aplicación está orientada únicamente a los/as administradores/as de red del sistema, por lo que estos serán los únicos actores.

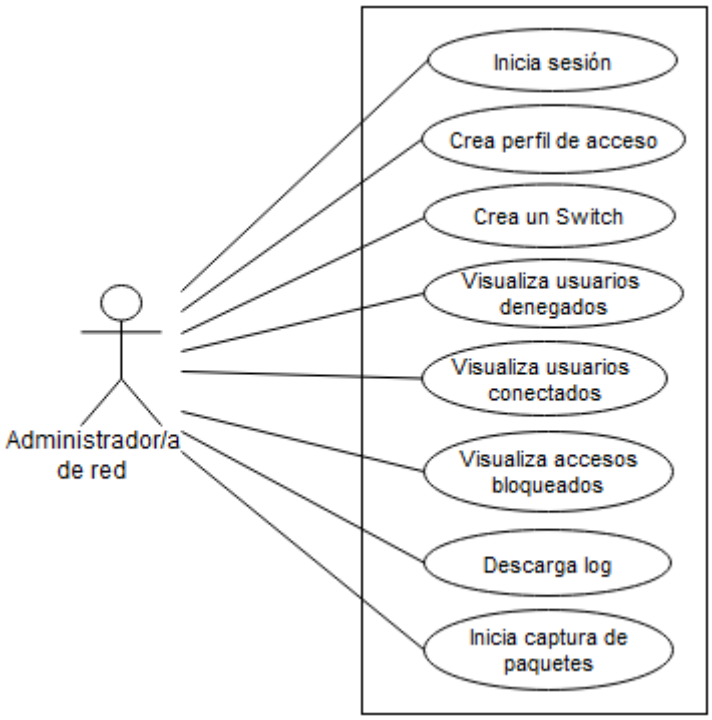


Figura 14: Diagrama UML de casos de uso
Fuente propia (realizada con draw.io)

Donde cada uno de ellos deberá cumplir las siguientes características…

Caso de uso / ID	**Inicia sesión / CU001**
Descripción	El usuario intenta acceder al sistema. El sistema deberá validarlo.
Ámbito	Sistema
Actor	Administrador/a de red
Requisitos	El usuario debe disponer de usuario y contraseña.
Escenario principal de éxito	1. El usuario accede a la página principal. 2. El usuario introduce su usuario y contraseña en el formulario 3. El sistema valida los datos. 4. El usuario accede al nivel dos de navegación.
Escenarios alternativos	1. Los datos introducidos son incorrectos. 2. El sistema impide el acceso al usuario y lo mantiene en el nivel uno de navegación.

Caso de uso / ID	**Crea perfil de acceso / CU002**
Descripción	El usuario crea un nuevo perfil de acceso en el sistema NAC. El servidor ejecuta todas las tareas necesarias.
Ámbito	Administración

Actor	Administrador/a de red
Requisitos	El usuario deberá proporcionar un nombre de perfil, id de VLAN y rango de red.
Escenario principal de éxito	1. El usuario introduce los datos en el formulario. 2. El servidor se conecta a los diferentes dispositivos del sistema NAC y ejecuta las tareas necesarias. 3. El servidor muestra al usuario un mensaje de éxito.
Escenarios alternativos	1. Los datos introducidos son incorrectos. 2. El perfil indicado ya existe. 3. El servidor encuentra algún error al intentar configurarlo. 4. Para todos los escenarios anteriores, el servidor muestra mensaje de error al usuario y lo devuelve a la página de creación de nuevo rol.

Caso de uso / ID	**Crea un Switch / CU003**
Descripción	El usuario desea agregar un nuevo Switch al sistema NAC.
Ámbito	Administración
Actor	Administrador/a de red
Requisitos	La IP proporcionada por el usuario debe ser la del Switch.
Escenario principal de éxito	1. El usuario introduce los datos en el formulario. 2. El servidor se conecta al Switch y configura todos los parámetros necesarios. 3. El servidor muestra al usuario un mensaje de éxito.
Escenarios alternativos	1. La IP facilitada no tiene el formato adecuado. 2. La IP es válida, pero no corresponde a un Switch. 3. El servidor no puede conectarse al Switch. 4. Para todos los escenarios anteriores, el servidor muestra mensaje de error al usuario y lo devuelve a la página de creación de un nuevo Switch.

Casos de uso / ID	**Visualiza usuarios denegados / CU004** **Visualiza usuarios conectados / CU005** **Visualiza accesos bloqueados / CU006**
Descripción	El usuario desea visualizar información de usuarios del sistema NAC.
Ámbito	Monitorización
Actor	Administrador/a de red
Requisitos	El sistema debe tener acceso al sistema de ficheros del servidor NAC y firewall.
Escenario principal de éxito	1. El usuario accede a visualizar información de los usuarios del sistema NAC. (válido para los tres casos de uso). 2. El servidor se conecta al servidor NAC o al firewall y analiza los registros necesarios. 3. El sistema muestra la información al usuario.
Escenarios alternativos	1. El sistema no puede analizar los ficheros. 2. El servidor muestra mensaje de error y devuelve al usuario a la página de monitorización.

Caso de uso / ID	**Inicia captura de paquetes / CU007**
Descripción	El usuario desea analizar tráfico de autenticación en tiempo real. El sistema se conecta al servidor NAC y realiza una captura de paquetes.
Ámbito	Troubleshooting
Actor	Administrador/a de red

Requisitos	El sistema debe tener acceso al servidor NAC.
Escenario principal de éxito	1. El usuario introduce el número de segundos que durará la captura. 2. El servidor se conecta al servidor NAC e inicia la captura de paquetes. 3. Pasado el tiempo indicado por el usuario el servidor detiene la captura. 4. El sistema almacena el resultado en un fichero *pcap*. 5. El usuario descarga el fichero en su equipo.
Escenarios alternativos	1. El sistema no puede conectarse al servidor NAC. 2. El servidor muestra mensaje de error y devuelve al usuario a la página captura de paquetes.

Caso de uso / ID	**Descarga logs / CU008**
Descripción	El usuario selecciona alguno de los logs disponibles para descargar. El servidor se conecta al dispositivo oportuno y lo descarga en el equipo del cliente.
Ámbito	Troubleshooting
Actor	Administrador/a de red
Requisitos	El servidor debe disponer de acceso a los dispositivos del sistema NAC para descargar los ficheros.
Escenario principal de éxito	1. El usuario selecciona el fichero log que desea descargar. 2. El servidor se conecta al dispositivo adecuado. 3. El usuario descarga el fichero en su equipo.
Escenarios alternativos	1. El fichero no existe en el dispositivo de destino. 2. El servidor no se puede conectar al dispositivo. 3. Para todos los escenarios anteriores, el servidor muestra mensaje de error al usuario y lo devuelve a la página de descarga de logs.

Con todo ello, se da por finalizado el diseño de la aplicación y se podría comenzar con la implementación, la cual deberá llevar a la práctica todas las características y modo de operar aquí definido.

4. Implementación de los productos

4.1. Implementación de una red segura basada en perfiles de acceso

Basándonos en la topología lógica a alto nivel definida en la fase de diseño (Figura 8), se puede comprobar que un sistema NAC requiere de diferentes elementos interconectados entre sí para poder llevar a cabo su objetivo. Es por ello que la comunicación entre todos resulta crucial, por lo que la primera acción que se ha llevado a cabo en la fase de implementación ha sido habilitar la misma entre todos ellos.

En este sentido, las consideraciones en cuanto a direccionamiento y necesidades de configuración en los dispositivos que forman parte de la topología de red son las siguientes:

- Actualmente, el Switch de usuarios dispone de tres VLAN de departamentos y una de Impresoras. A su vez, este Switch conecta directamente con el Firewall y el punto de acceso. Por tanto, a través de dichos enlaces se deberá permitir la comunicación de todas las VLAN (enlaces troncales).

- Los puertos del Switch destinados para la conexión de usuarios o impresoras serán las **interfaces de la 1 a la 20**, por lo que dicho rango requerirá **autenticación 802.1x o MAB**. Por otro lado, los enlaces troncales serán el 23 (hacia el punto de acceso) y el 24 (hacia el Firewall), mientras que las interfaces 21-22 quedarán reservadas para futuros enlaces o pruebas.

- El firewall dispone de tres interfaces, una para usuarios, otra para servidores y otra para la salida a Internet. La interfaz que conecta con el Switch de usuarios deberá ser capaz de enrutar todas las VLANs de estos, por lo que resulta necesario crear una subinterfaz para cada una de ellas, que a su vez actuará como puerta de enlace para los dispositivos de cada subred. Por su contra, las dos interfaces restantes no requieren subinterfaces, ya que no transportarán VLANs.

- La configuración del direccionamiento en los equipos de los usuarios se realizará mediante **DHCP**, siendo el **firewall** quien actúe como tal. Por su contra, los servidores serán configurados con IP estática.

Por tanto, los dispositivos de red deberán ser configurados de tal manera que cumplan con el siguiente esquema de comunicación.

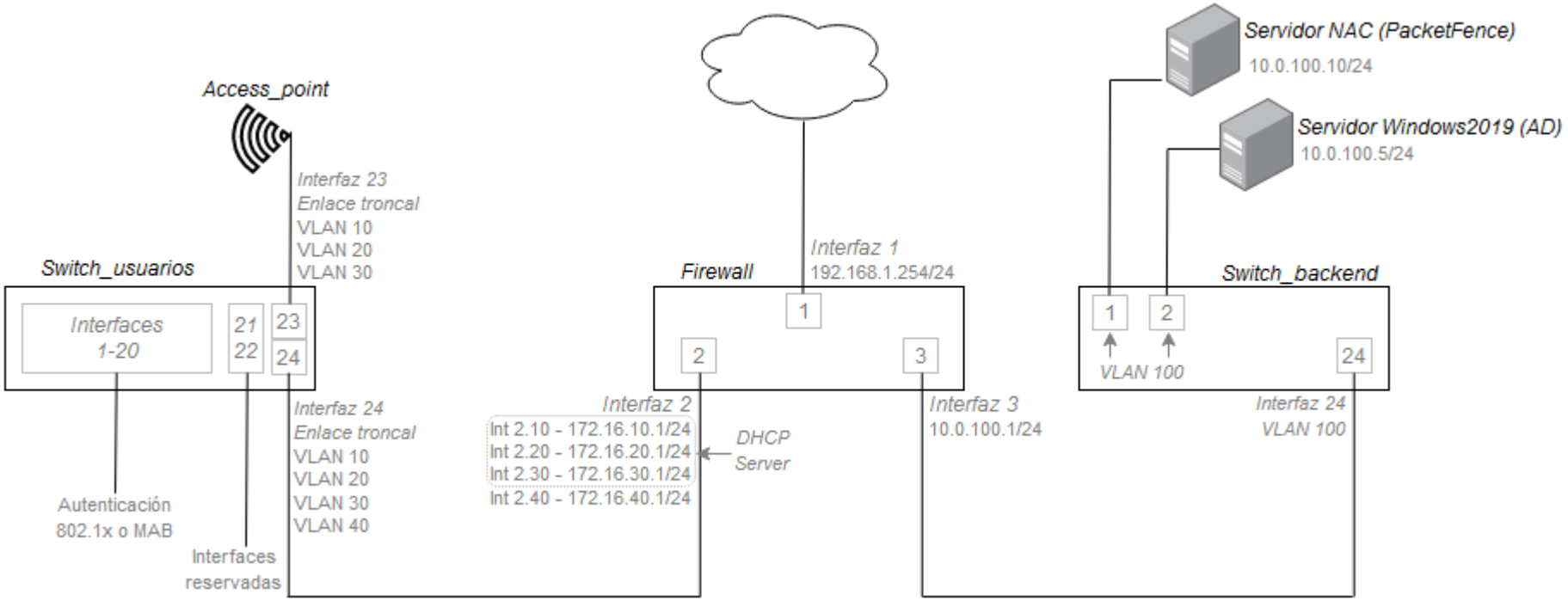


Figura 15: Esquema de direccionamiento de red
Fuente propia (realizada con draw.io)

Gracias a ello se logra la comunicación entre todos los elementos del sistema NAC, lo cual, conjuntamente con la configuración aplicada en cada uno de los dispositivos y servidores, logramos el objetivo de habilitar el acceso a la red basado en perfiles de seguridad.

Con ello, la fase de implementación del producto se ha llevado a cabo aplicando la siguiente metodología en cada uno de los dispositivos…

4.1.1. Implementación del firewall (pfSense)

Tanto en el diseño de la red como en el esquema de direccionamiento, se ha designado al firewall como **enrutador** de las diferentes VLAN de la topología. Además, en cuanto a NAC se refiere, también es el encargado de aplicar las reglas de **filtrado de tráfico en capa 3** para cada uno de los perfiles de acceso. Por tanto, tendremos que tener cuenta todo ello durante su configuración e integración en el proyecto que estamos desarrollando. Para lograrlo, se han llevado a cabo las siguientes acciones durante el proceso de implementación de este dispositivo: [31]

- Se han configurado las interfaces de tal manera que comuniquen, y a la vez actúen como puerta de enlace para cada uno de los perfiles de acceso. Para lograr este aspecto, se ha realizado la siguiente configuración.

    - La interfaz 1, denominada en la configuración del dispositivo como **WAN**, se ha habilitado y configurado con la IP 192.168.1.5/24.
    - La interfaz 3, denominada como **LAN** en el firewall, y que conecta con el switch de backend, se ha habilitado y configurado con la IP 10.0.100.1/24.
    - La interfaz 2, nombrada como **USR** en el firewall, y que conecta con el switch de usuarios, debe transportar 4 segmentos de red, por lo que resulta necesario configurar una **VLAN** para cada una de ellos y posteriormente asociarlos con la interfaz física. Los datos configurados para la implementación de este aspecto son los siguientes:
        - **usr1.10** – 172.16.10.1/24
        - **usr1.20** – 172.16.20.1/24
        - **usr1.30** – 172.16.30.1/24
        - **usr1.40** – 172.16.40.1/24

- Además, al crear cada interfaz se puede habilitar un **servidor DHCP** para el rango de red de la misma. En cuanto al sistema NAC se refiere, este aspecto resulta relevante, ya que, de esta manera, los usuarios autenticados podrán recibir automáticamente el direccionamiento IP correspondiente al rango de red del perfil de acceso que se le ha asignado. Por tanto, se habilita el servidor DHCP, pero tan solo sobre las VLAN de usuarios, es decir, la 10,20 y 30. Los dispositivos del resto de VLAN (impresoras y servidores) serán configurados de manera estática, ya que se considera una buena práctica para este tipo de equipos.

- Por último, para terminar de integrar el firewall en el sistema NAC que estamos implementando, faltaría por crear las reglas de filtrado en capa 3 para cada uno de los perfiles de acceso. Estas reglas fueron definidas en la etapa de

diseño de la solución, por lo que se han aplicado aquellas que se han especificado en el apartado 3.1.4.

Resumiendo, con los cambios que se han llevado a cabo se ha logrado que el firewall gestione el enrutamiento de los diferentes perfiles de acceso, facilite el direccionamiento IP a los usuarios que se autentiquen con éxito, y realice el filtrado de tráfico en capa 3 correspondiente a cada perfil, por lo que las funciones que se definieron para este dispositivo dentro del sistema NAC que estamos implementando se llevarían a cabo.

Su instalación y configuración en detalle puede ser consultada en el Anexo 1.

4.1.2. Implementación del servidor NAC (PacketFence)

El servidor NAC puede ser considerado como el núcleo del entorno seguro que estamos implementado, ya que, por un lado, actúa como servidor **Radius**, y, por otro, **aplica los perfiles de acceso** sobre los usuarios, siendo estas dos funciones imprescindibles para que el resultado del proyecto sea el deseado.

Debido a los motivos expuesto en la fase de diseño, la solución seleccionada para llevar a cabo este rol es el software ***PacketFence***, el cual requiere una configuración meticulosa para lograr implementarlo en la topología que se ha diseñado para este proyecto. Las acciones que se han llevado a cabo para ello son las siguientes: [32]

- Configurarle de manera estática la IP 10.0.100.10/24 y conectarlo a la VLAN de servidores.

- Integrar el servidor en el dominio del directorio activo y posteriormente configurarlo como fuente de autenticación. Gracias a ello, el servidor Radius podrá llevar a cabo la autenticación basada en AD.

- Crear los perfiles de conectividad basados en 802.1x y MAB. De esta manera, el servidor NAC podrá operar con ambos métodos, tal y como se definió en la fase de diseño.

- El siguiente paso que se ha llevado a cabo ha sido crear los roles necesarios, es decir, uno para cada perfil de acceso. Este paso es bastante sencillo y tan solo conlleva asignar un nombre al rol y definir el máximo de conexiones simultáneas que se permitirán a un mismo usuario en el mismo rol, siendo para este caso, 1.

- Crear las reglas de acceso, que en PacketFense son denominadas como "*Authentication Rules*", en las cuales se definen las condiciones que se deben dar para asignar a un usuario un determinado perfil. Además, en la definición de la misma también hay que indicar el tipo de autenticación sobre la cual se aplicará. Como el proyecto se basa en 4 perfiles de acceso diferentes, se han aplicado las siguientes reglas:

- *Tipo de autenticación: LDAP → Si un usuario pertenece al grupo del directorio activo RRHH, se le asigna el rol denominado RRHH.*
- *Tipo de autenticación: LDAP → Si un usuario pertenece al grupo del directorio activo Ventas, se le asigna el rol denominado Ventas.*
- *Tipo de autenticación: LDAP → Si un usuario pertenece al grupo del directorio activo Finanzas, se le asigna el rol denominado Finanzas.*
- *Tipo de autenticación: LOCAL → Si un dispositivo tiene la MAC 00-00-00-11-11-11, se le asigna el rol denominado IMPRESORAS.*

De esta manera, cuando el servidor reciba una solicitud de autenticación, analizará las reglas de manera secuencial en el mismo orden en que han sido introducidas, y desde que alguna coincida, le aplica el perfil indicado y no continúa analizando las restantes. Si no coincide con ninguna, se deniega el acceso. También se puede comprobar cómo las 3 primeras reglas se basan en 802.1x, mientras que la última en MAB, destinada para la impresora.

- Por último, se agrega el Switch y el punto de acceso como orígenes de autenticación, y en cada uno de ellos se indica qué VLAN se asignará a cada rol. Es decir, si por ejemplo se recibe una solicitud de autenticación desde el Switch, y el usuario pertenece al rol de RRHH (analizado por las reglas anteriores), se le permite el acceso a la red y se le asigna la VLAN 10.

Gracias a todo lo anterior, el servidor NAC llevará a cabo todas las tareas que se esperan de él en esta implementación de una red segura basada en perfiles de acceso.

Su instalación y configuración en detalle puede ser consultada en el Anexo 2.

4.1.3. Implementación del Switch de usuarios (Enterasys B5G124-24)

En cuanto a NAC se refiere, el objetivo principal del Switch consiste en habilitar la autenticación del usuario o dispositivo en los puertos habilitados para ello, de tal manera que aplicará el perfil de acceso indicado por el servidor Radius para cada usuario en cuestión. Para lograr tanto este objetivo como el plan de direccionamiento indicado anteriormente, se han llevado a cabo las siguientes acciones en el dispositivo: [33]

- Se han configurado las VLAN 10, 20 y 30 para los perfiles de usuarios, mientras que la 40 para las impresoras. Este paso es necesario por dos motivos. Primero, para asignar a cada perfil de acceso una subred independiente, y, segundo, para poder habilitar la VLAN indicada por el servidor NAC en el puerto indicado por el mismo.

- Tras ello, se han configurado las interfaces **23 y 24** como **enlaces troncales**, con el objetivo de poder transportar la comunicación de todas las VLAN hacia el firewall y hacia el punto de acceso. Por un lado, el enlace troncal hacia el firewall es totalmente necesario ya que este lleva a cabo, tanto el enrutamiento, como el filtrado de seguridad de todas las VLAN. Mientras que hacia punto de acceso también lo es, ya que los usuarios inalámbricos también pertenecerán a estas VLAN.

- Continuando, se ha definido el servidor Radius con el que el Switch deberá contactar para llevar a cabo la autenticación de los usuarios. Aunque esta configuración puede variar dependiendo del Switch utilizado, normalmente es necesario configurar los siguientes aspectos:

    - Habilitar el servicio Radius en el Switch.
    - Indicar la IP del servidor (10.0.100.10), el puerto utilizado para ello (1812) y la frase de paso (contraseña) para poder comunicarse.
    - Definir el tipo de acceso sobre el que se aplicará la autenticación, que en este caso será para la conectividad de los usuarios (denominado por el Switch de este proyecto como *network-access all*).

- Además, como el servidor Radius y el Switch están en segmentos de red diferentes, es necesario configurar una IP a este último para poder llevar a cabo la comunicación. Por tanto, se ha configurado en el Switch la IP 172.16.10.2, desde la cual enviará las peticiones de autenticación de usuarios al Radius.

- Por último, se han configurado los puertos que requerirán autenticación para conectarse a la red, los cuales, como se ha establecido anteriormente, corresponden al rango de **interfaces 1-20**. Además, cada puerto deberá permitir dos tipos de autenticación, donde en primer lugar se ejecutará **802.1x**, y en segundo **MAB**. Para lograr todo ello se ha llevado a cabo el siguiente procedimiento:

    - Habilitar 802.1x, EAPOL y MAB de manera global en el Switch.
    - Permitir dos métodos de autenticación en cada puerto, que se ejecutarán en orden, primero 802.1x, y si esta falla, MAB.
    - Permitir la conexión de un solo usuario por puerto.
    - Forzar la autenticación de puerto para el rango de interfaces 1-20.

Con todas las acciones realizadas, el Switch ya estaría integrado y preparado para llevar a cabo sus funciones dentro del sistema NAC, es decir, aplicar a cada usuario el perfil de acceso indicado por el servidor Radius.

La configuración en detalle puede ser consultada en el Anexo 3.

4.1.4. Implementación del punto de acceso (Asus RT-AX59U)

Desde el punto de vista del sistema NAC que se está implementado, la función del punto de acceso coincide con la del Switch, sin embargo, en este caso la configuración es mucho más sencilla, ya que tan solo resulta necesario llevar a cabo las siguientes acciones:

- Configurar un **SSID** para la red Wifi, es decir, su nombre de publicación. Para el proyecto se ha decidido nombrarla como "***TFGDANIELPEREZ*****".**

- Seleccionar como método de seguridad la opción "***WPA2 Enterprise***", la cual, como se ha analizado en el capítulo 2, se basa en 802.1x.

- Indicar la IP del servidor Radius y la frase de paso (contraseña) necesaria para poder llevar a cabo la comunicación.

Con ello, cuando un usuario se conecte vía inalámbrica, el punto de acceso contactará con el servidor Radius para autenticarlo, y, si sus credenciales son correctas, etiquetará toda la comunicación del mismo en la VLAN correspondiente al perfil de acceso asignado. De esta manera, se ha logrado aplicar la funcionalidad que debe presentar este dispositivo dentro del sistema NAC que estamos implementando.

La configuración del punto de acceso puede ser consultada en el Anexo 4.

4.1.5. Implementación del directorio activo (Windows Server 2019)

En cuanto al directorio activo, su función dentro del sistema NAC que estamos implementando consiste en actuar como base de datos de usuarios, de tal manera que el servidor Radius contactará con él cada vez que necesite verificar las características y validación de un usuario. Cabe recordar que el diseño del sistema en este aspecto se basa en, primero, verificar la autenticidad del usuario, para posteriormente comprobar a qué grupo del AD pertenece y así asignarle el perfil de acceso que le corresponda.

Por tanto, las tareas que han sido necesarias realizar para cumplir con este objetivo son:

- Configurarle la IP estática 10.0.100.5/24 y conectarlo a la VLAN de servidores.

- Crear un dominio, el cual se ha denominado como "*TFGdanielperez.es".*

- Crear un servidor DNS, necesario para que las tareas y los registros del dominio no presenten problemas.

- Crear un grupo dentro del AD para cada departamento, es decir, RRHH, Ventas y Finanzas.

- Crear usuarios de prueba y agregarlos a cada uno de los grupos anteriores, para así poder comprobar la asignación de perfiles de acceso en base al grupo al que pertenecen.

Por otro lado, la instalación de Windows 2019 Server y la puesta en marcha de los servicios de directorio activo y DNS son tareas que suelen corresponder a los administradores de sistemas. Es por ello que en el Anexo 5 tan solo se documentará la creación de grupos y usuarios, ya que estos aspectos sí que están relacionados directamente con la implementación del sistema NAC que estoy llevando a cabo en este proyecto.

Con todo ello, se da por finalizada la implementación de la infraestructura de red basada en perfiles de acceso, la cual debería dar como resultado el siguiente comportamiento cuando algún usuario se conecte a la red (ejemplo para un usuario del departamento de Ventas).

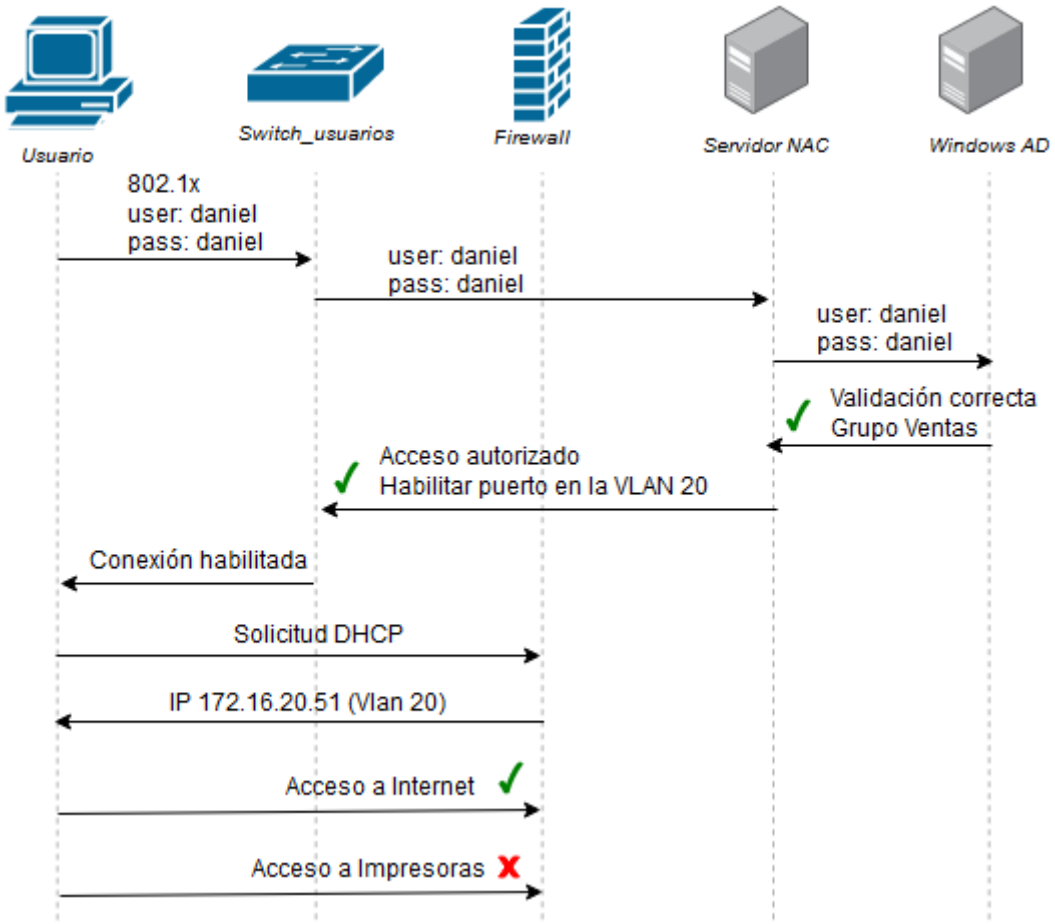


Figura 16: Comportamiento esperado de control de acceso
Fuente propia (realizada con draw.io)

4.2. Implementación de una herramienta de gestión centralizada

La implementación de la herramienta de gestión debe llevarse a cabo teniendo en cuenta las consideraciones y requisitos establecidos en la fase de diseño, de tal manera que incluya y satisfaga todas las funciones y objetivos allí definidos. En este sentido, uno de dichos requisitos consiste en que el código de la aplicación debe basarse en Phyton y HTML, por lo que la primera decisión que se ha tomado en esta fase ha sido seleccionar el IDE más adecuado para operar con ambos lenguajes. Tras analizar varias opciones, he decidido hacer uso del software ***PyCharm Community Edition***, ya que, además de operar con los ya mencionados lenguajes, resulta bastante intuitivo, es gratuito, y dispone de multitud de opciones útiles para el objetivo de este proyecto. Una vez instalado, el desarrollo ha sido el siguiente…

4.2.1. Estructura y modo de operar de la aplicación: *Flask* y *nacApp.py*

Otra de las decisiones tomadas durante el diseño de la aplicación fue que esta debía implementarse sobre ***Flask*.** De manera muy resumida, se trata de un *framework* basado en Phyton que, por un lado, monta un servidor web en el equipo donde es ejecutada, y por otro, interpreta código en este lenguaje para posteriormente integrar el resultado en HTML y así poder mostrarlo vía web. Gracias a ello, podremos ejecutar los *scripts* necesarios para llevar a cabo las funciones sobre los diferentes elementos del sistema NAC, integrando así la aplicación con la infraestructura, para luego mostrar la información al usuario en una interfaz amigable.

En cuanto a su instalación, es un proceso bastante sencillo que tan solo requiere la ejecución del comando "`pip install Flask`" (con el requisito de tener instalado Phyton previamente), mientras que en *PyCharm* ya viene integrado, por lo que tan solo bastará con iniciar un nuevo proyecto basado en Flask. [34]

Con el IDE y **Flask** ya instalado, comienzo el desarrollo de la aplicación, y más concretamente con su estructura. Llegados a este punto resulta necesario mencionar que el modo de operar de Flask se basa en un fichero de configuración en lenguaje Phyton, el cual contiene las rutas a las cuales responderá el servidor web, y, a su vez, el contenido a mostrar y operaciones que se deberán ejecutar en cada una de ellas. A este fichero de configuración lo he denominado como ***nacApp.py***, y será el que contenga el núcleo de la aplicación. Su contenido al completo (con comentarios), se encuentra en el Anexo 6, aunque los aspectos más importantes también se irán mencionando a lo largo del capítulo.

Por otro lado, con lo mencionado hasta ahora se puede deducir que para lograr el objetivo serán necesarios, aparte del fichero *nacApp.py*, ***scripts*** en lenguaje Phyton y **ficheros HTML**. La función, contenido y relaciones entre ellos serán expuestas a lo largo del capítulo, con el fin de facilitar la explicación.

Volviendo a la estructura en sí, agrego las siguientes rutas a la aplicación (fichero *nacApp.py*), es decir, se define el siguiente mapa web, cumpliendo así con el diseño definido en el apartado 3.2.3.

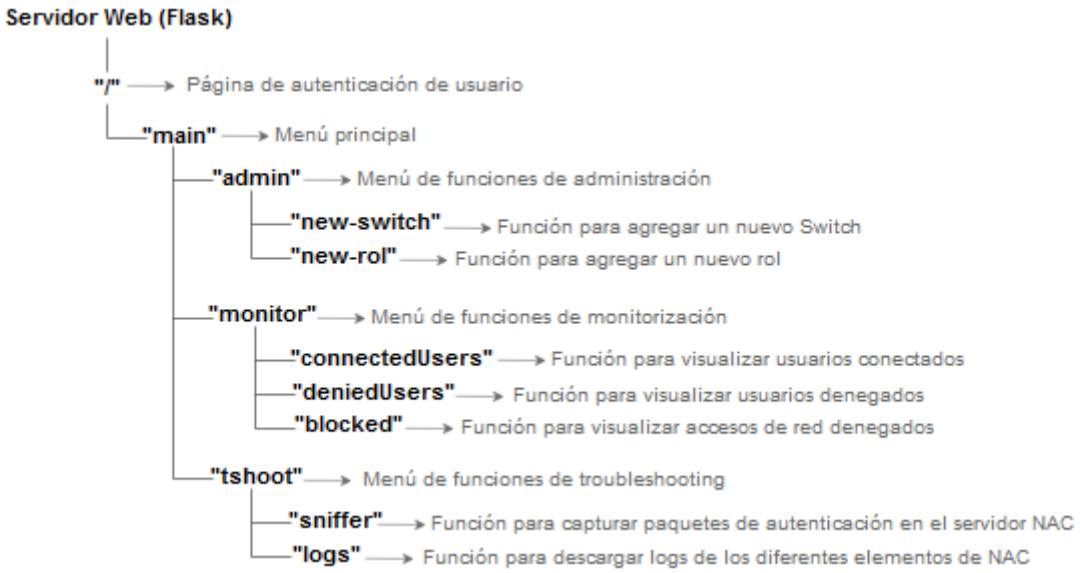


Figura 17: Diagrama de rutas de la aplicación
Fuente propia (realizada con draw.io)

De todas ellas, la página de autenticación y los menús corresponden a los niveles uno y dos de la fase de diseño (Figura 12), por tanto, simplemente son utilizadas como **interfaces de navegación** por el usuario, por lo que su código es muy sencillo. Por su contra, las funciones corresponden al nivel tres, donde resulta necesario ejecutar instrucciones sobre los diferentes elementos del sistema NAC. Además, en este nivel se encuentran la gran mayoría de requisitos funcionales y casos de uso identificados en la fase de diseño, por lo que el desarrollo de estas funciones adquiere aún mayor relevancia.

4.2.2. Nivel 1 de la aplicación: Validación de usuario

Como recién se ha mencionado, el nivel uno corresponde a una interfaz de navegación que únicamente se encarga de la validación del usuario, por lo que no interactúa directamente con la infraestructura NAC. Este hecho facilita su desarrollo, pero aun así hay que tener en cuenta diferentes aspectos para cumplir con los requisitos que exige esta implementación, siendo los siguientes:

- Por un lado, el servidor cargará el formulario cuando el usuario acceda a la ruta "***/***". Sin embargo, hay que tener en cuenta que, si el usuario se valida correctamente se le debe redirigir a la ruta **"/main",** que equivale al nivel 2 (menú principal).
- La validación del usuario fue diseñada como un requisito (**Req00**) y un caso de uso (**CU001**), por lo que se deberá desarrollar de tal manera que cumpla con el comportamiento definido en el modelado de la aplicación web (apartado 3.2.4)

Con ello, para lograr el objetivo ha sido necesario, primero, crear un formulario HTTP para la validación del usuario, denominado **login.html** y, segundo, desarrollar una función Phyton en la ruta "/" del fichero *nacApp.py* que cumpla con el siguiente comportamiento…

1. Cuando se acceda a la ruta, se carga el fichero **login.html.**
2. Cuando el usuario introduce los datos, son capturados por el fichero *nacApp.py,* el cual se encargará de verificarlos.
3. Si la autenticación es correcta, se redirige al usuario a la página **main.html**, de lo contrario, se muestra un mensaje de error y se mantiene en la página de validación.

En cuanto al proceso de autenticación utilizado, se ha aplicado un método muy sencillo, donde el nombre de usuario y contraseña son validados localmente en la propia aplicación (*admin/admin)*. No se han utilizado bases de datos ni autenticación contra el directorio activo porque en esta primera versión de la aplicación he creído conveniente focalizar el código en todo lo relacionado con el sistema NAC.

El código de este apartado de la aplicación puede ser consultado en los anexos 6 (*nacApp.py*) y 7 (*login.html*), mientras que su aspecto visual es el siguiente…

Figura 18: Página de inicio de la aplicación. Autenticación de usuario

4.2.3. Nivel 2 de la aplicación: Menú principal

El menú principal únicamente es accesible cuando el usuario se ha validado con éxito en el nivel uno. Cuando esto sucede, se carga la ruta ***"/main***" definida en *nacApp.py*, la cual simplemente contiene el código Phyton necesario para redirigir al usuario al fichero ***"main.html"***, creado para tal propósito y que a su vez contiene los enlaces para los submenús de las funciones de administración, monitorización y t*roubleshooting*.

Está sección de la aplicación no contiene control de errores ni ningún tipo de dato que se deba verificar, por lo que su código es muy sencillo y no requiere

mayor explicación, aún así puede ser consultado en los anexos 6 (*nacApp.py*) y 7 (*main.html*). Su aspecto visual es el siguiente

Figura 19: Menú principal de la aplicación

4.2.4. Nivel 3 de la aplicación: Funciones

Entrando ya en la parte importante de la aplicación, comienzo con el desarrollo de las funciones, que corresponderían al nivel 3 definido en la fase de diseño. En este caso, todas las opciones deberán ejecutar determinadas acciones en los dispositivos que forman parte del sistema NAC (servidor NAC, AD, Switch…), por lo que resulta imprescindible desarrollar ***scripts*** en phyton para lograr el objetivo y **plantillas html** para mostrar el resultado final. Además, algunas de estas funciones también requieren datos introducidos por el usuario, por lo que, para estos casos, también resulta necesario crear **formularios html** para poder capturar dicha información.

Todas estas funciones serán explicadas a continuación de manera individual, sin embargo, antes conviene conocer cómo interactúan todos estos ficheros entre sí, con el fin de lograr una mayor comprensión del modo de operar de la aplicación…

Para las funciones en las que intervienen formularios, el modo de operar es el siguiente…

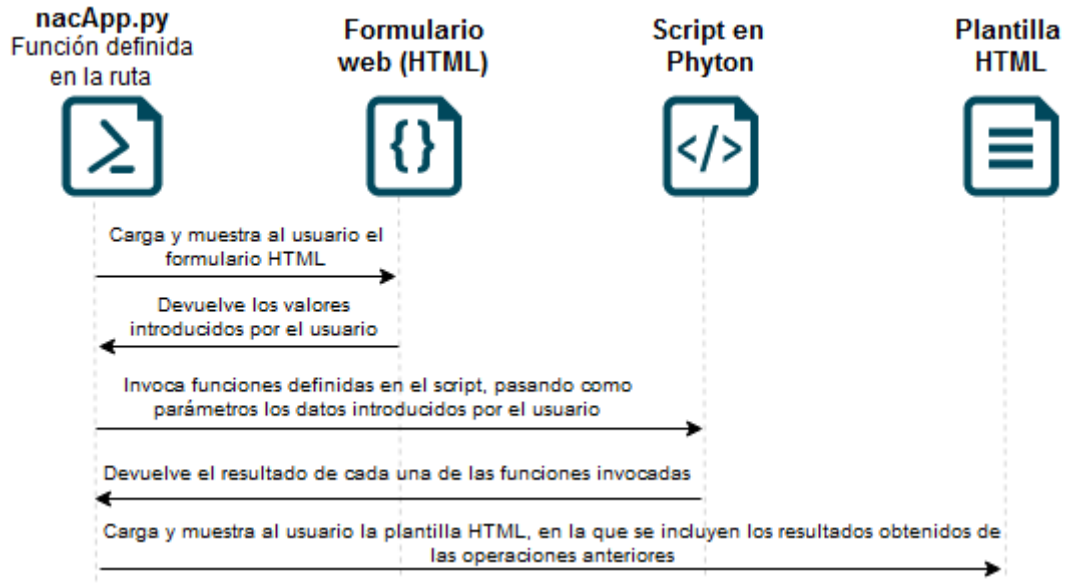


Figura 20: Interacción entre ficheros al ejecutar una operación
Fuente propia (realizada con draw.io)

Mientras que, si no intervienen formularios, se omiten las dos primeras acciones y se invocan las funciones de los scripts sin necesidad de parámetros adicionales, para luego mostrar el resultado obtenido en la plantilla HTML.

Cabe mencionar que un elemento común en todos los scripts que se analizarán es que requieren de algún método de conexión por **SSH** a los dispositivos, para lo cual se ha utilizado la librería de Phyton denominada **paramiko** [35]. El resto de librerías utilizadas se irán mencionando a lo largo del análisis.

Por último, es importante resaltar que esta parte de la aplicación resulta clave para el proyecto, ya que gracias a la misma se logran dos objetivos prioritarios. Por un lado, lograr que esta herramienta de gestión sea **parte activa de la infraestructura NAC**, y por otro lado, la **automatización de tareas**, eliminando así carga administrativa a los administradores/as de red.

4.2.4.1 Funciones de administración: Agregar un nuevo Switch

El objetivo de esta función consiste en configurar todos los elementos necesarios para integrar un nuevo Switch en la infraestructura. Para ello, será necesario realizar las siguientes acciones; primero, **configurar el nuevo Switch**, y segundo, **integrarlo en el servidor NAC**.

Además, responde a un requisito funcional (**Req02**) y a un caso de uso (**CU003**), por lo que se ha desarrollado cumpliendo con las características definidas para los mismos en la fase de diseño.

Para su implementación, el primer paso que se ha llevado a cabo ha sido desarrollar el código de la ruta ***"/admin/new-switch"***, desde la cual se

gestionará la función, así como los ficheros necesarios para lograr el objetivo, siendo los siguientes:

nacApp.py	Ruta: ***/admin/new-switch*** Función de la ruta: ***newSwitch()***
Formulario HTML	***new_switch.html***
Script Phyton	***add_new_switch.py*** Funciones: *new_switch(), add_switch_to_nac()*
Plantilla HTML	***new_switch_result.html***

El funcionamiento de la aplicación sería el siguiente:

1.- Cuando un usuario accede a la ruta */admin/new-switch* se mostrará el formulario ***new_switch.html***, en el cual se le solicitará la IP del nuevo Switch.
2.- Una vez introducida, es capturada por la función ***newSwitch()*** de *nacApp.py*, donde se verificará que tiene el formato correcto, y, de ser así, será utilizada como parámetro para llamar a las funciones del script ***add_new_switch.py.***
3.- Ambas funciones realizarán los cambios necesarios en los dispositivos del entorno NAC y devolverá el resultado nuevamente al fichero *nacApp.py.*
4.- Los resultados son mostrados al usuario en la plantilla HTML.

De todo este proceso, lo que realmente interesa para este proyecto es el paso 3, donde se configuran automáticamente los elementos de la infraestructura. Para ser más exactos, las tareas que se han tenido que desarrollar para lograr el objetivo son las siguientes:

- **Configurar el nuevo Switch** (función *new_switch()*): Establecer la conexión por SSH con la IP indicada por el usuario → Crear las VLAN de todos los perfiles de acceso y permitirlas en los enlaces troncales → Definir el servidor Radius → Aplicar la autenticación 802.1x y MAB en los puertos destinados a usuarios.

- **Intregrar el Switch en el servidor NAC** (función *add_switch_to_nac()*): Acceder vía SSH al servidor NAC → Editar el fichero "*switches.conf*" y agregar las líneas necesarias. → Reiniciar el servicio Radius.

Por último, un dato curioso que me ocurrió mientras desarrollaba la función para configurar el Switch, es que entre comando y comando he tenido que aplicar una pausa de 0.5 segundos (*time.sleep(0.5)*). Esto ha sido necesario ya que de lo contrario el Switch no era capaz de procesar todos los comandos a la velocidad que se los enviaba el Script.

Si desea ser consultado, el código de estas funciones se puede encontrar en el Anexo 8.

4.2.4.2 Funciones de administración: Crear un nuevo Rol

El objetivo de esta opción de administración consiste en configurar todos los elementos necesarios para agregar un nuevo perfil de acceso en el sistema NAC, lo cual implica realizar configuraciones en el **Switch**, **Firewall**, **Directorio Activo**, y **servidor NAC**. Para su implementación se ha desarrollado el código necesario en la ruta *"/admin/new-rol"*, así como los ficheros necesarios para lograr el objetivo, siendo los siguientes:

nacApp.py	Ruta: ***/admin/new-rol*** Función de la ruta: ***newRol()***
Formulario HTML	***new_rol.html***
Script Phyton	***add_new_rol.py*** Funciones: *switchConf(), firewallConf, adConf, nacRolConf()*
Plantilla HTML	***new_rol_result.html***

Los cuales interactúan entre sí de la siguiente manera:

1.- Cuando un usuario acceda a la ruta */admin/new-rol",* se mostrará el formulario ***new_rol.html***, el cual le solicitará un nombre para el nuevo perfil de acceso, el id de vlan, y el rango de red.
2.- Tras ello, los datos son capturados y verificados por el fichero *nacApp.py*, que los utilizará para llamar a las funciones definidas en el script ***add_new_rol.py*.**
3.- Estas funciones serán las encargadas de interactuar con los diferentes elementos de la infraestructura para ejecutar las acciones necesarias. Tras ello, devuelve los resultados obtenidos al fichero *nacApp.py*.
4.- Los resultados son mostrados en la plantilla HTML.

Nuevamente, la interacción con los elementos de la infraestructura se lleva a cabo en el paso 3. En este caso, para lograr los objetivos ha sido necesario desarrollar el script para que lleve a cabo las siguientes tareas…

- **Switch** (Función *switchConf()):* Establecer la conexión vía SSH con el Switch → Crear la VLAN obtenida por parámetro → Propagar la VLAN por los enlaces troncales.

- **Firewall** (Función *firewallConf()*): Establecer la conexión vía SSH con el firewall → Crear una nueva interfaz que actúe como puerta de enlace para la nueva subred → Crear un rango en el servidor DHCP para la nueva subred.

  Cabe mencionar que para crear la puerta de enlace se ha hecho uso del comando ***ifconfig*** de Linux (con los parámetros adecuados), mientras que para el DHCP se han agregado las líneas necesarias al fichero "***dhcpd.conf***", ya que en el mismo se incluye la configuración del servidor DHCP.

- **Directorio Activo** (Función *adConf()*): Establecer conexión con el directorio activo y crear un nuevo grupo. Para lograr este objetivo ha sido necesario hacer uso de la librería de Phyton ***ldap3*.**

- **Servidor NAC** (Función nacRolConf()): Establecer conexión vía SSH con el servidor NAC → Editar y agregar las líneas necesarias en el fichero *"**roles.conf**"*.

Esta función resulta especialmente interesante, ya que interactúa con todos los elementos de la infraestructura. Además, gracias a las acciones realizadas se da respuesta al requisito funcional **Req01** y al caso de uso **CU002**, ya que cumple con las características definidas para ello, especialmente aquellas presentes en el modelado de la aplicación web (apartado 3.2.4).

En caso de éxito, el resultado mostrado por la aplicación sería el siguiente

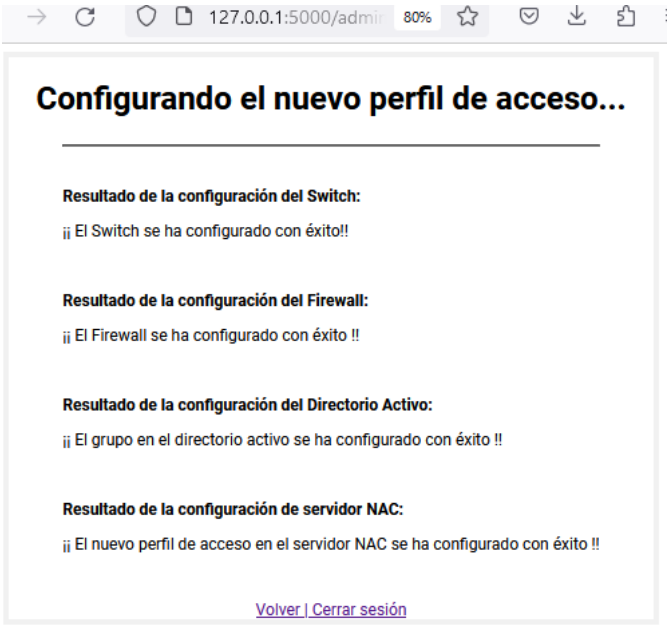


Figura 21: Resultado tras crear un nuevo Rol

Por último, todo el código relativo a esta función puede ser consultado en el Anexo 9.

4.2.4.3 Funciones de monitorización: Ver usuarios conectados, Ver usuarios denegados y Ver accesos bloqueados.

El objetivo de estas funciones consiste en obtener información del estado actual de usuarios y accesos en capa 3, con el fin de poder monitorizar la seguridad del entorno NAC. En este caso, se analizarán todas de manera conjunta, ya que todas realizan las mismas tareas. Además, no son necesarios formularios, por lo que la ejecución de los *scripts* no requiere parámetros adicionales. Las rutas y ficheros creados para tal propósito son los siguientes:

	Ver accesos bloqueados	Ver usuarios conectados	Ver usuarios denegados
nacApp.py	Ruta: ***/monitor/blocked*** Función de la ruta: ***blocked()***	Ruta: ***/monitor/cUsers*** Función de la ruta: **cUsers*()***	Ruta: ***/monitor/dUsers*** Función de la ruta: ***dUsers()***

Script Phyton	***blocked_access.py*** Función: *blockedAccess()*	***users.py*** Función: *connectedUsers()*	***users.py*** Función: *deniedUsers()*
Plantilla HTML	***blocked_access_result.html***	***cUsers_result.html***	***dUsers_result.html***

Donde en todos los casos el funcionamiento sigue la siguiente lógica:

1.- El usuario accede a la ruta definida en el fichero *nacApp.py,* el cual llama a la función del Script asociado a la ruta en cuestión.
2.- Tras la ejecución del Script, este devuelve el resultado a *nacApp.py*.
3.- Se muestra el resultado al usuario en la plantilla HTML.

En cuanto a los accesos bloqueados en capa 3, ha sido necesario obtenerlos desde el fichero "***filter.log***" ubicado en el firewall, por lo que en primer lugar se establece la conexión SSH con el mismo, para posteriormente obtener el contenido del fichero. El resultado será una lista extensa de filas de texto. El problema que me he encontrado ha sido que al ejecutar el script y obtener la información, se almacenaba todo en una sola línea, imposibilitando la legibilidad del mismo. Para solucionarlo, lo que he hecho es, en el script, incluir un salto de línea cada vez que se encontrara el carácter *"\n*", y en la plantilla *html*, leer el fichero línea a línea, imprimiendo cada una de ellas.

Otro dato que puede resultar curioso al analizar el *script blocked_access.py* son los parámetros que se envían al firewall para leer el fichero. De ellos, "***filterparser.php***" [31] hace referencia a un script propio de *pfsense* que hace que las entradas sean más reducidas y legibles, eliminando algunos campos que no tienen mucha trascendencia a la hora de analizar los bloqueos. Mientras que ***"grep -wv ff02***" y "***grep block***" lo que harán será aplicar otros filtros sobre el resultado. Para el primero de los casos, elimina todas las filas que contengan el texto *ff02*, ya que hacen referencia a direcciones IPv6, cuando las que nos interesan son IPv4, mientras que el segundo de los casos tan solo mostrará las líneas que contengan el texto "*block*", que para el caso que tratamos son las que hacen referencia a accesos bloqueados, siendo estas las que nos interesan.

Por otro lado, en cuanto a los usuarios conectados y denegados ha sido necesario analizar la base de datos utilizada por el servidor NAC, con nombre **pf**, ya que es ahí donde se registran los accesos. Para obtener la información que necesitamos se realizará una consulta SQL, filtrando los datos en base al contenido del campo **status**, de la tabla **node**. Si dicho campo contiene el valor "*reg*" significa que los datos de esa fila identifican a un equipo/usuario conectado. Por su contra, si su valor es "*unreg*", se tratará de un acceso denegado (o en proceso). Además, el resultado también mostrará otra información interesante, como puede ser el nombre de equipo, mac, y usuario, los cuales se extraen de los campos *computername, mac* y *user_agent* de la misma tabla. Para poder llevar a cabo todo ello ha sido necesario hacer uso de la librería **mysql.connector** de Phyton. [37]

Por último, esta implementación da respuesta a los requisitos funcionales **Req03, Req04 y Req05,** así como a los casos de uso **CU004, CU005** y **CU006**. [37]

El código completo de todas estas funciones y ficheros puede ser consultado en los Anexos 10 y 11.

4.2.4.4. Funciones de troubleshooting: Captura de paquetes de autenticación

Las funciones de troubleshooting tienen el objetivo de ayudar al administrador/a a detectar y localizar incidencias para posteriormente poder resolverlas. Una herramienta de mucha utilidad para el caso que tratamos es la de poder realizar una captura en tiempo real de paquetes de autenticación en el servidor NAC. Es decir, obtener todo el tráfico que atraviese dicho servidor que tenga que ver con intentos de acceso de usuarios. La implementación de esta función se ha llevado a cabo en la ruta *"/tshoot/sniffer",* y cabe mencionar que en este caso no resulta necesaria la plantilla HTML para mostrar el resultado final, ya que el mismo será la descarga de la captura en el equipo del usuario. Por tanto, tan solo ha sido necesario crear los siguientes ficheros:

nacApp.py	Ruta: ***/tshoot/sniffer*** Función de la ruta: ***sniffer()***
Formulario HTML	***sniffer.html***
Script Phyton	***sniffer.py*** Funciones: *sniffer()*

El modo de operar que se ha implementado coincide el definido durante el diseño de la aplicación, más concretamente en el apartado de modelado de la arquitectura web. De esta manera, también hacemos cumplir el requisito **Req06** y el caso de uso **CU007.** El comportamiento de esta función en cuestión es el siguiente:

1.- Cuando un usuario acceda a la ruta */tshoot/sniffer,* se mostrará el fichero ***snnifer.html*****,** el cual contiene un formulario que solicitará al usuario introducir el número de segundos que desea que dure la captura.

2.- Una vez introducido, el dato es capturado y verificado por *nacApp.py,* y, si es correcto, solicita al usuario la ubicación local donde desea almacenar la captura. Para esta última acción ha sido necesario hacer uso de la librería ***tkinder*** de Phyton.

3.-Tras ello, se llama al script ***sniffer.py***, pasando los segundos y la ubicación como parámetro, para que este ejecute la captura durante el tiempo indicado por el usuario, y posteriormente descargue el fichero en la ubicación recibida. Para la captura se ha hecho uso del comando ***tcpdump*** (con sus correspondientes parámetros)

4.- Una vez finalizado, se indicará al usuario si la descarga fue correcta o se produjo algún error, y, para el primero de los casos, se le dará la opción de abrirlo.

En lo relativo a la interacción con la infraestructura de red, cabe mencionar que la captura de paquetes se realiza directamente en el servidor NAC, para lo cual el código desarrollado en el script *sniffer.py* lo primero que hará será establecer conexión SSH con el mismo, para posteriormente ejecutar el comando ***tcpdump*** (entre otras acciones).

Un dato curioso, pero también lógico, es que ha sido necesario detener la ejecución del *script* durante el tiempo indicado por el usuario justo después de comenzar la captura de paquetes *(time.sleep(seconds))*. Esto es necesario realizarlo porque, de lo contrario, se seguirían ejecutando sentencias si haber finalizado dicha captura en el servidor NAC, generando el consiguiente error. De resto, se ha aplicado el parámetro "***any port 1812***", que significa que se capturarán paquetes de cualquier origen con puerto de destino 1812, ya que es el utilizado por Radius para la autenticación. Por otro lado, con ***-w*** logramos que la captura se almacene en un fichero, en este caso denominado como *captura_auth.pcap*, mientras que con ***timeout*** indicamos que *tcpdump* se ejecute durante un tiempo determinado, siendo el indicado por el usuario.

El código completo de todas estas funciones y ficheros puede ser consultado en el Anexo 12.

4.2.4.5 Funciones de troubleshooting: Descarga de logs

Una tarea imprescindible en cuanto a labores de *troubleshooting* se refiere es el análisis de logs. En este sentido, el entorno NAC que se ha implementado está compuesto por diferentes dispositivos, donde cada uno de ellos generará sus propios ficheros de registro. Por tanto, una manera de facilitar la tarea a los administradores/as de red consiste en poder acceder a todos ellos de manera centralizada, siendo esto precisamente lo que se ha pretendido con esta función de la aplicación. De esta manera, también damos respuesta al requisito funcional **Req07** y al caso de uso **CU008** definidos durante el diseño de la aplicación. Su implementación se ha llevado a cabo en la ruta *"/tshoot/logs*", y consta de las siguientes funciones y ficheros:

nacApp.py	Ruta: ***/tshoot/logs*** Función de la ruta: ***getlogs()***
Formulario HTML	***logs_download.html***
Script Phyton	***logs_download.py*** Funciones: *getLogs()*

Cabe mencionar que, en este caso, el formulario *html* no incluye campos a rellenar por el usuario, sino una lista desplegable que mostrará todos los logs disponibles a descargar, y donde una variable obtendrá un valor u otro dependiendo de la opción seleccionada.

Con ello, al acceder a la ruta *"/tshoot/logs*" se mostrará el fichero ***logs_download.html,*** el cual contiene la ya mencionada lista desplegable y un botón denominado "*Descargar*". De esta manera, cuando el usuario selecciona el fichero que desea y pulsa sobre el botón, *nacApp.py* realiza dos acciones, primero, captura el valor de la variable del log seleccionado, y segundo, muestra

una ventana emergente, solicitando al usuario el directorio local donde desea almacenar el archivo (librería ***tkinder***).

Tras ello, con ambos datos se llama al script ***logs_download.py***, y más concretamente a su función *getLogs(opción, directorio)*, pasando como parámetros los datos recopilados anteriormente. El código de este *script* lo que hará será analizar el valor del parámetro "*opción*", y en base al mismo, conectarse al dispositivo adecuado por SSH y descargar el fichero seleccionado por el usuario en la carpeta indicada por el mismo, es decir, la recibida en el parámetro "*directorio*".

Por último, el código completo de todas estas funciones y ficheros puede ser consultado en el Anexo 13.

Con ello, se da por finalizado el desarrollo de la aplicación, sobre la cual se realizarán las pruebas necesarias y se propondrán las mejoras oportunas para posteriores versiones.

5. Pruebas

Una vez implementados los productos, las pruebas que se han realizado para verificar su funcionamiento han sido las siguientes…

5.1. Pruebas de conectividad

Las pruebas de conectividad se han llevado a cabo haciendo uso de dos equipos con Windows 10, donde solo uno de ellos pertenece al dominio ***TFGdanielperez***. A su vez, en el directorio activo se han creado los grupos correspondientes a los departamentos de la compañía y un usuario en cada uno de ellos. Las comprobaciones han sido las siguientes:

- **Prueba 1:** Conectar el equipo que **no** pertenece al dominio a cualquier interfaz del Switch destinada a usuarios. El resultado esperado debería ser denegar su acceso a la red, ya que no coincide con ninguna de las políticas definidas en el servidor NAC.

  Analizando el tráfico con tcpdump, se pueden observar los paquetes de solicitud y posterior rechazo por parte del servidor…

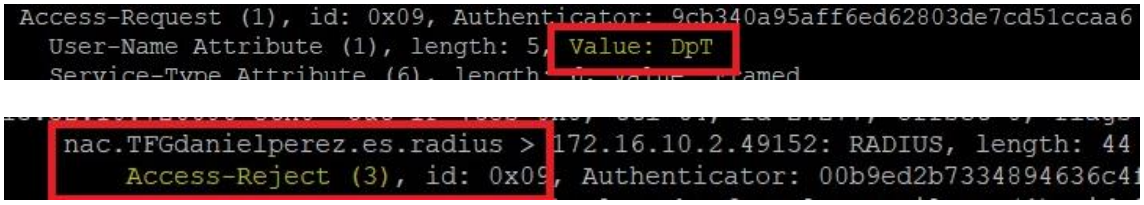


Figura 22: Acceso denegado a la red de equipo desconocido (consola)

Mientras que en la interfaz web del servidor también se pueden visualizar…

Figura 23: Acceso denegado a la red de equipo desconocido (web)

Además, el PC no obtiene los datos de conexión, Por lo que el resultado para esta prueba se puede considerar **satisfactorio**.

- **Prueba 2**: Conectar el equipo que pertenece al dominio e iniciar sesión con un usuario que **no** corresponde a ninguno de los grupos definidos en el

servidor NAC. En este caso, el resultado esperado también debería ser la denegación del acceso.

Tras conectarlo a la red, el resultado en consola es el siguiente…

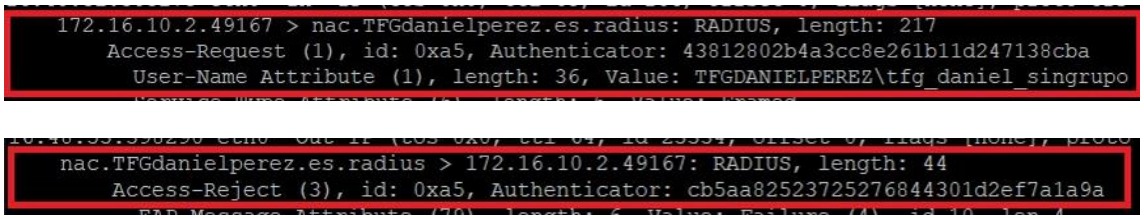


Figura 24: Acceso denegado a la red de usuario sin grupo (consola)

Mientras que en la interfaz web…

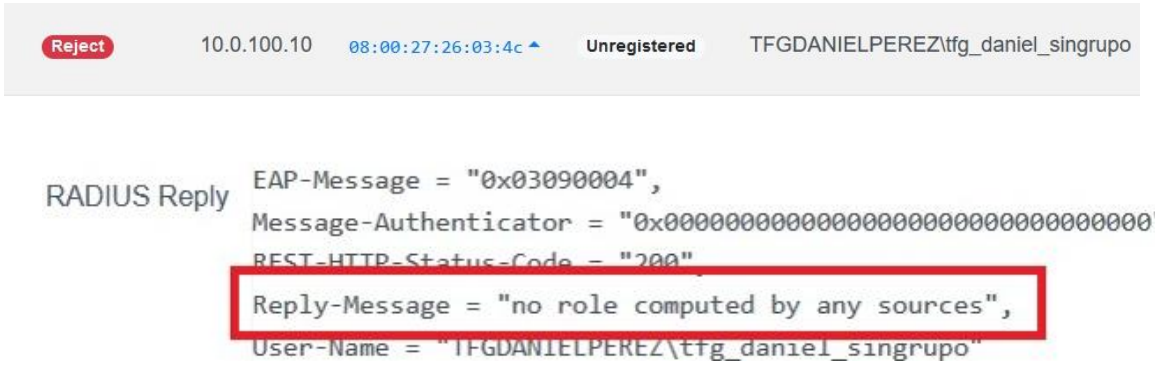


Figura 25: Acceso denegado a la red de usuario sin grupo (web)

Tras ello, evidentemente, el equipo no obtiene datos de red. Por tanto, el resultado para esta prueba se puede considerar **satisfactorio**.

- **Prueba 3**: Conectar el equipo que pertenece al dominio e iniciar sesión con cada uno de los usuarios de los diferentes grupos. En este caso, a cada uno de ellos se le asignó la VLAN correspondiente a su departamento. (Se muestra el resultado para el usuario de RRHH, pero se ha probado con todos con éxito).

Tras conectarlo a la red, el resultado en consola es el siguiente…

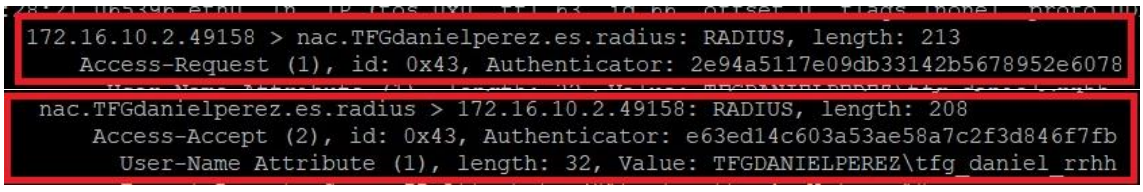


Figura 26: Acceso autorizado a la red de usuario de RRHH (consola)

Mientras que en la interfaz web, aparece registrado y se puede comprobar que se le ha asignado la VLAN10.

Figura 27: Acceso autorizado a la red de usuario de RRHH (web)

Tras ello, el equipo obtiene los datos de red correspondientes a la VLAN10, por lo que el resultado para esta prueba también se puede considerar **satisfactorio**.

5.2. Pruebas con la aplicación de gestión centralizada

En el caso de la aplicación de gestión centralizada se han realizado las siguientes comprobaciones…

- **Nivel 1**: Intentos de validación erróneos e intentos de acceso a rutas sin estar autenticado previamente. En ambos casos, la aplicación redirige al usuario a la página de inicio de sesión.

  El intento de validación erróneo muestra el siguiente resultado…

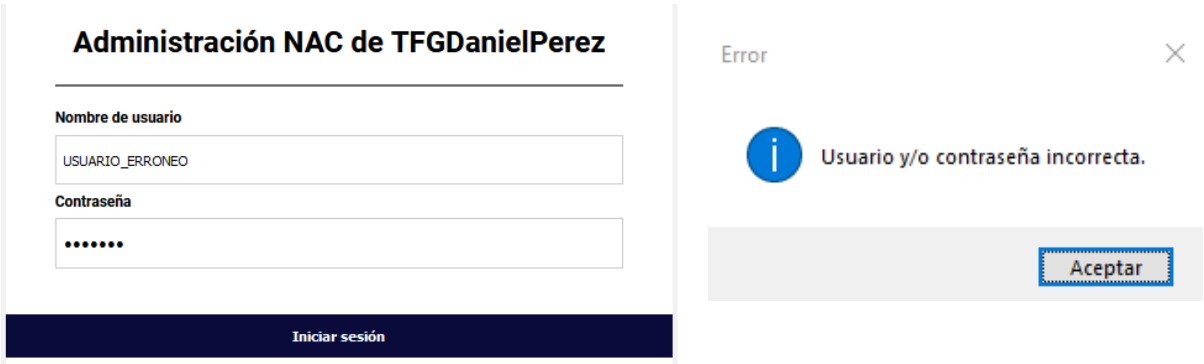


Figura 28: Nivel 1 de la herramienta de gestión

Mientras que el intento de acceso a rutas sin estar validado redirige automáticamente al usuario a la página de validación. Por tanto, estos resultados se pueden considerar como los esperados.

- **Nivel 2**: Iniciar sesión correctamente y navegar por los menús, en cuyo caso no se ha detectado ninguna anomalía. Un ejemplo de navegación a través de los menús es el siguiente, donde el usuario inicia sesión, se le muestra el menú principal y posteriormente accede al submenú de las funciones de administración…

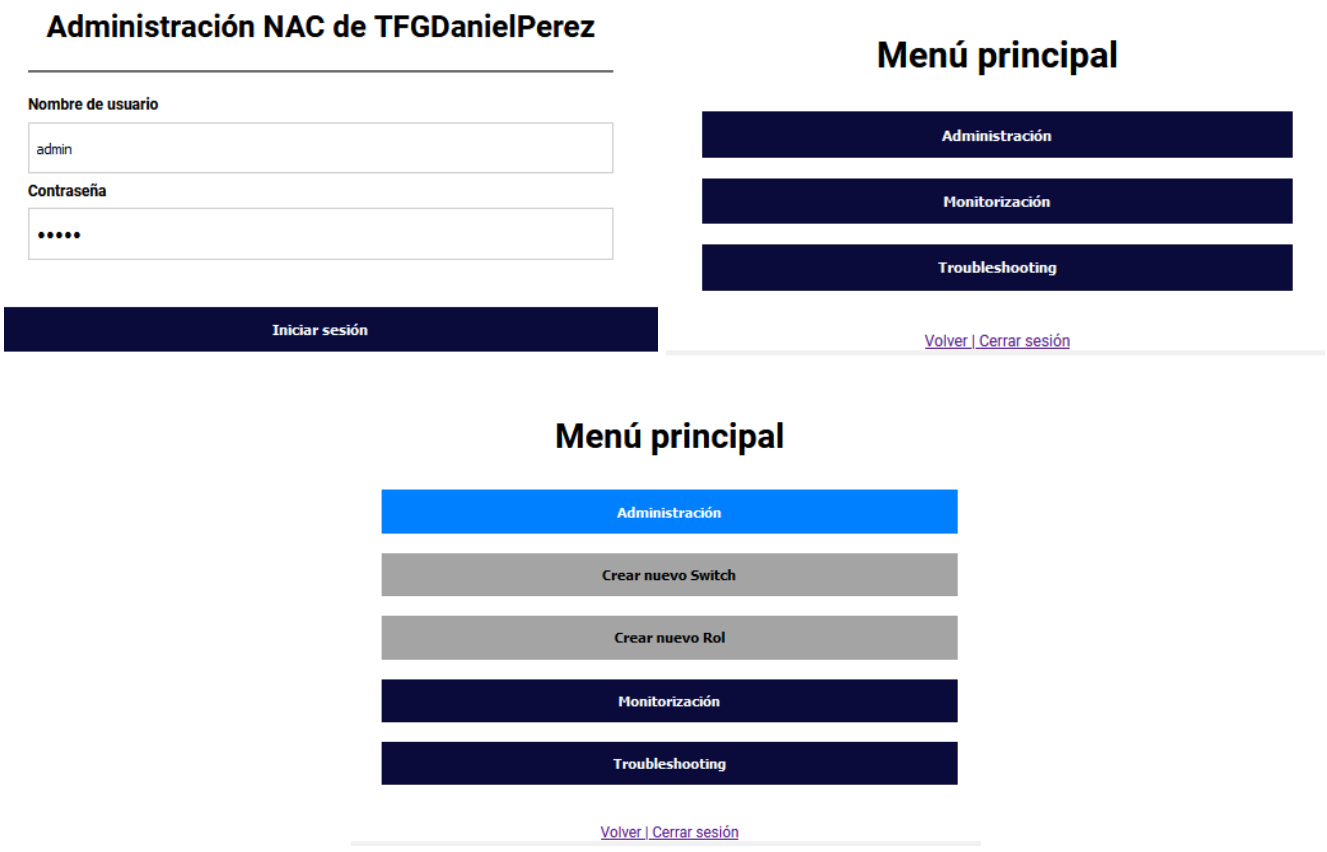


Figura 29: Nivel 2 de la herramienta de gestión

- **Nivel 3**: Ejecutar todas las funciones, tanto generando errores de manera intencionada, como realizando el proceso correctamente. Para el primero de los casos, la aplicación devuelve el error que se ha producido, mientras que, para el segundo, el resultado de la operación que se ha obtenido.

  El detalle de todas estas pruebas es demasiado extenso como para incluirlo en el cuerpo de la memoria, por lo que pueden ser consultadas en el Anexo 14.

6. Conclusiones y trabajos futuros

Una vez finalizado el proyecto y obtenidos los resultados de las pruebas realizadas, la conclusión principal que he obtenido tras estudiar este tipo de sistemas y trabajar con todas las herramientas necesarias para su implementación, es que he explotado una mínima parte del enorme potencial que puede ofrecer NAC. Si bien es cierto que los resultados han sido los esperados y se ha logrado el objetivo de implementar una red basada en perfiles de acceso y administrarla de manera centralizada, se podría haber obtenido un producto más robusto y con mejores características de seguridad. Sin duda, los trabajos futuros irán en esta línea.

En cuanto a su implementación, me ha llamado la atención la cantidad de recursos necesarios para su puesta en marcha. En este sentido, no solo se requieren conocimientos de redes, sino también de sistemas, bases de datos y desarrollo de aplicaciones. En una red tan pequeña como la que he montado puede ser asumible afrontar su puesta en marcha por un solo técnico (aunque a nivel profesional no lo veo adecuado) y lograr todos los objetivos planteados, como ha sido el caso de este proyecto. Sin embargo, en una red corporativa de tamaño medio o grande pienso firmemente que una implementación adecuada de NAC se deberá llevar a cabo mediante la creación de un equipo formado por técnicos de dichas especialidades.

Continuando con los objetivos, en un principio tenía pensado implementar una red **ZTNA** (*Zero Trust Network Access*), la cual, además de llevar a cabo el acceso basado en perfiles, puede analizar vulnerabilidades en los clientes y con ello disponer de multitud de valores más para tomar una decisión sobre si es seguro su acceso. Sin embargo, lograrlo implicaba montar dos nuevos servidores e integrarlos con *packetfence*, uno para el análisis de vulnerabilidades, y otro que actuara como entidad certificadora para el acceso con certificado. Debido al tiempo limitado del proyecto y a la falta de recursos, no he podido llevar a cabo esta implementación.

Por otro lado, otro de los objetivos planteados inicialmente y que estuve a punto de no cumplir fue desarrollar las funciones "*Ver usuarios conectados*" y "*Ver usuarios denegados*". El problema que me encontré fue que no lograba obtener información de la base de datos que los almacenaba, más concretamente el nombre de la *bd*, y la tabla y campos necesarios. Finalmente, pude dar con la misma analizando el fichero de configuración *graph.pm* de *packetfence*, el cual genera gráficas en la interfaz web, para lo cual lanza consultas a la bd.

En cuanto a la planificación, he podido cumplir con los plazos indicados inicialmente en el diagrama de Gantt, aunque bien es cierto que la falta de experiencia en este tipo de entornos y los problemas que he tenido que ir resolviendo han hecho que tenga que dedicarle muchísimo más tiempo del que en un principio tenía pensado para cumplir los hitos, por lo que la conclusión en este aspecto es que **no** ha sido lo suficientemente adecuada. Con los conocimientos que he adquirido, muy probablemente ahora enfocaría el diagrama y plazos de otra manera.

Por tanto, sobre dicha planificación inicial no hizo falta realizar ningún cambio, pero no ha sucedido lo mismo con algunas herramientas utilizadas. Por ejemplo, tenía pensado hacer uso de *IDLE* de Phyton como IDE de desarrollo, sin embargo, posteriormente cambié de idea y utilicé *PyCharm,* ya que, bajo mi punto de vista, ofrece un entorno más intuitivo y mejores opciones para el desarrollo que se buscaba. Lo mismo sucedió con las librerías Phyton que se han utilizado, que no corresponden a las que me había planteado inicialmente.

Una última conclusión, esta a nivel personal, es me ha parecido una experiencia totalmente enriquecedora, gracias a la cual he adquirido una gran cantidad de conocimientos y habilidades de las que no disponía y que de ahora en adelante me serán de mucha utilidad, no solo a nivel técnico, sino también en cuanto a desarrollo de proyectos y planificación se refiere.

Cambiando de orden, con el proyecto ya finalizado se podría afirmar que el impacto que este genera en cuanto a los ODS de sostenibilidad, ético-social y de diversidad no han variado con respecto a los previstos en el apartado 1.3.

Por último, los trabajos futuros se centrarán en lograr una mayor seguridad y robustez del sistema. Para ello, se proponen las siguientes actuaciones:

- Aplicar EAP-TLS cómo método de autenticación, es decir, mediante certificados digitales tanto en el cliente como en el servidor. Para lograrlo, será necesario montar un servidor que actúe como entidad certificadora.
- Integrar *packetfence* con *Nessus* u *Openvass*, con el objetivo de poder analizar vulnerabilidades en los equipos antes de permitirles el acceso a la red. Con esta actuación y la anterior, estaremos logrando lo que se conoce como una red ZTNA (*Zero Trust Network Access*).
- Mejorar la interfaz gráfica de la aplicación web.
- Basar la autenticación de la aplicación en directorio activo y cifrar la comunicación mediante HTTPS.
- Incluir nuevas funcionalidades y mejoras en las actuales, como podría ser permitir definir filtros en la captura de paquetes o detectar automáticamente tráfico de red anómalo en el servidor NAC.
- Simplificar y unificar el código de diferentes scripts, ficheros *html* y funciones.

7. Glosario

802.1x – *Protocolo de control de acceso a la red basada en puertos.*
AD – *Active Directory*
BYOD – *Bring Your Own Device*
BD – *Base de Datos*
DHCP – *Dynamic Host Configuration Protocol*
ENISA - *European Union Agency for Cybersecurity*
EDR – *Endpoint Detection Response*
EAP - *Extensible Authentication Protocol*
EAPoL – *EAP over Lan*
HTTP – *HyperText Transfer Protocol*
HTML - *HyperText Markup Language*
IoT – *Internet of Things*
IPS – *Intrusion Prevention System*
IP – *Internet Protocol*
IDE - *Integrated Development Environment*
LAN – *Local Area Network*
LDAP – *Lightweight Directory Access Protocol*
MAC – *Media Access Control*
MAB – *MAC Authentication Bypass*
MITM – *Men In The Middle*
NAC – *Network Access Control* (Control de acceso a la red)
OSI – *Open System Interconnection*
OOWS – *Object Oriented Web Solution*
RGPD – *Reglamento General de Protección de datos*
SSID – *Service Set Identifier*
SSH – *Secure Shell*
SFTP - *Secure File Transfer Protocol*
UDP – *User Datagram Protocol*
VLAN – *Virtual LAN*
WPA - *Wi-Fi Protected Access*
WAF – *Web Application Firewall*
Wifi – *Wireless Fidelity*
WAN – *Wide Area Network*

8. Bibliografía

1. **Joseph Matthews (2021). Challenges to Implement Network Access Control.** Disponible en: **https://sansorg.egnyte.com/dl/V1XLQeMYgp** (Consultado: 05/10/2023)

2. **Trevor J. Dildy (2016). Network Access Control: Has It Evolved Enough for Enterprises?**. Disponible en: **https://www.isaca.org/en/resources/isaca-journal/issues/2016/volume-4/network-access-control-has-it-evolved-enough-for-enterprises** (Consultado: 05/10/2023)

3. **Sarah Laoyan (2022). Qué es la metodología waterfall y cuándo utilizarla.** Disponible en: **https://asana.com/es/resources/waterfall-project-management-methodology** (Consultado: 07/10/2023)

4. **Govloop (2018). Network Access Control: Your First Line of Cyber Defense.** Disponible en: **https://go.govloop.com/rs/231-DWB-776/images/Network%20Access%20Control.pdf?** (Consultado: 17/10/2023)

5. **Robert Grimmick (2022). What is Network Access Control? Explaining NAC Solutions.** Disponible en: **https://www.varonis.com/blog/network-access-control-nac** (Consultado: 17/10/2023)

6. **ENISA (2023). Enisa Threat Landscape 2023**. Disponible en: **https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023/@@download/fullReport** (Consultado: 16/10/2023)

7. **Elisa Vivancos (INCIBE – 2022). Los 10 vectores de ataque más utilizados por los ciberdelincuentes.** Disponible en: **https://www.incibe.es/empresas/blog/los-10-vectores-ataque-mas-utilizados-los-ciberdelincuentes** (Consultado: 16/10/2023)

8. **Cisco (2023). ¿Qué es la seguridad de red?** Disponible en: **https://www.cisco.com/c/es_mx/products/security/what-is-network-security.html** (Consultado: 16/10/2023)

9. **Caballero Tobajas, José María (2019). Control de acceso sobre red cableada. Trabajo de fin de grado**. Disponible en: **https://addi.ehu.es/bitstream/handle/10810/36034/TFG_Final.pdf?sequence=2&isAllowed=y** (Consultado: 19/10/2023)

10. **Adanéz Arroyo, Darío (2020). Sistema de control de acceso a redes (NAC) basado en SNMP y VLAN.** Disponible en: **https://uvadoc.uva.es/bitstream/handle/10324/42477/TFG-G4145.pdf?sequence=1&isAllowed=y** (Consultado: 20/10/2023)

11. **Jenifa, Ashlin (2023). Por qué es importante el control de acceso a la red y cómo implementarlo**. Disponible en: **https://geekflare.com/es/network-access-control/** (Consultado: 21/10/2023)

12. **Fortinet (2023). What is Network Access Control (NAC)?** Disponible en: **https://www.fortinet.com/resources/cyberglossary/what-is-network-access-control** (Consultado: 20/10/2023)

13. **Network Interview (2023). Common Challenges in Implement NAC Solutions.** Disponible en: **https://networkinterview.com/challenges-in-implementing-nac-solutions/** (Consultado: 20/10/2023)

14. **Check Point (2023). Why is Network Access Control Important (NAC)?** Disponible en: **https://www.checkpoint.com/cyber-hub/network-security/what-is-network-access-control-nac/** (Consultado: 21/10/2023)

15. **Raj, Vivec (SecureW2 - 2023). Network Acces Control: Explained.** Disponible en: **https://www.securew2.com/blog/network-access-control** (Consultado: 23/10/2023)

16. **Vina, Suresh (2022). Everything you need to know about NAC, 802.1x and MAB.** Disponible en: **https://www.packetswitch.co.uk/everything-you-need-to-know-about-nac-802-1x-and-mab/** (Consultado: 23/10/2023)

17. **Intel (2021). Descripción general de 802.1x y tipos de EAP.** Disponible en: **https://www.intel.la/content/www/xl/es/support/articles/000006999/wireless/legacy-intel-wireless-products.html** (Consultado: 24/10/2023)

18. **Huawei (2023). Understanding 802.1x Authentication.** Disponible en: **https://support.huawei.com/enterprise/en/doc/EDOC1100086527** (Consultado: 24/10/2023)

19. **Study-ccnp.com (2023). MAC Authentication Bypass (MAB) Authentication Explained.** Disponible en: **https://study-ccnp.com/mac-authentication-bypass-mab-authentication-explained/** (Consultado: 25/10/2023)

20. **SecureW2 (2023). Simplifying WPA2-Enterprise and 802.1x.** Disponible en: **https://www.securew2.com/solutions/wpa2-enterprise-and-802-1x-simplified** (Consultado: 26/10/2023)

21. **Microsoft (2023). Network Policy Server (NPS).** Disponible en: **https://learn.microsoft.com/en-us/windows-server/networking/technologies/nps/nps-top** (Consultado: 26/10/2023)

22. **Huawei (2023). What is Network Admission Control (NAC)?** Disponible en: **https://info.support.huawei.com/info-finder/encyclopedia/en/NAC.html** (Consultado: 27/10/2023)

23. **Cisco (2023). What is Network Access Control?** Disponible en: **https://www.cisco.com/c/en/us/products/security/what-is-network-access-control-nac.html** (Consultado: 27/10/2023)

24. **Raphaely, Eytan (SecureW2 2023). 802.1x Network Attack Vectors.** Disponible en: **https://www.securew2.com/blog/802-1x-network-attack-vectors** (Consultado: 01/11/2023)

25. **Secure2W (2023). What is 802.1x? How does it Work?** Disponible en: **https://www.securew2.com/solutions/802-1x#documentation-link-5-6** (Consultado: 01/11/2023)

26. **FuProject (2019). Network Access Control Bypass.** Disponible en: **https://www.flu-project.com/2019/09/network-access-control-bypass.html** (Consultado: 02/11/2023)

27. **Bourbonnais, Roch (Oracle - 2005). ZFS to UFS Performance Comparision on Day 1.** Disponible en: **https://blogs.oracle.com/solaris/post/zfs-to-ufs-performance-comparison-on-day-1** (Consultado: 20/11/2023)

28. **Ionos (2022). Flask vs Django: una comparativa de los frameworks de Phyon**. Disponible en**: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/flask-vs-django/** (Consultado: 23/11/2023).

29. **TecnoINTELECTO (2018). Modelado conceptual de una aplicación web usando la metodología OOWS: Un caso práctico.** Disponible en: **https://www.itvictoria.edu.mx/oferta//maestria_sistemas/Archivos/2.%20Modelado%20conceptual%20de%20una%20aplicaci%c3%b3n%20Web%20usando%20la%20metodolg%c3%ada%20OOWS.pdf** (Consultado: 24/11/2023)

30. **Monografías.com (2023). Modelando aplicaciones web con UML**. Disponible en: **https://www.monografias.com/trabajos107/modelando-aplicaciones-web-uml/modelando-aplicaciones-web-uml** (Consultado: 24/11/2023)

31. **NetgateDocs (2023). pfSense Documentation**. Disponible en: **https://docs.netgate.com/pfsense/en/latest/** (Consultado: 21/11/2023)

32. **PacketFence.org (2023). Documentation.** Disponible en: **https://www.packetfence.org/support.html#/documentation** (Consultado: 23/11/2023)

33. **Manualslib (2023). Enterasys B5 CLI Reference.** Disponible en: **https://www.manualslib.com/manual/1201974/Enterasys-B5.html** (Consultado:23/11/2023)

34. **Flask (2023). Flask User Guide**. Disponible en: **https://flask.palletsprojects.com/en/3.0.x/** Consultado: 10/10/2023)

35. **Paramiko (2023). Paramiko documentation.** Disponible en: **https://www.paramiko.org/** (Consultado: 15/12/2023)

36. **Ldap3 (2023). Ldap3 documentation.** Disponible en: **https://ldap3.readthedocs.io/** (Consultado: 17/12/2023)

37. **W3Schools.com (2023). Phyton MySQL.** Disponible en: **https://www.w3schools.com/python/python_mysql_getstarted.asp** (Consultado: 21/12/2023)

38. **Phyton.org (2023). Tkinder.** Disponible en: **https://docs.python.org/es/3/library/tkinter.html** (Consultado: 21/12/2023)

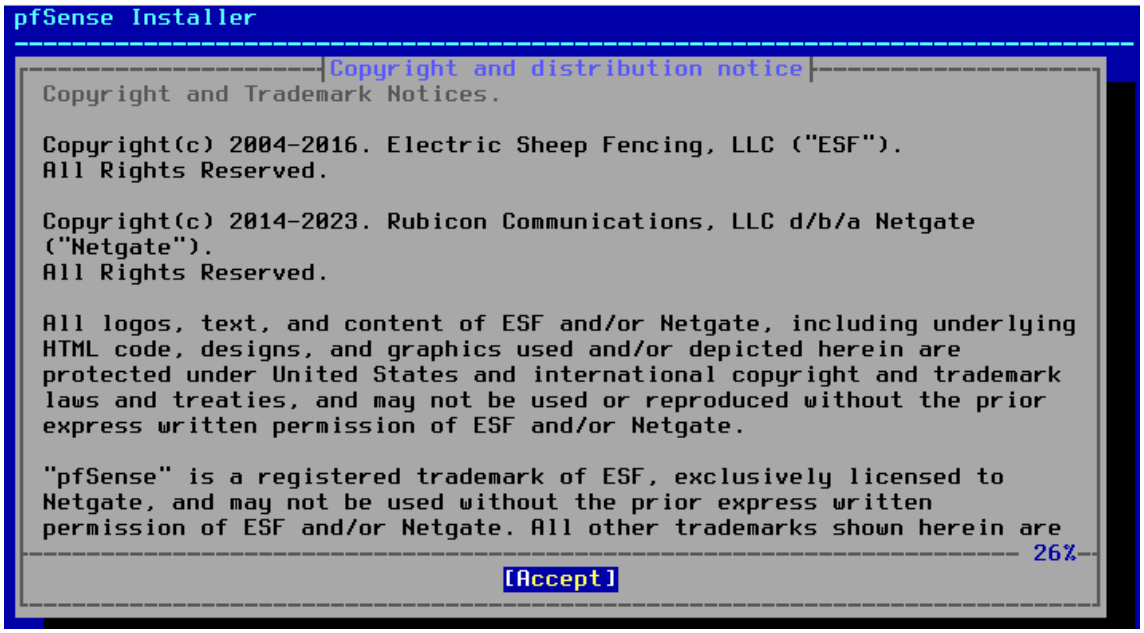
9. Anexos

Anexo 1: Instalación y configuración de pfSense

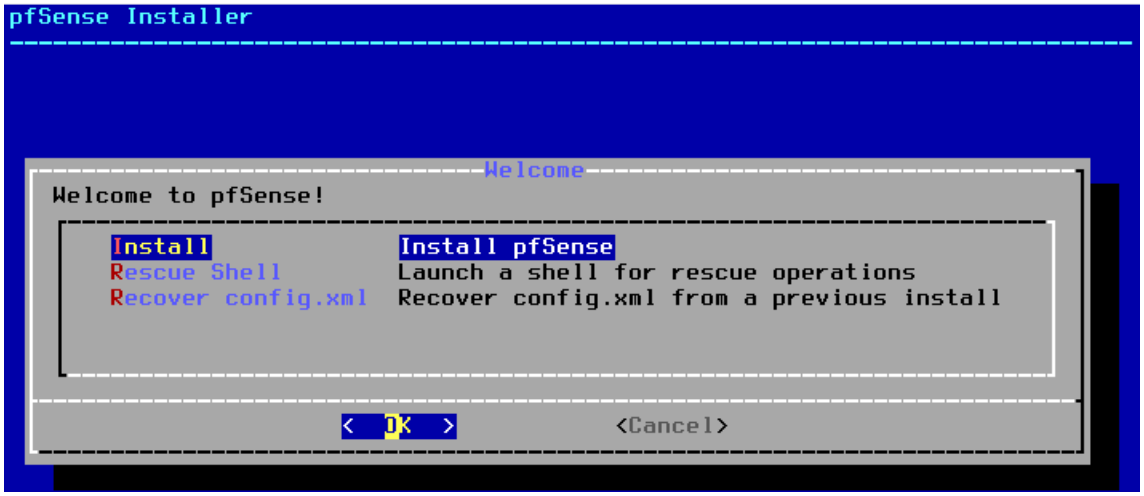
1.1. Instalación

Una vez descargado el software desde la página web oficial, bastará con montarlo en una memoria USB e iniciar el equipo desde la misma. Con ello, comienza el proceso de instalación y posterior configuración, que se basa en los siguientes pasos:

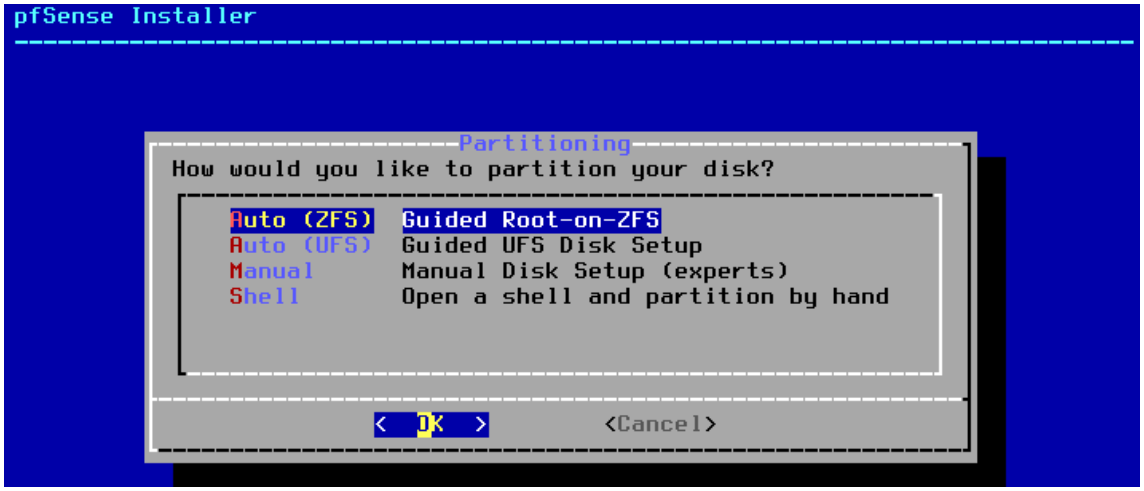
- *Paso 1*: Aceptar los términos y condiciones…



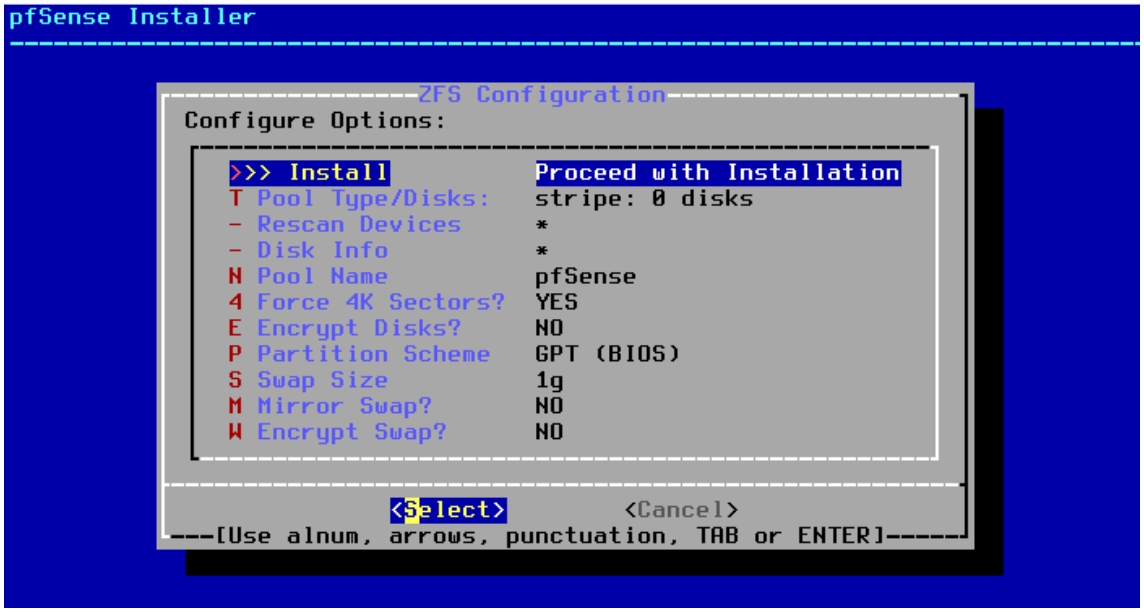
- *Paso 2*: Seleccionar la opción "*Install*"



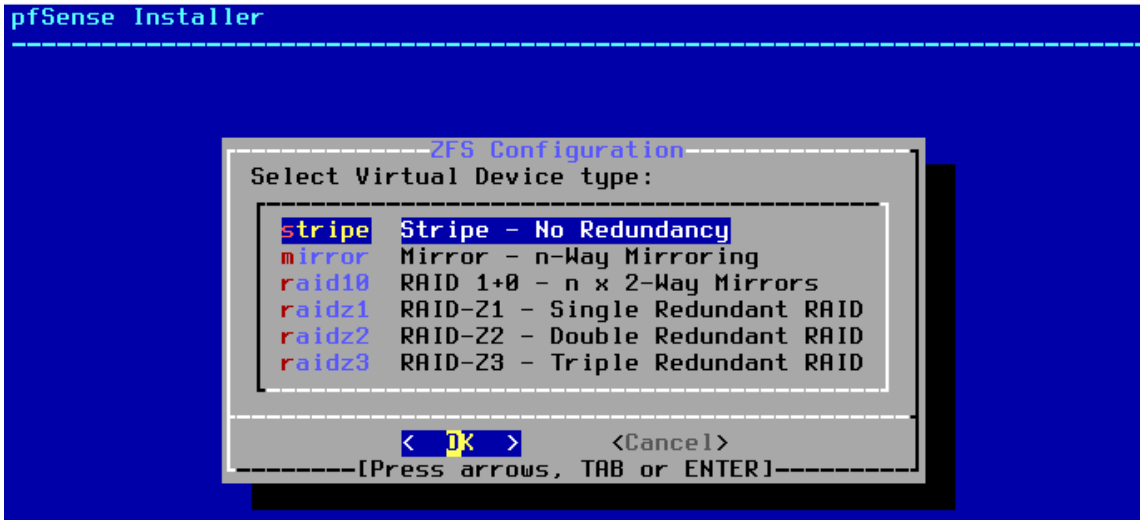
- *Paso 3*: Indicar el método que se utilizará para crear las particiones necesarias en el disco. Para este caso utilizaremos la opción "*Auto (ZFS)*", de tal manera que se crearán automáticamente tras definir una serie de datos de manera guiada. Se ha seleccionado el sistema de archivos ZFS ya que en comparación con UFS presenta un mejor rendimiento y eficiencia [27].



- *Paso 4*: Tras aceptar, el sistema nos presentará una serie de datos para definir las particiones, los cuales podremos modificar. Sin embargo, para esta instalación mantendremos los valores por defecto, por tanto, tan solo seleccionamos la opción "*Install*" y continuamos…



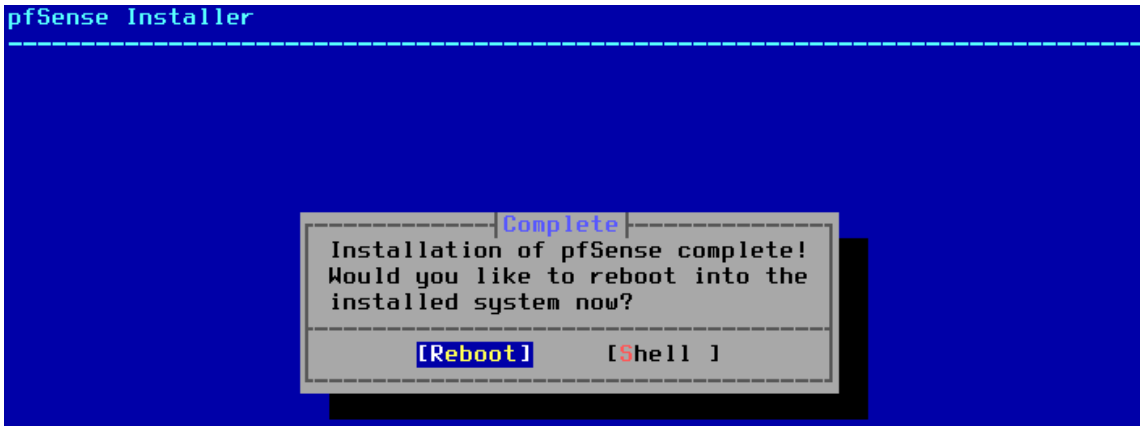
- *Paso 5*: Seleccionamos el tipo de redundancia de disco que deseemos. Para este caso no hará falta, por lo que bastará con marcar la opción "*stripe*".



- *Paso 6*: Seleccionamos el disco físico en el cual se crearán las particiones ZFS…

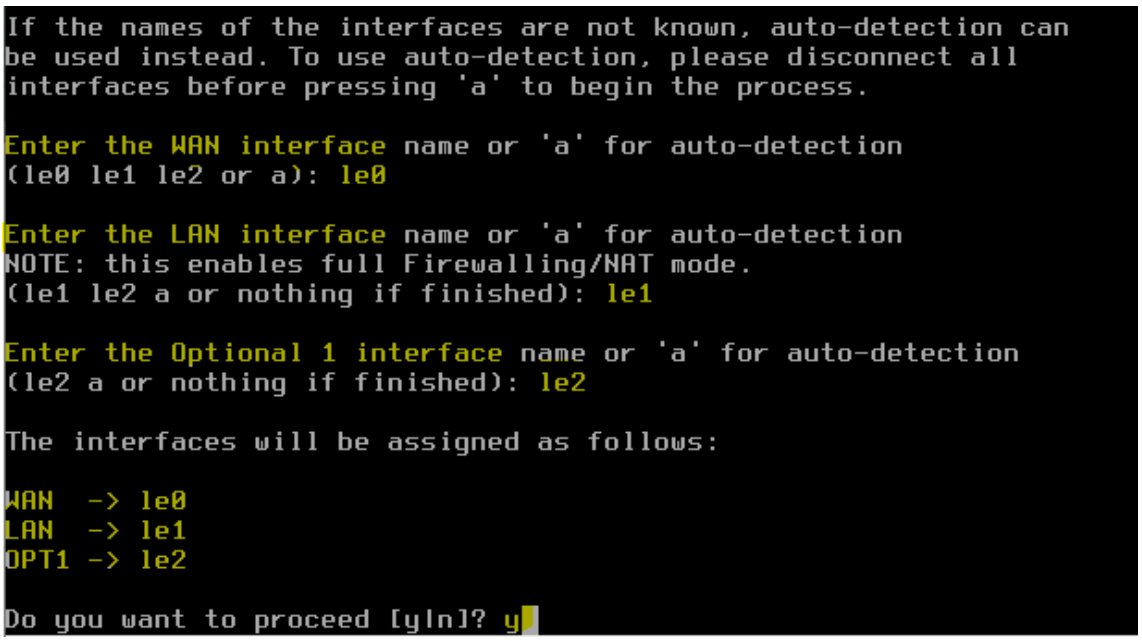
- *Paso 7*: Por último, aceptamos el aviso de que los datos del disco se borrarán...

- *Paso 8*: Tras ello, comienza la instalación de pfSense, la cual tardará unos minutos en finalizar. Una vez concluida, aparecerá el siguiente mensaje, donde se requiere el reinicio del sistema para comenzar a utilizar el software...

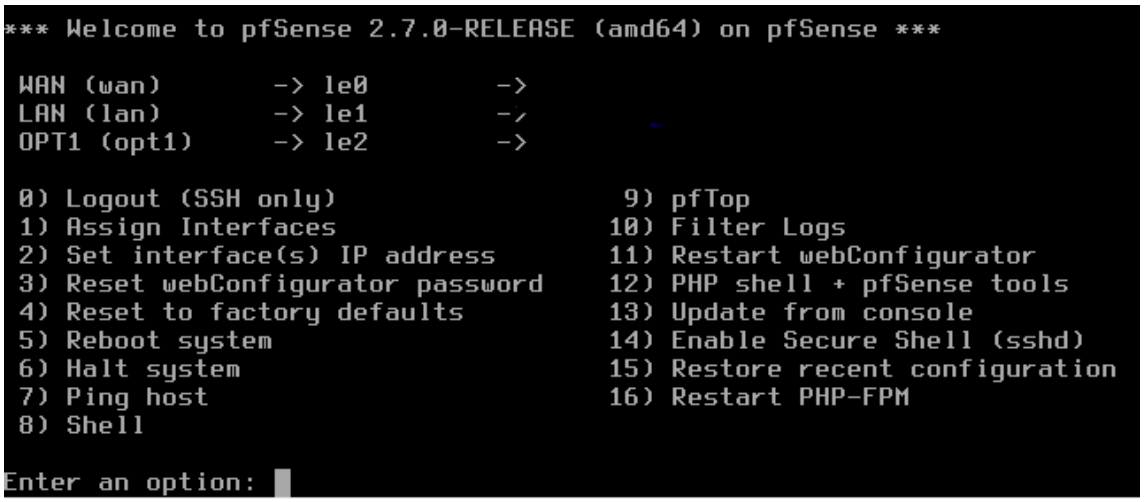


1.2. Configuración

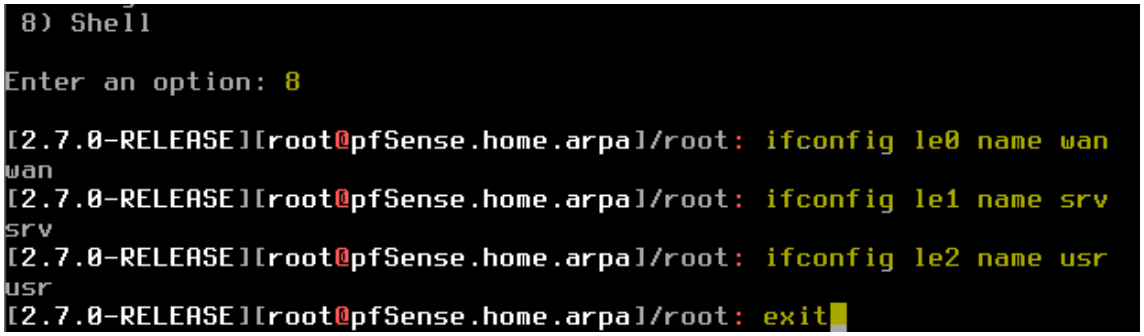
- *Paso 9*: Una vez reiniciado el sistema se mostrará una consola que, en primer lugar, solicitará identificar la conexión física de cada una de las interfaces. Para el caso de la red WAN, se utilizará la interfaz "*le0*", para la LAN de servidores, la "*le1*", y para la LAN de usuarios, la "*le2*". Cabe mencionar que esta tercera interfaz en un principio es denominada como *Optional* por *pfSense*, pero luego le modificaremos el nombre para no generar confusión.



- *Paso 10*: Tras asignar las interfaces, el sistema arrancará numerosos servicios, y, al finalizar, mostrará el siguiente menú…



- *Paso 11* (opcional): Antes de proceder a la configuración, y con el fin de evitar posibles confusiones, renombraré las interfaces de la siguiente manera; la interfaz "*le0*" se llamará "*wan*", la "*le1*" se denominará "*srv*", mientras que "*le2*" pasará a identificarse como "*usr*". Para ello, accedemos a la opción 8 (*Shell*) e introducimos los siguientes comandos:

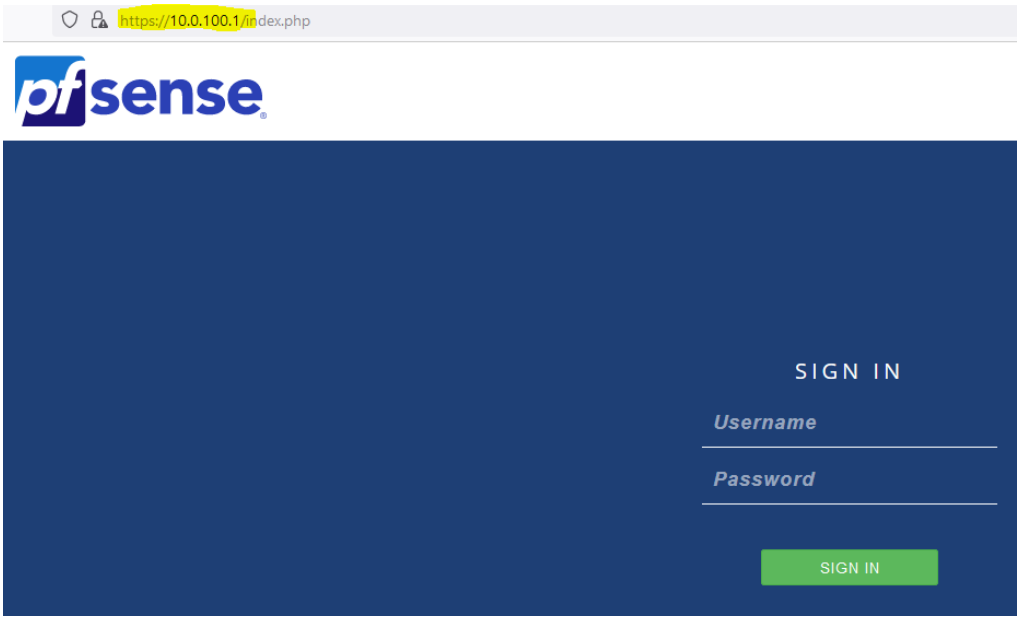


- *Paso 12*: Configuramos una dirección IP en alguna de las interfaces LAN, gracias a lo cual podremos acceder vía web al firewall y continuar con la configuración de manera más intuitiva. Se considera adecuado hacer uso de la interfaz que conecta con la red de servidores, es decir, la denominada como "*srv*", con IP 10.0.100.1/24. Para lograrlo, accedemos a la opción 2 del menú, seleccionamos la interfaz "*srv*" y respondemos a las preguntas (IP, máscara, etc.) que se nos irán presentando…

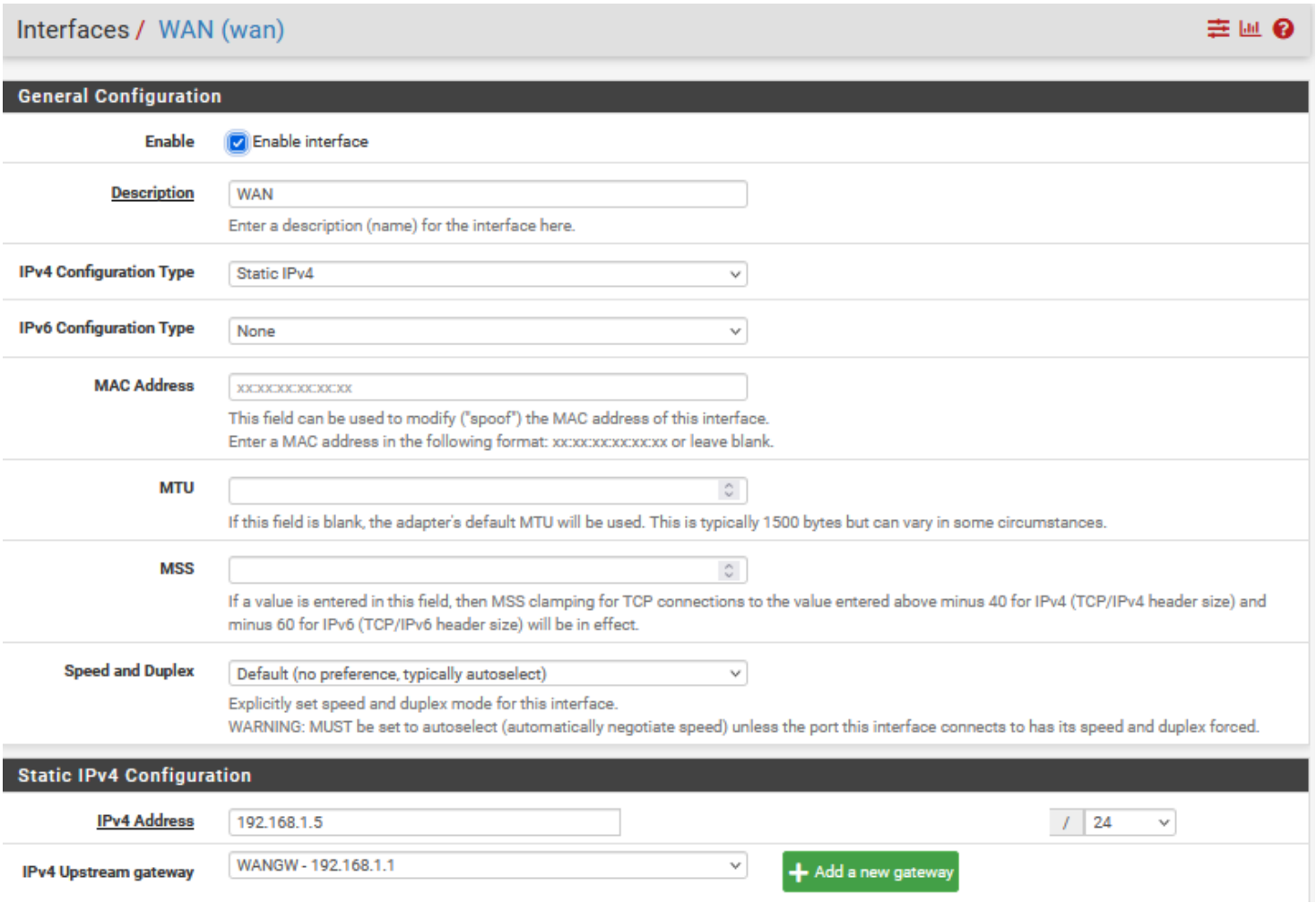
```
Enter an option: 2

Available interfaces:

1 - WAN (wan - dhcp, dhcp6)
2 - LAN (srv - static)
3 - OPT1 (usr)

Enter the number of the interface you wish to configure: 2

Configure IPv4 address LAN interface via DHCP? (y/n) n

Enter the new LAN IPv4 address.  Press <ENTER> for none:
> 10.0.100.1

Subnet masks are entered as bit counts (as in CIDR notation) in pfSense.
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new LAN IPv4 subnet bit count (1 to 32):
> 24
```

```
The IPv4 LAN address has been set to 10.0.100.1/24
You can now access the webConfigurator by opening the following URL in your web
browser:
                https://10.0.100.1/

Press <ENTER> to continue.
VirtualBox Virtual Machine - Netgate Device ID: 5cce1f59c2842ce7463a

*** Welcome to pfSense 2.7.0-RELEASE (amd64) on pfSense ***

 WAN (wan)       -> wan          ->
 LAN (lan)       -> srv          -> v4: 10.0.100.1/24
 OPT1 (opt1)     -> usr          ->
```

- *Paso 13*: Con los cambios realizados, el servidor ya es accesible vía web, por lo que continuaremos la configuración desde dicha interfaz, accediendo a la URL https://10.0.100.1 con las credenciales "*admin*" (usuario) y "*pfsense*" (contraseña)…

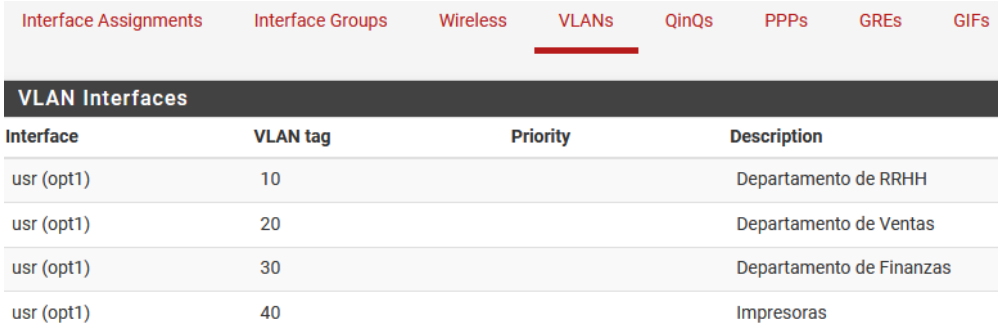


- *Paso 14*: Una vez dentro, lo primero que haremos será terminar de configurar las interfaces, comenzando por la denominada "*wan*", que es aquella que conecta con Internet…

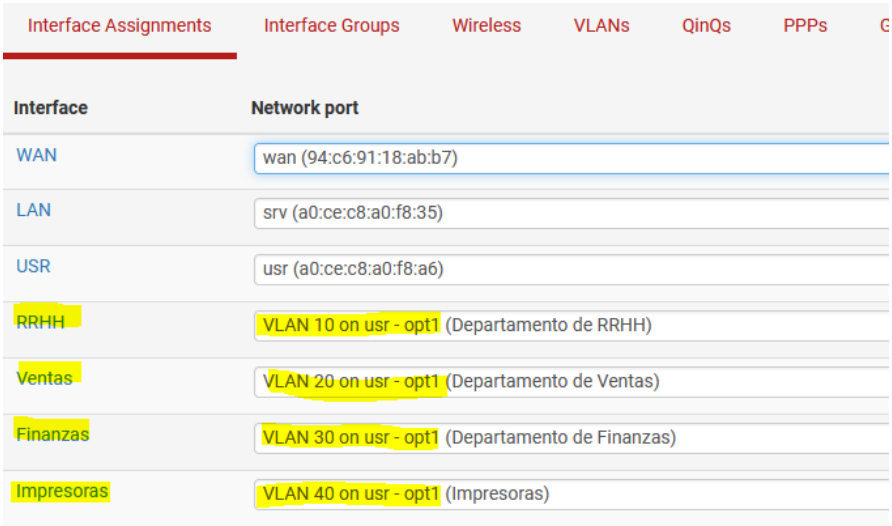


- *Paso 15*: Continuamos con la interfaz "*usr*", que es aquella que conecta con el Switch de usuarios. Para este caso será necesario crear y asociar las VLANs de los diferentes perfiles a la interfaz, ya que será la puerta de enlace de todas ellas. Además, también actuará como servidor DHCP de dichas subredes (a excepción de la VLAN de impresoras) …

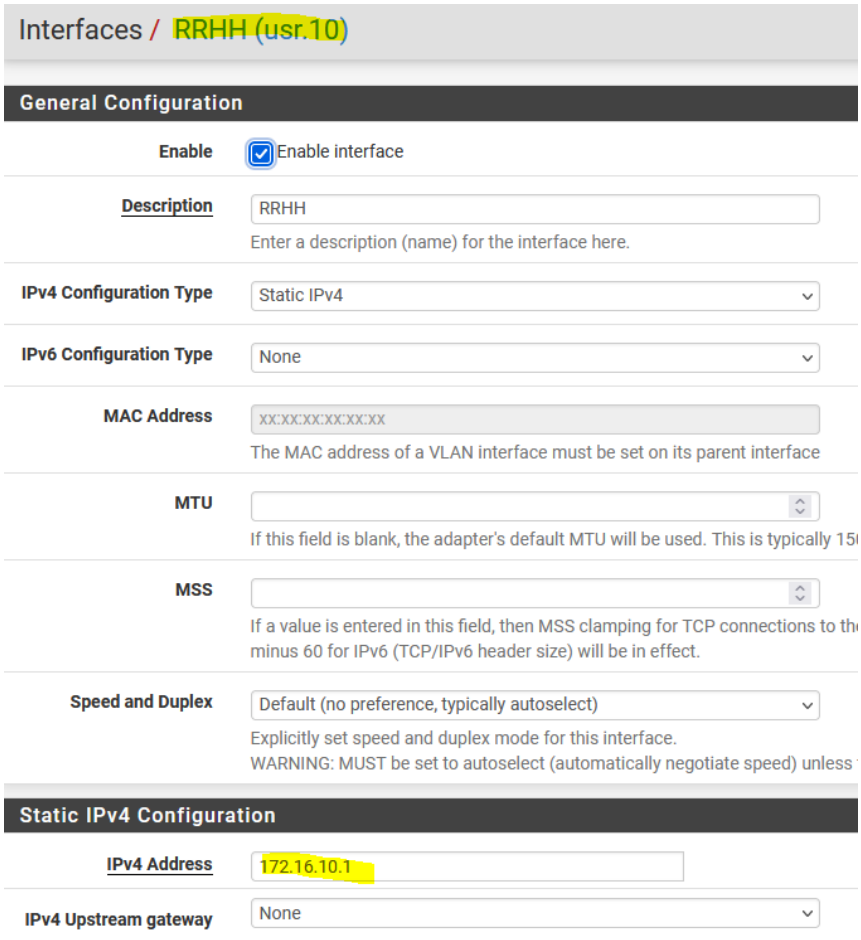
  Primero, creamos todas las VLAN…



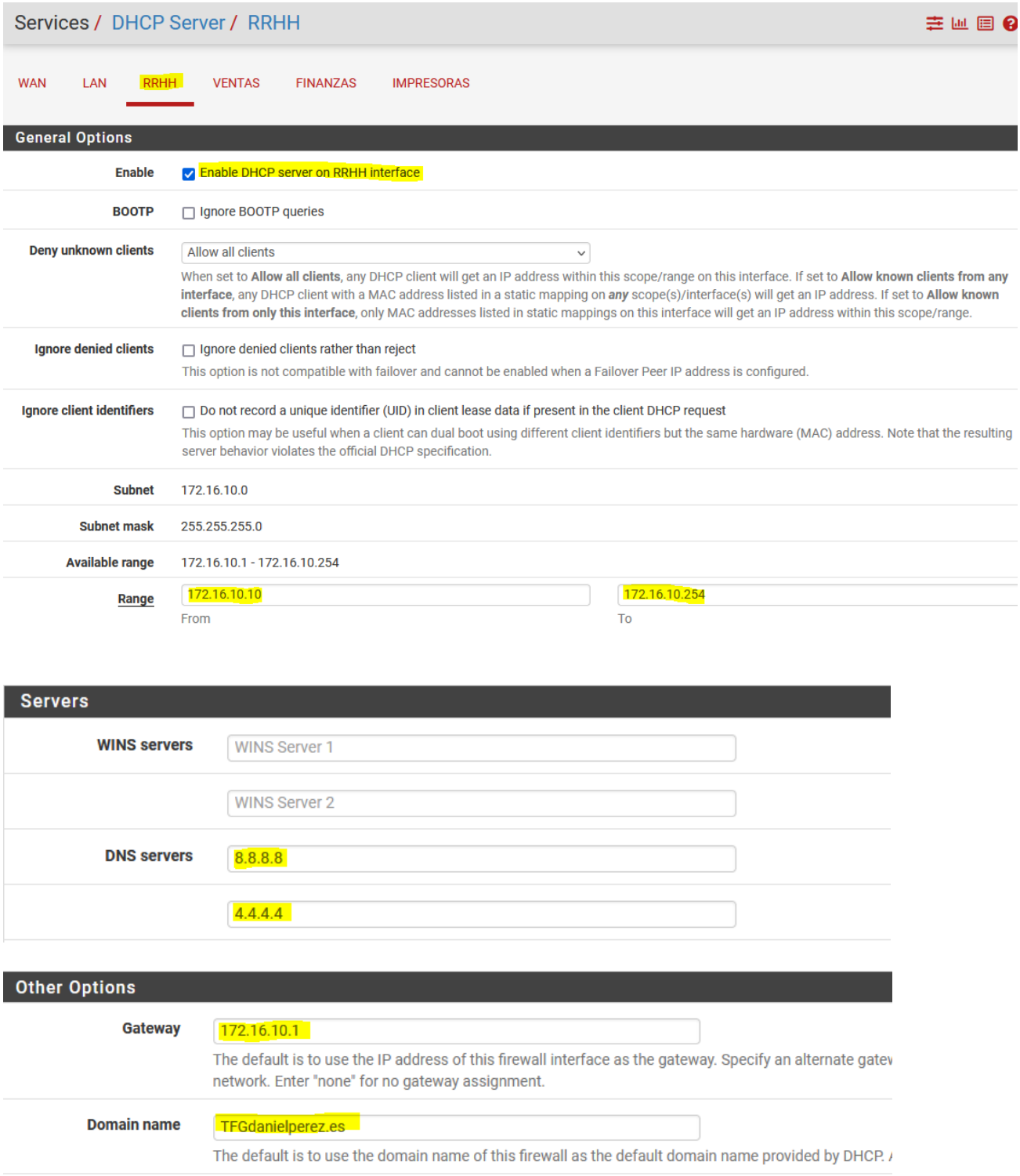
Segundo, creamos nuevas interfaces de VLAN, asociadas a la interfaz física "*usr*" …



Continuamos configurando cada una de ellas, asignando su correspondiente IP. Por ejemplo, la VLAN 10… (no se muestran todas por brevedad).

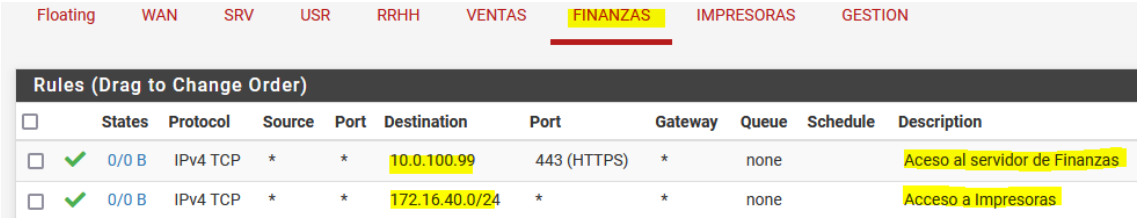


Y finalizamos habilitando el servidor DHCP para cada una de las subredes de usuarios, exceptuando la de impresoras… (por brevedad, solo se mostrará la configuración para RRHH).



- *Paso 16*: Por último, se configuran los accesos permitidos en capa 3 para cada uno de los perfiles de acceso, los cuales fueron definidos en el diseño del producto (apartado 3.1.4).

  Para ello, debemos acceder al apartado "*Firewall → Rules*", seleccionar la interfaz sobre la cual se aplicarán las reglas y crearlas. Por brevedad, solo se mostrarán aquellas aplicadas sobre el perfil de acceso de Finanzas… (La IP del servidor de finanzas es ficticia y solo se ha configurado a modo de ejemplo).

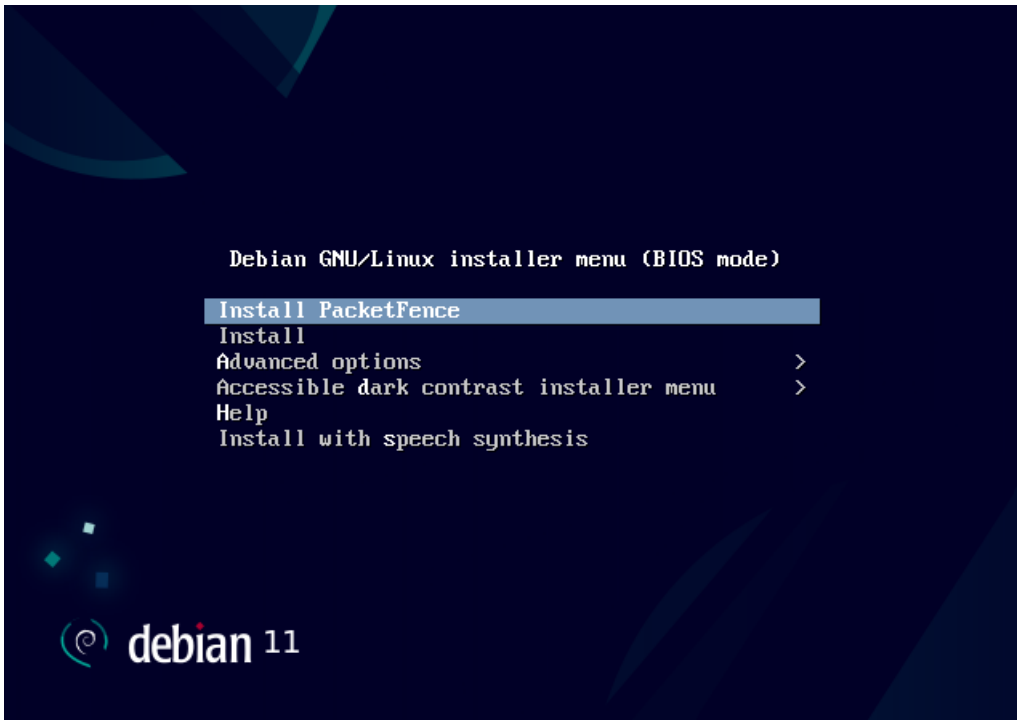


Anexo 2: Instalación y configuración de PacketFence

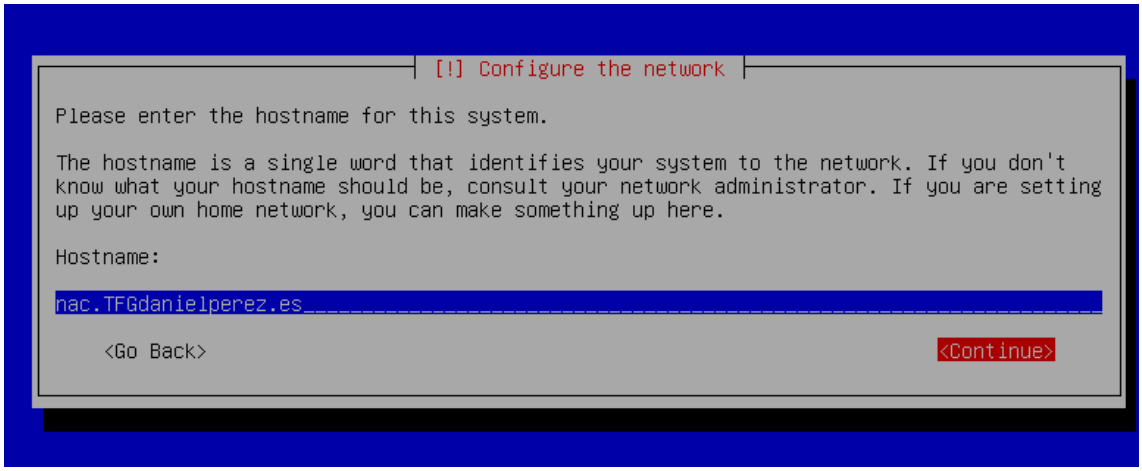
2.1. Instalación

- *Paso 1*: Descargar la imagen ISO desde la web oficial, montarla en una unidad USB e iniciar el equipo desde la misma.

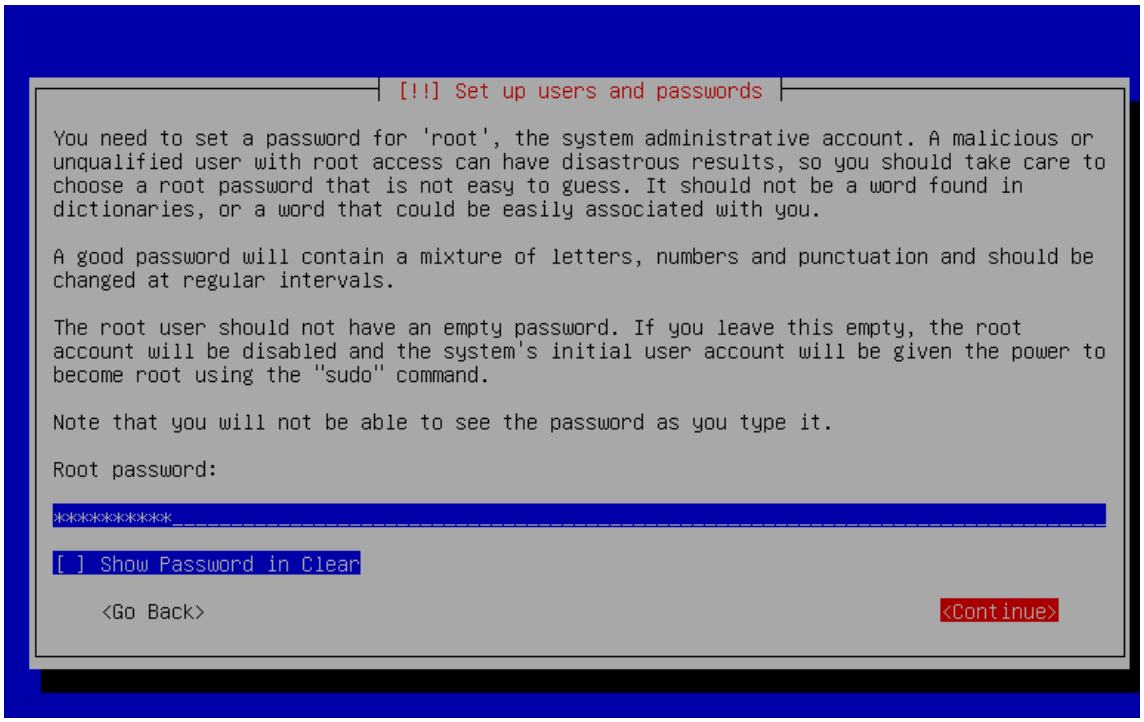
- *Paso 2*: En el menú inicial, seleccionar la opción "*Install PacketFence*".

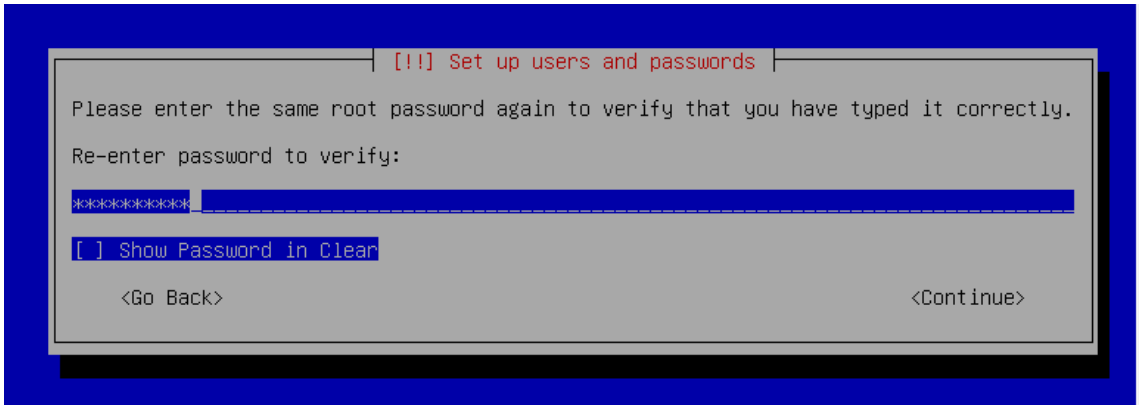


- *Paso 3*: El asistente de instalación comienza con la siguiente pantalla, donde debemos introducir el nombre que se le asignará al servidor. En este caso será "*nac.TFGdanielperez.es*".



- *Paso 4*: A continuación, se solicitará la contraseña para el usuario *root*, y posteriormente la confirmación de la misma…





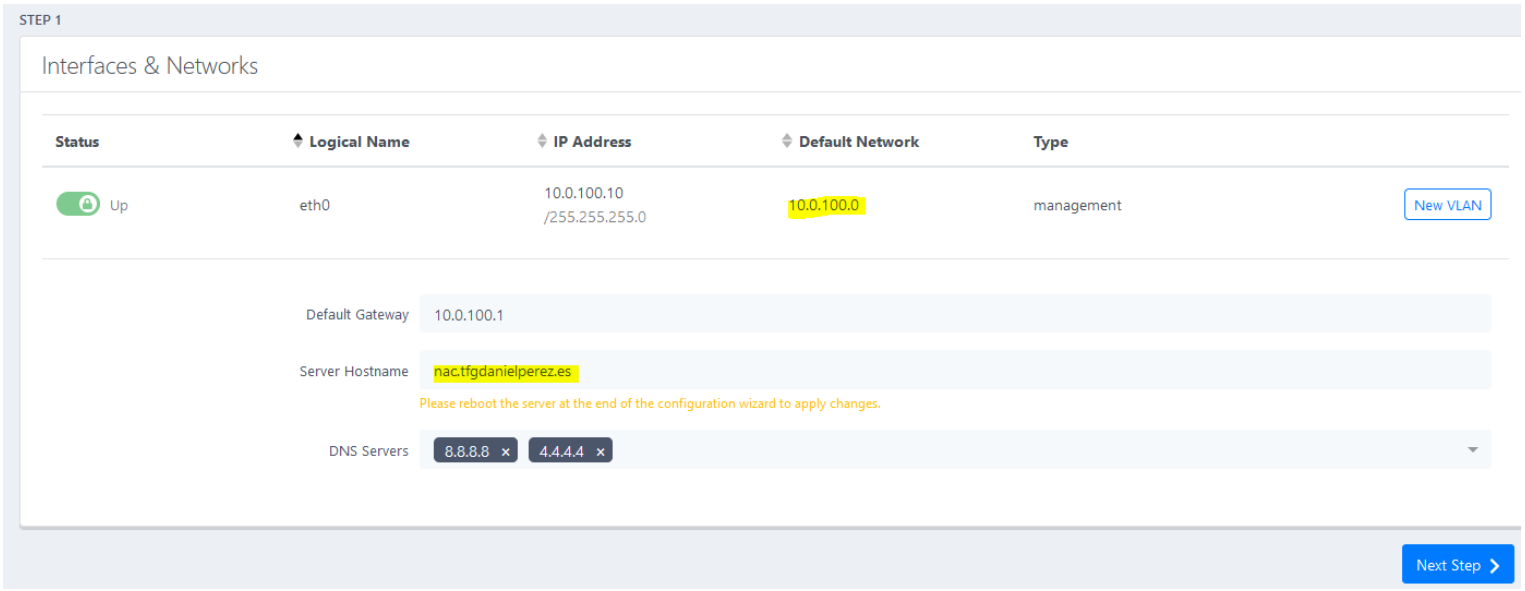
- *Paso 5*: En este momento comienza la instalación de los paquetes necesarios, lo cual tarda varios minutos en finalizar. Una vez concluya, el sistema se reinicia y carga en modo consola, solicitando iniciar sesión…

2.2. Configuración general

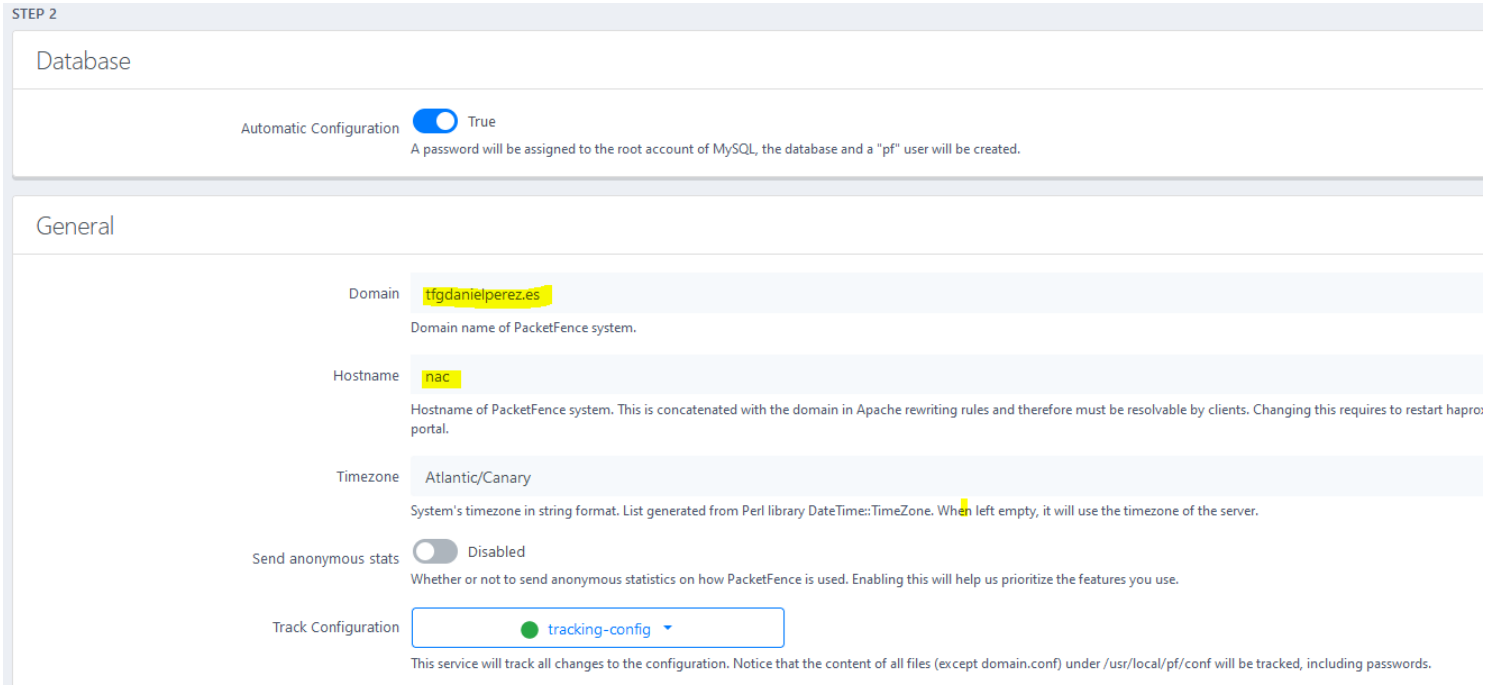
- *Paso 6*: Tras introducir las credenciales definidas en el paso 4 accedemos al sistema operativo, donde lo primero que haremos será configurar una IP estática al servidor, ya que por defecto tiene habilitada la opción por DHCP. Gracias a ello, posteriormente podremos acceder vía web y continuar con la configuración de manera más intuitiva. Por tanto, editamos el fichero "*/etc/network/interfaces*" y agregamos las siguientes líneas…

```
 GNU nano 5.4                                    interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 10.0.100.10/24
        gateway 10.0.100.1
```

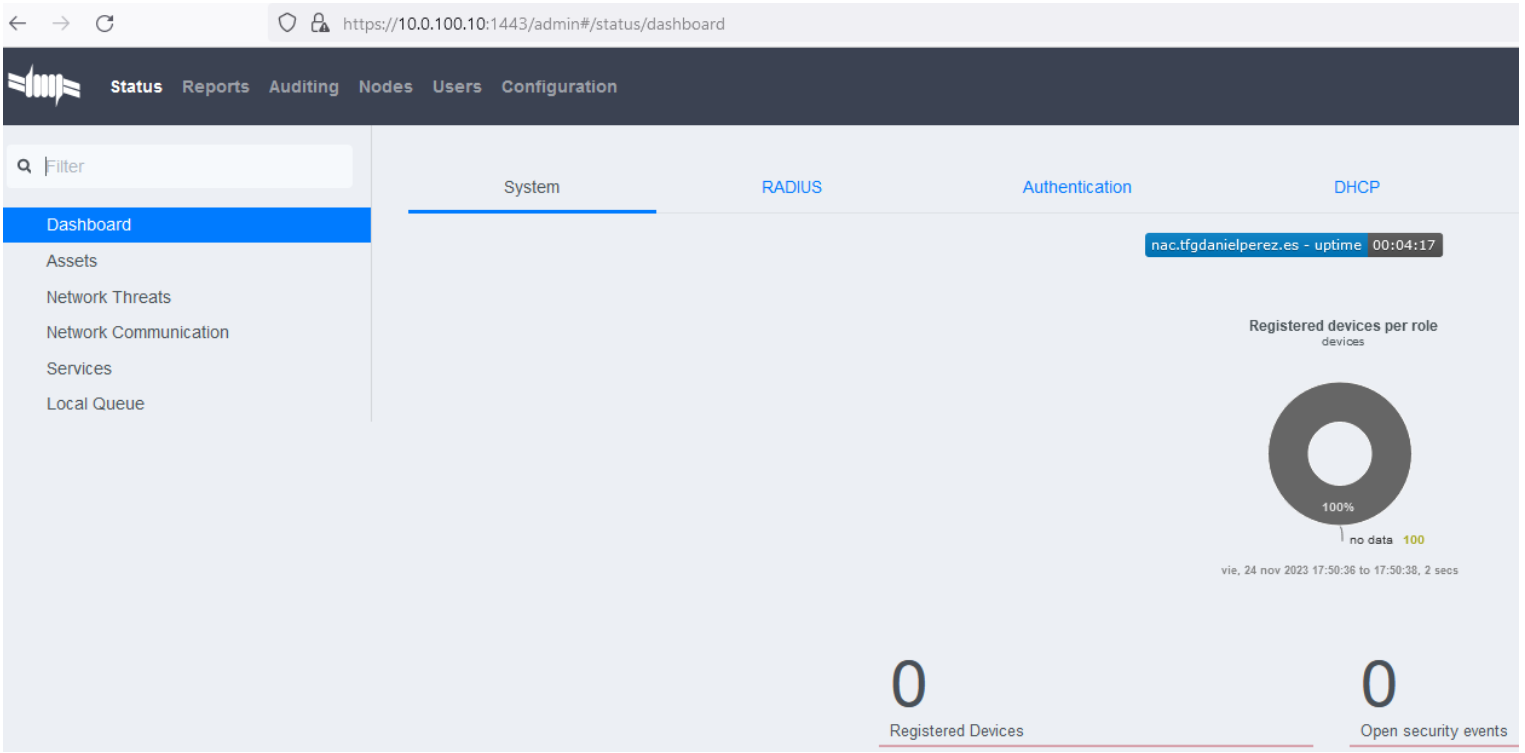
- *Paso 7*: Una vez hecho, guardamos el fichero y aplicamos los cambios, bien reiniciando el servidor, o bien reiniciando el servicio de red mediante el comando "*/etc/init.d/networking restart*". Tras ello, el servidor es accesible vía web a través de la URL *https://10.0.100.10:1443* , a la cual accedemos para continuar con la configuración. La primera pantalla que muestra es la siguiente, donde se permite modificar la configuración de red. En este caso es correcta, por lo que no hacemos ningún cambio.



- *Paso 8*: El siguiente paso permite configurar la base de datos y algunos otros aspectos generales del servidor, como el dominio, nombre, zona horaria y contraseña de administrador…

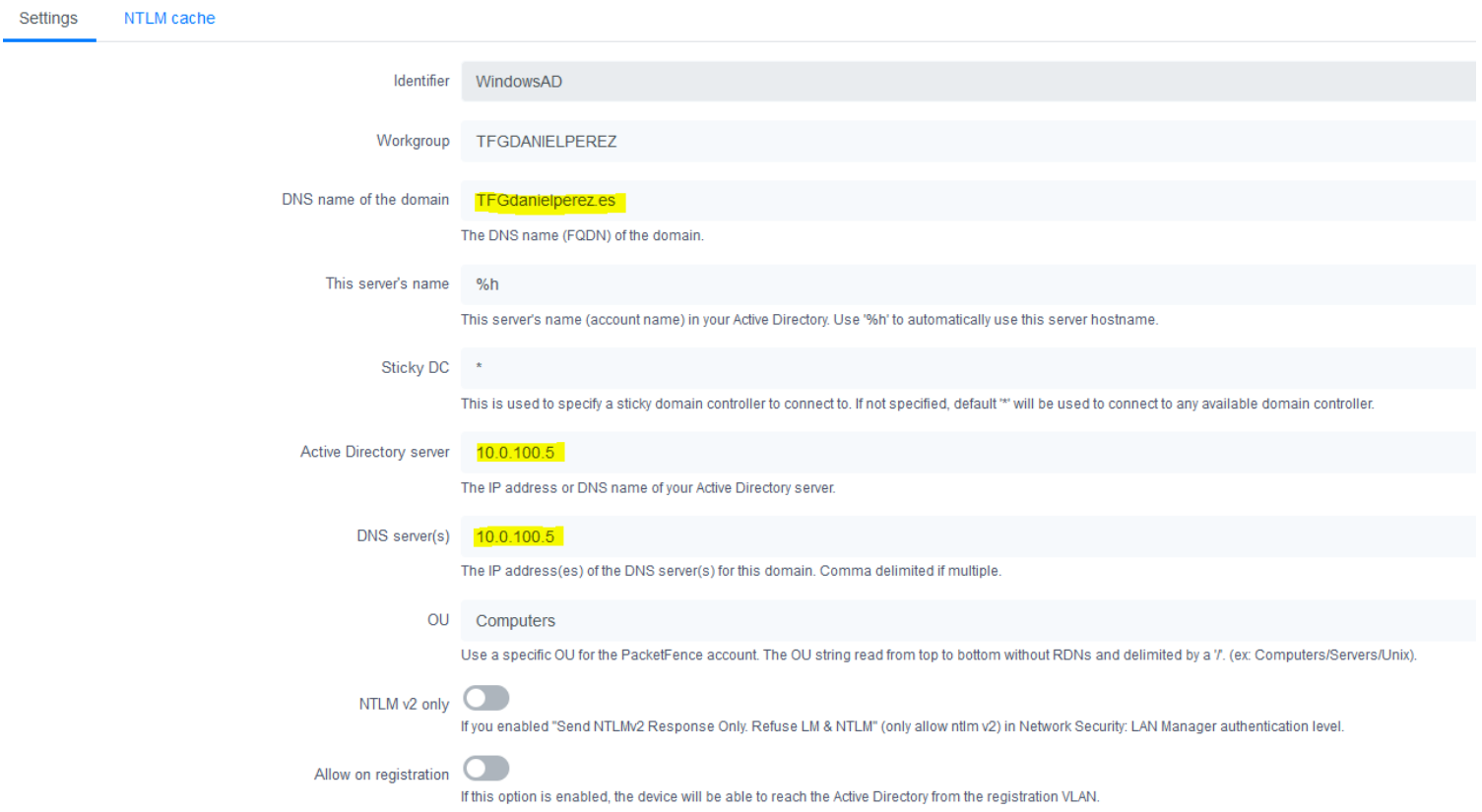


- *Paso 9*: Continuando con el asistente de configuración, finalmente mostrará las contraseñas de acceso a la base de datos, para, posteriormente, solicitar el usuario y *password* para acceder al sistema. Una vez introducido, se mostrará el menú principal del servidor NAC…

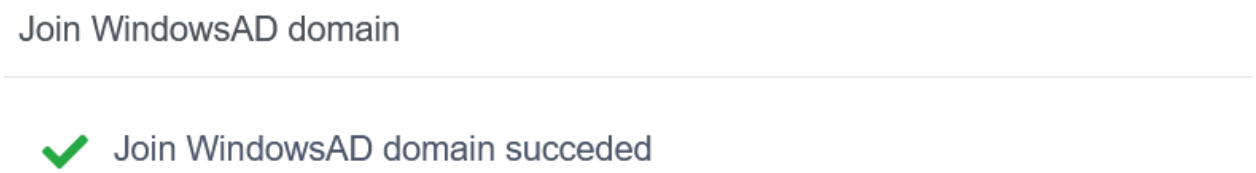


2.3. Configuración de autenticación basada en AD

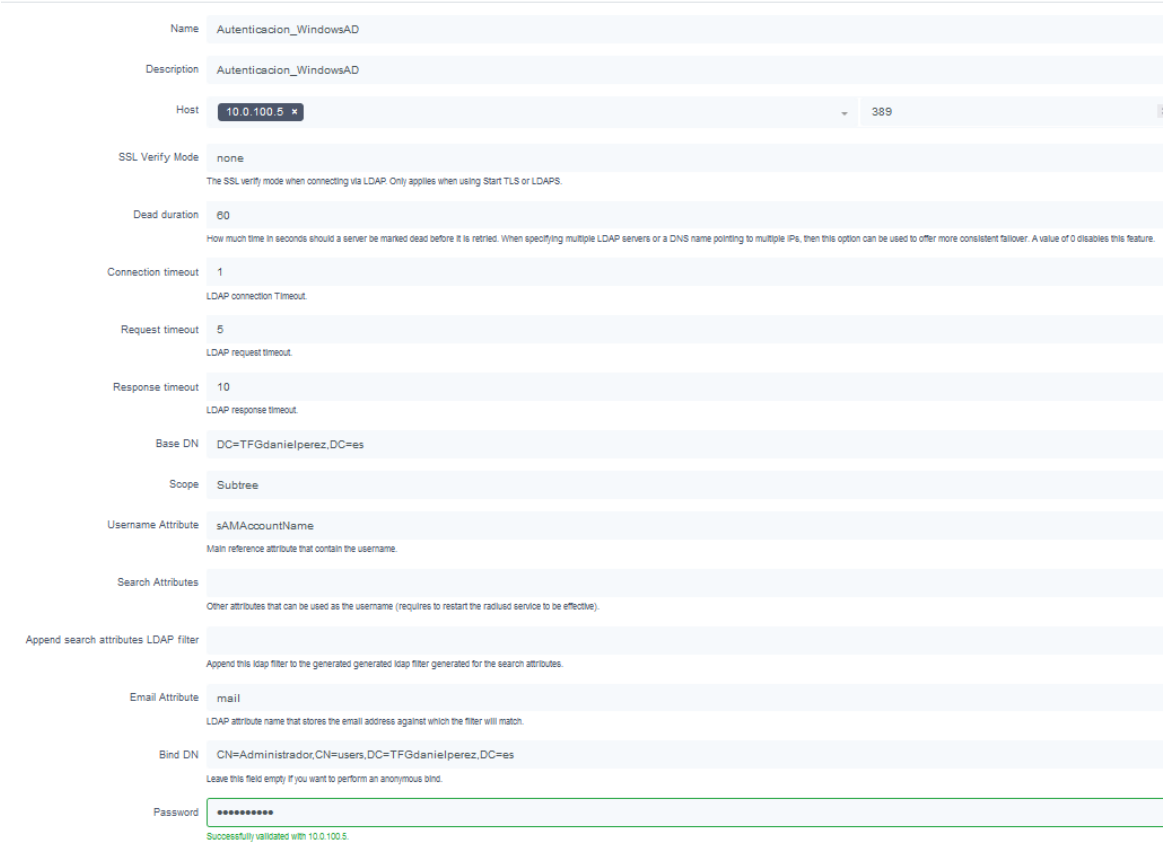
- *Paso 10:* Para poder definir la autenticación del servidor Radius de PacketFence por AD, primero debemos establecer la conexión de este con el controlador de dominio. Cabe recordar que la IP de dicho controlador es la 10.0.100.5, y el nombre de dominio, *TFGdanielperez.es*. Por tanto, el primer paso será acceder a la pantalla de configuración "*Configuration → Policies and Access control → Domains → Active directory domains"* y pulsar sobre el botón "*New domain".* La nueva pantalla que se mostrará solicitará los datos del servidor de dominio que deseamos agregar…



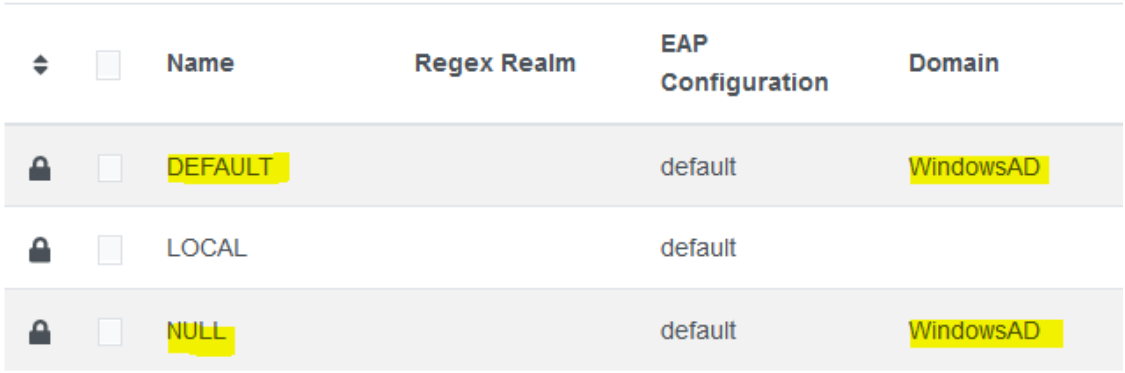
Tras guardar los cambios, deberemos introducir las credenciales de algún administrador de dominio para establecer la conexión. Si los datos son correctos, se mostrará la siguiente pantalla, que muestra el éxito de la conexión.



- *Paso 11*: Una vez agregado el AD, debemos definir este servidor como fuente de autenticación, para posteriormente utilizarlo en el Radius. Para ello, bastará con acceder a la pantalla de configuración "*Configuration → Policies and Access control → Authentication Sources",* y en el apartado *Internal Sources* agregar una nueva fuente basada en Active Directory. Se mostrará la siguiente pantalla, donde deberemos introducir los datos del directorio activo…

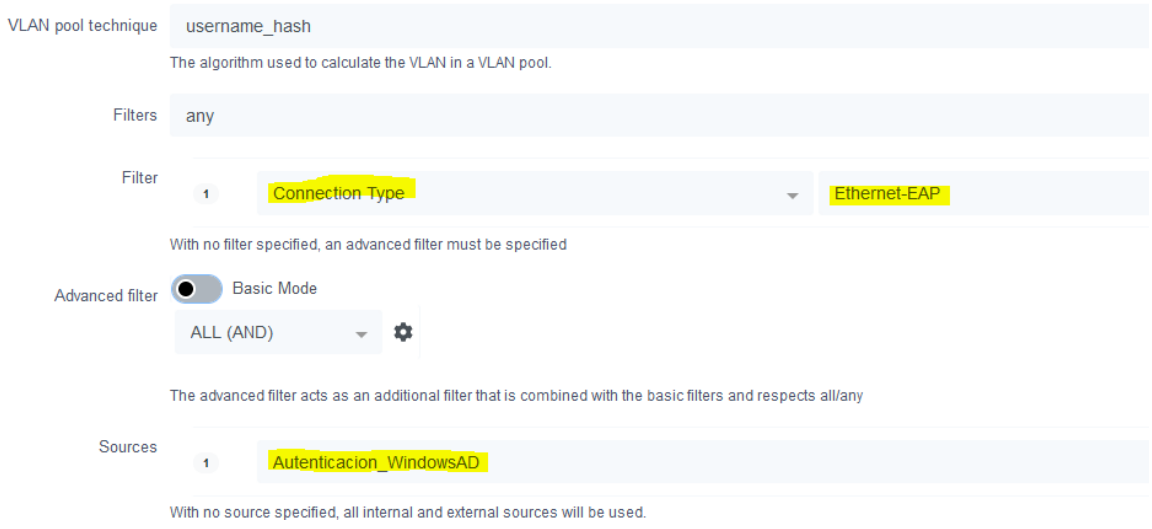


- *Paso 12*: Tras comprobar que los datos son correctos y verificar que la conexión se lleva a cabo (gracias al botón "test" incluido en la pantalla anterior), ya podríamos hacer uso del método de autenticación basado en AD. Para establecerlo, debemos acceder a la pantalla "*Configuration → Policies and Access Control → Realms",* editar los realms "*DEFAULT"* y "*NULL",* y en la opción NTLM Auth, establecer el dominio creado anteriormente. Gracias a ello, indicamos a PacketFence que haga uso de la base de datos del AD como método de autenticación predeterminado. Con los cambios realizados, la pantalla principal de configuración de "*Realms*" muestra el siguiente resultado…

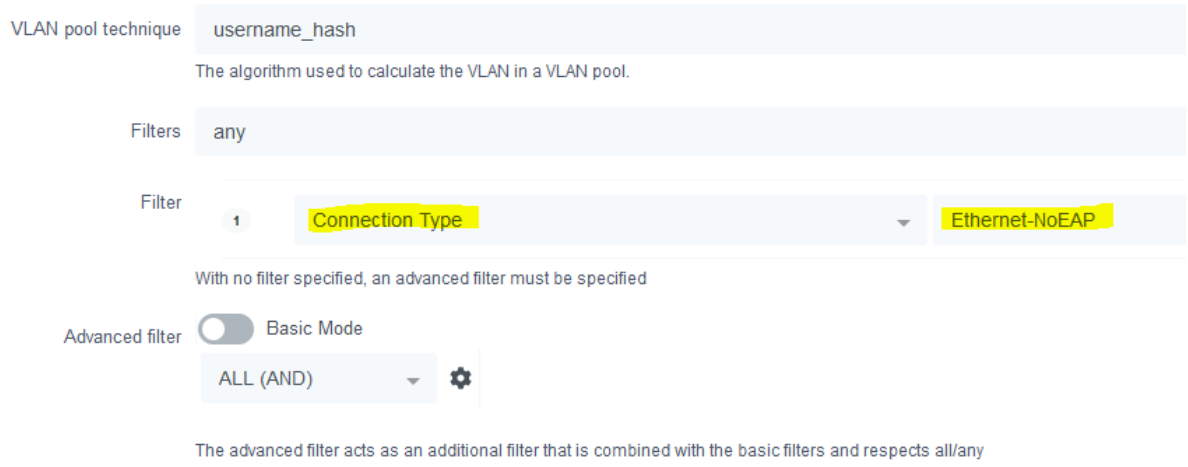


2.4. Configuración de perfiles de conexión 802.1x y MAB

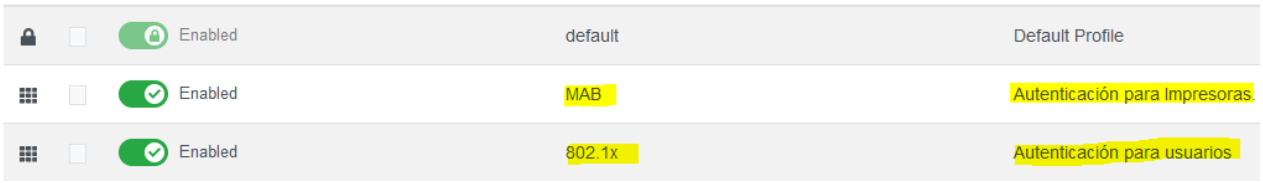
- *Paso 13*: El siguiente paso consistirá en configurar los métodos de autenticación permitidos, los cuales se definen en el apartado "*Configuration → Policies and Access Control → Connection Profiles".* Como se ha mencionado a lo largo del proyecto, se permitirán dos métodos, 802.1x y MAB, por tanto, deberemos crear dos perfiles de conexión, uno para cada uno de ellos. Para 802.1x, bastará con permitir únicamente el tipo de conexión EAP e indicar que la fuente de autenticación es el AD creado anteriormente…



Mientras que para MAB bastará con indicar que la autenticación no será EAP. En este caso no definiremos la fuente de autenticación, ya que las MAC permitidas se indicarán posteriormente en el perfil de acceso creado para ello…

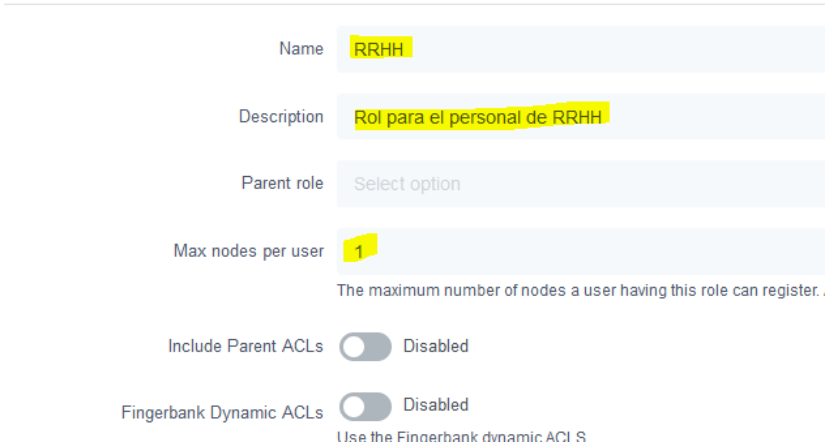


Una vez hecho, la pantalla principal deberá mostrar ambos perfiles de conectividad…

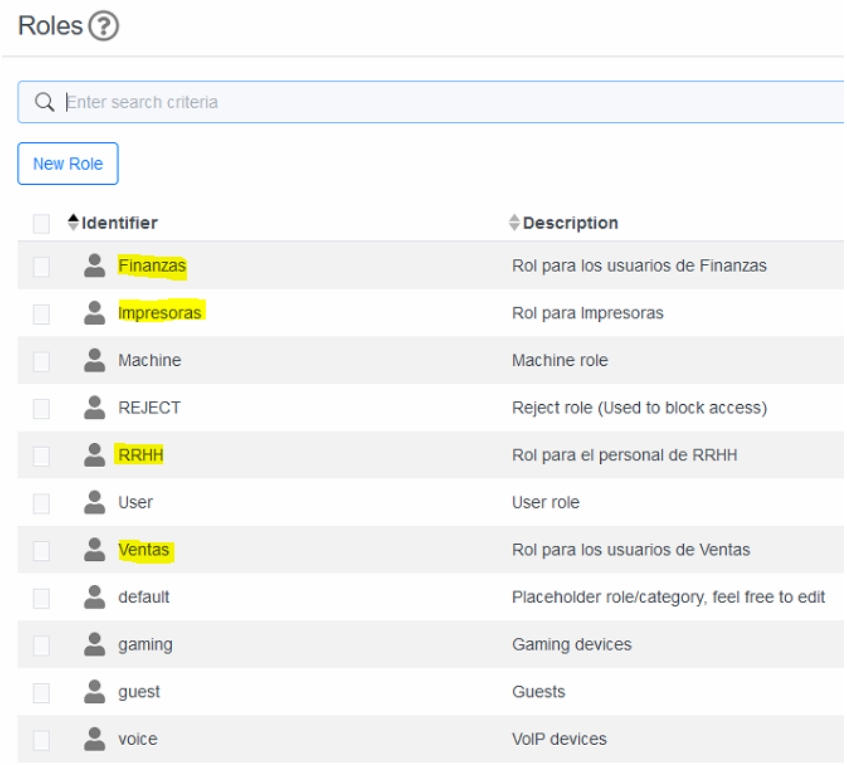


2.5. Configuración y asignación de roles y reglas de acceso

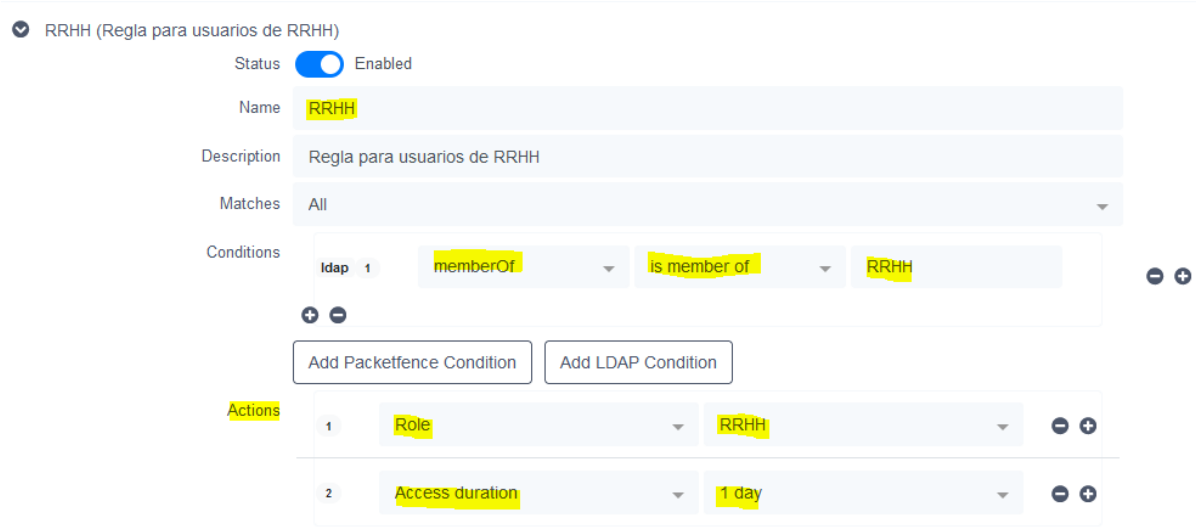
- *Paso 14*: Para lograr que a cada tipo de usuario se le asigne un determinado perfil de acceso debemos crear un rol para cada uno de ellos. Esta tarea se lleva a cabo desde la pantalla "*Configuration → Policies and Access Control → Roles",* en la cual podremos comprobar que ya existen varios definidos por defecto. La creación de uno nuevo se logra mediante el botón "*New Role",* desde el cual se abrirá una nueva pantalla con todas las opciones disponibles. Para este caso, bastará con asignarle un nombre, descripción, y un máximo de un usuario por nodo, es decir, que el mismo usuario no pueda utilizar más de un dispositivo de forma simultánea. Un ejemplo es el siguiente, que será utilizado para el personal de RRHH.



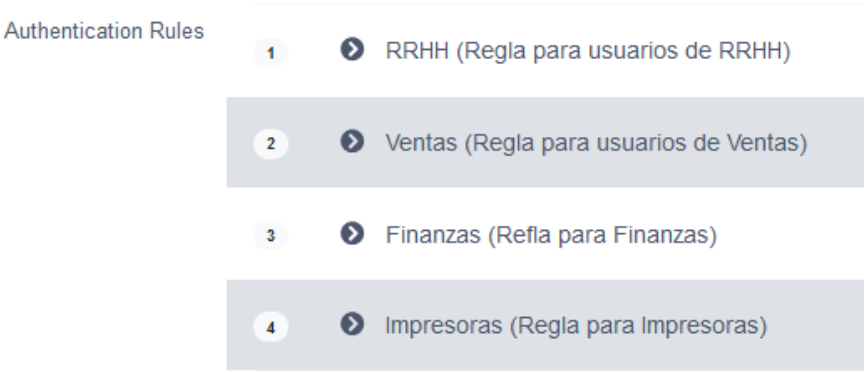
Una vez creados todos los roles, los podremos verificar desde la pantalla principal…



- *Paso 15*: Tras los roles, se deben configurar las reglas de acceso, es decir, definir los requisitos que se deben cumplir para que, cuando un usuario se autentique, se le asigne un rol específico. Como se ha decidido durante el diseño de la solución, este requisito será el grupo del AD al que pertenezca el usuario, mientras que, para el caso de las impresoras, se indicará directamente su MAC. Para lograrlo deberemos editar la fuente de autenticación creada en el paso 11, y en la opción "*Authentication rules",* crear cada uno de las reglas necesarias. Un ejemplo es el siguiente, donde se indica que, si el usuario pertenece al grupo RRHH, se le asigne el perfil de acceso (rol) de RRHH…



Cabe mencionar que, para los usuarios, la condición está basada en LDAP, mientras que para las impresoras lo debe estar en local, ya que simplemente se analizará la MAC. En cuanto a la duración, es un campo obligatorio que indica el tiempo que debe pasar antes de volver a realizar la validación del usuario o dispositivo. Con ello, se han creado las siguientes reglas de autenticación…

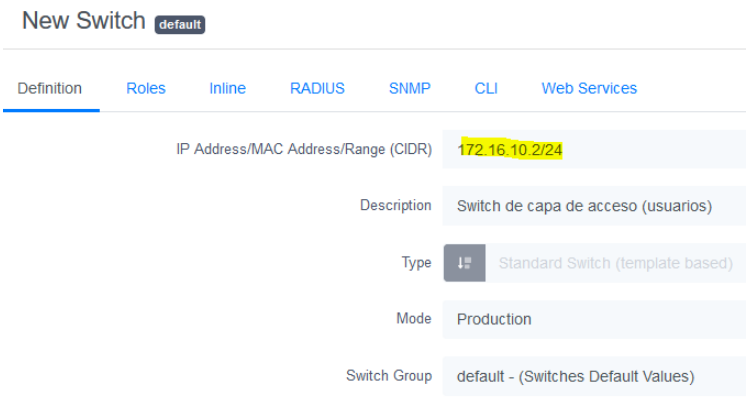


2.6. Integración del Switch y del punto de acceso

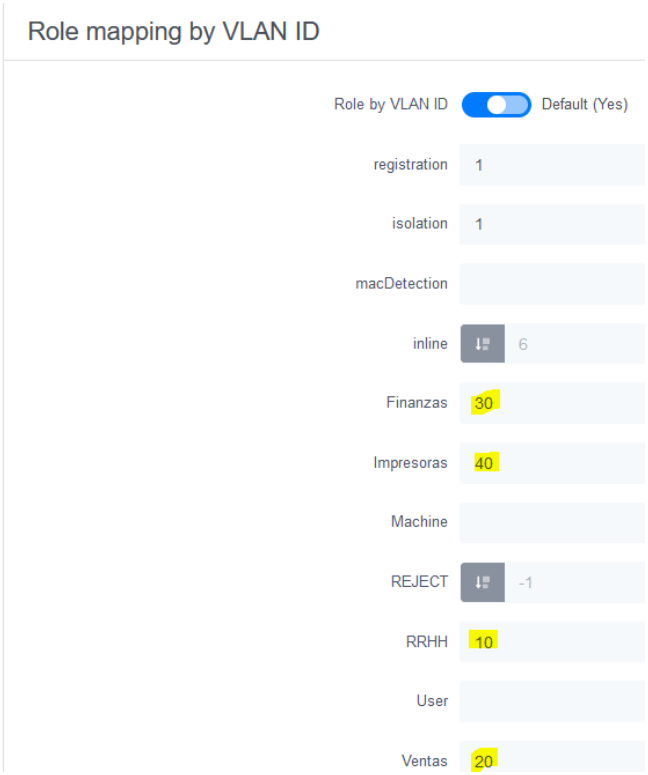
- *Paso 16*: Por último, se debe integrar el Switch y el punto de acceso con el PacketFence, para que este los identifique como origen de autenticación y pueda gestionarlos, es decir, aplicar la VLAN correcta a cada rol de usuario. Esta acción la llevamos a cabo desde la pantalla "*Configuration → Policies and Access Control → Switches".* Al agregar cada uno de ellos tendremos disponibles varias pestañas, donde las más importante para nuestro objetivo son "*Definition*", "*Roles*" y "*Radius*". (Solo se muestra la configuración del Switch por brevedad)

  - En *Definition* indicaremos las opciones generales del dispositivo, como su IP, tipo o modo. El tipo lo marcaremos como *standard*, mientras que el

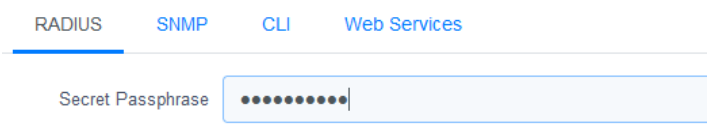
modo será en producción, con el fin de asemejar el laboratorio lo más posible a un entorno real.



- En *Roles* debemos indicar qué VLAN asignamos a cada rol



- Por último, en la pestaña *Radius* es importante indicar la clave secreta para establecer la comunicación entre ambos sistemas.



Anexo 3: Configuración del Switch (Enterasys B5)

- *Paso 1*: Crear las diferentes VLAN…

```
TFG-DANIELPEREZ(su)->set vlan create 10
TFG-DANIELPEREZ(su)->set vlan name 10 "RRHH"
TFG-DANIELPEREZ(su)->set vlan create 20
TFG-DANIELPEREZ(su)->set vlan name 20 "VENTAS"
TFG-DANIELPEREZ(su)->set vlan create 30
TFG-DANIELPEREZ(su)->set vlan name 30 "FINANZAS"
TFG-DANIELPEREZ(su)->set vlan create 40
TFG-DANIELPEREZ(su)->set vlan name 40 "IMPRESORAS"
```

- *Paso 2*: Configurar los enlaces troncales permitiendo las VLAN necesarias. Para el caso que tratamos, son necesarios los siguientes…

  - Un enlace, que conecta con el Firewall y que deberá permitir el acceso de todas las VLAN, es decir, la 10, 20, 30 y 40. Se utilizará el puerto *ge.1.24* para ello.
  - Otro enlace, que conectará con el punto de acceso. Como solo se permitirá la conectividad inalámbrica a usuarios (no a impresoras) bastará con permitir las VLAN 10, 20 y 30 a través del mismo. Se utilizará la interfaz *ge.1.23* para ello.

```
TFG-DANIELPEREZ(su)->set vlan egress 10,20,30,40 ge.1.24 tagged
TFG-DANIELPEREZ(su)->set port alias ge.1.24 "ENLACE_FIREWALL"
TFG-DANIELPEREZ(su)->set vlan egress 10,20,30 ge.1.23 tagged
TFG-DANIELPEREZ(su)->set port alias ge.1.24 "ENLACE_ACCESS_POINT"
```

- Paso 3: Definir la conectividad y características con el servidor RADIUS, que en este caso será el NAC, con IP 10.0.100.10. Las características que se consideran necesarias son las siguientes:

  - Habilitar el Switch como cliente Radius mediante el comando "*set radius enable".*
  - Definir la dirección IP y puerto del servidor Radius utilizado para recibir dicha comunicación. Como ya sabemos, el servidor será el NAC, con IP 10.0.100.10, mientras que el puerto utilizado por Radius para la autenticación es el UDP 1812. En este paso nos solicitará la clave secreta configurada en el servidor Radius anteriormente.
  - Indicar en qué casos se requerirá la autenticación, pudiendo ser para la conectividad de usuarios, para la conectividad de administración del Switch, o para ambos. Para el caso que tratamos, solo nos interesa la autenticación para usuarios, lo cual logramos con el parámetro "*network-access".*
  - Definir la interfaz que se utilizará para enviar las peticiones al servidor Radius. En este caso se configurará la IP 172.16.10.2 en la VLAN 10 para este propósito.

  Resumiendo, cuando un usuario se conecte a través de los puertos definidos para ello (se configurarán en el próximo paso), el switch hará uso de la IP

172.16.10.2 para comunicar con el servidor Radius y verificar las credenciales del usuario en cuestión.

```
TFG-DANIELPEREZ(su)->set radius enable
TFG-DANIELPEREZ(su)->set radius server 1 10.0.100.10 1812
Enter secret:
Re-enter secret:
TFG-DANIELPEREZ(su)->set radius realm network-access all
```

```
TFG-DANIELPEREZ(su)->router
TFG-DANIELPEREZ(su)->router>enable
TFG-DANIELPEREZ(su)->router#configure terminal
Enter configuration commands:
TFG-DANIELPEREZ(su)->router(Config)#interface vlan 10
TFG-DANIELPEREZ(su)->router(Config-if(Vlan 10))#ip address 172.16.10.2 255.255.255.0
TFG-DANIELPEREZ(su)->router(Config-if(Vlan 10))#exit
TFG-DANIELPEREZ(su)->router(Config)#exit
TFG-DANIELPEREZ(su)->router#exit
TFG-DANIELPEREZ(su)->router>exit
TFG-DANIELPEREZ(su)->
TFG-DANIELPEREZ(su)->set radius interface vlan 10
```

- *Paso 4*: Configurar y forzar la autenticación del usuario en los puertos del Switch antes de habilitar la conexión a la red. Como se ha comentado en capítulos anteriores, el switch deberá albergar dos tipos de autenticación, 802.1x y MAB, por tanto, los puertos deberán ser configurados para ello. Sin embargo, mediante MAB solo se podrán conectar las impresoras, gracias al filtro de MAC definido anteriormente en el servidor Radius. Con todo ello, en este paso primero habrá que configurar las características de 802.1x y MAB, para luego aplicarlo sobre las interfaces. Se deben cumplir los siguientes requisitos:

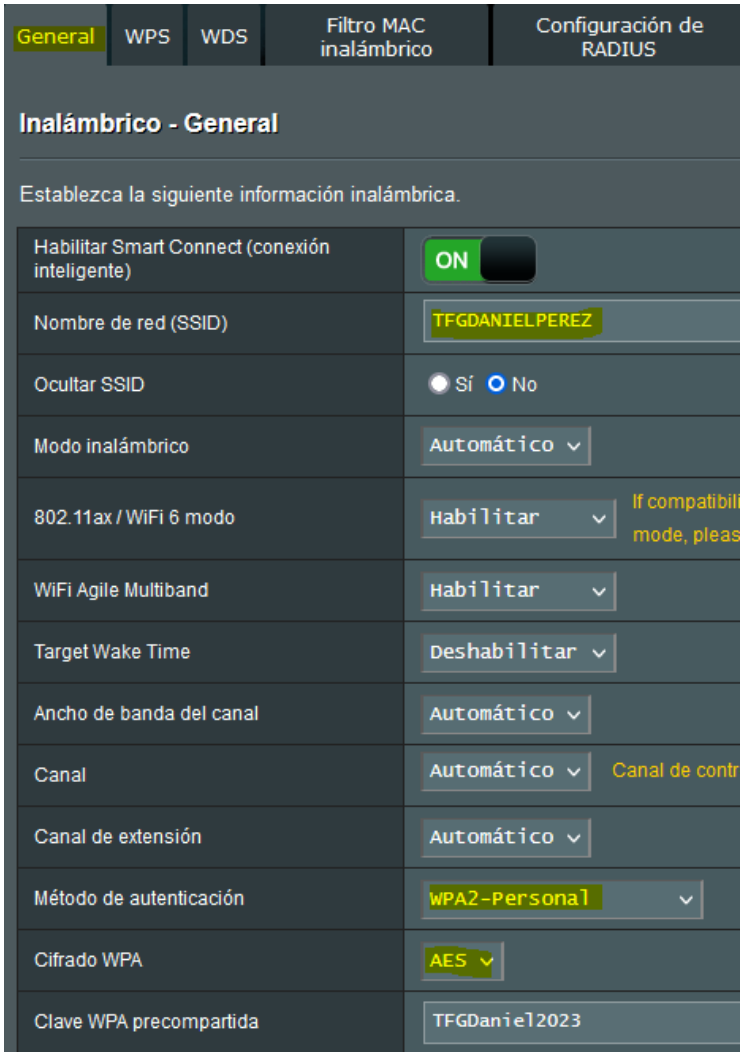
  - Se debe habilitar 802.1x, EAPOL y MAB de manera global en el switch, para posteriormente aplicarlos de manera particular en cada interfaz
  - Permitir dos métodos de autenticación en cada puerto, que se ejecutarán por orden, primero 802.1x y, si esta falla, MAB.
  - Permitir la conexión de solo un usuario por puerto.
  - Se forzará la autenticación en el rango de puertos 1-20. El 21 y 22, se dejarán libres para futuros enlaces hacia otros dispositivos de red. El 23 conecta con el punto de acceso, el cual gestionará la autenticación de los usuarios inalámbricos, mientras que el 24 es el enlace de comunicación con el Firewall.

```
TFG-DANIELPEREZ(su)->set dot1x enable
TFG-DANIELPEREZ(su)->set eapol enable
TFG-DANIELPEREZ(su)->set macauthentication enable
TFG-DANIELPEREZ(su)->set multiauth mode multi
TFG-DANIELPEREZ(su)->set multiauth precedence dot1x mac
```

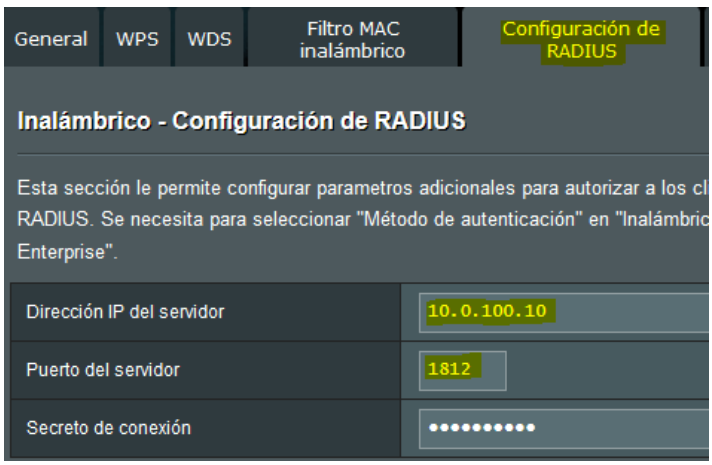
```
TFG-DANIELPEREZ(su)->set multiauth port mode auth-reqd ge.1.1-20
TFG-DANIELPEREZ(su)->set multiauth port numusers 1
```

Anexo 4: Configuración del punto de acceso (Asus RT-AX59U)

- *Paso 1*: Habilitar la red Wifi, con método de autenticación WPA2-Personal y cifrado WPA AES…

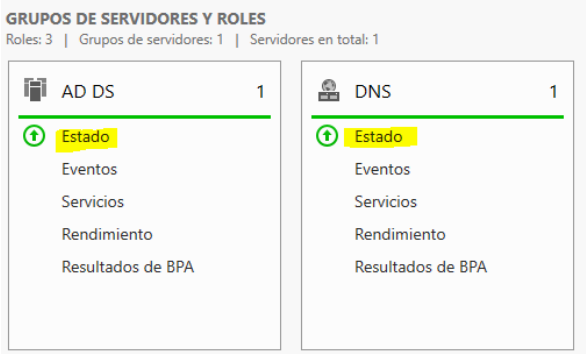


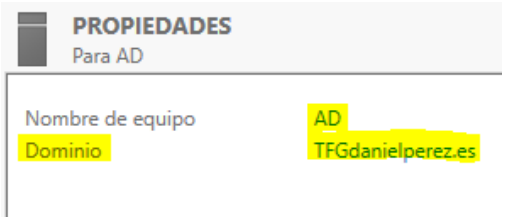
- *Paso 2*: Definir el servidor Radius y puerto al cual se enviarán las peticiones de autenticación.



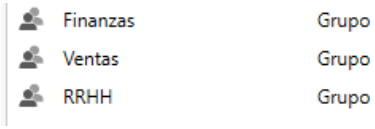
Anexo 5: Configuración del directorio activo (Windows 2019)

- *Paso 1*: Habilitar y configurar los servicios AD DS, DNS, y configurar un dominio en el AD…

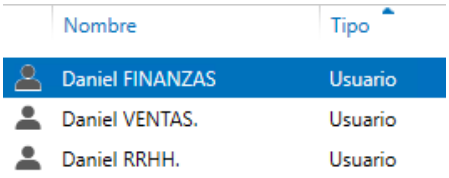


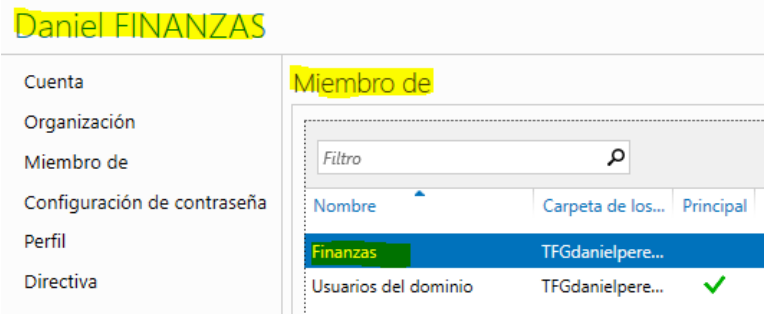


- *Paso 2*: Crear un grupo para cada uno de los perfiles de acceso…



- *Paso 3*: Crear usuarios de prueba y agregarlos al grupo correspondiente





Anexo 6: Código del fichero nacApp.py

```
import os
import tkinter.filedialog
from flask import Flask, render_template, request, redirect
import paramiko
from tkinter import messagebox
import ipaddress

import administracion.add_new_switch
import administracion.add_new_rol
import monitorizacion.blocked_access
import monitorizacion.users
import troubleshooting.logs_download
import troubleshooting.sniffer
import nacApp

app = Flask(__name__)

check_access = ""


#-------------------------PÁGINA PRINCIPAL (VALIDACIÓN DE USUARIO)------------------

@app.route('/', methods=["GET", "POST"])
def inicioSesion():
    nacApp.check_access = "False"
    if request.method == "POST":
        if request.form.get("uname") == "admin" and request.form.get("psw") == "admin":
            nacApp.check_access = str("True")
            return redirect("/main")
        else:
            messagebox.showinfo("Error","Usuario y/o contraseña incorrecta.")
    return render_template('login.html')

#--------------------------MENÚ PRINCIPAL----------------------------------------

@app.route('/main')
def main():
    if check_access == "True":
        return render_template('main.html')
    return redirect('/')


#---------------------MENÚ Y FUNCIONES DE ADMINISTRACION-----------------------------

@app.route('/admin')
def administration():
    if check_access == "True":
        return render_template('administration.html')
    return redirect('/')


@app.route ('/admin/new-switch', methods=["GET", "POST"])
def newSwitch():
    if check_access == "True":
        if request.method == "POST":
            confirm = messagebox.askquestion("Confirmar...","Se va a proceder a
                                        configurar el Switch. ¿Desea continuar?")
            if confirm == "yes":
                ip_address = request.form.get("ip_switch")
                try:
                    ipaddress.ip_network(ip_address)
                except ValueError:
```

```
                    messagebox.showinfo("Error", "Formato de IP incorrecto. se debe
                                        introducir un rango en formato IPv4")
                    return redirect('/admin/new-switch')

                resultado_switch =
str(administracion.add_new_switch.new_switch(ip_address))
                resultado_nac =
str(administracion.add_new_switch.add_switch_to_nac(ip_address))
                return render_template("new_switch_result.html",
switch=resultado_switch,nac=resultado_nac)
            else:
                return redirect('/admin')
        return render_template("new_switch.html")
    return redirect('/')
```

@app.route ('/admin/new-rol', methods=["GET", "POST"])

```
def newRol():
    if check_access == "True":
        if request.method == "POST":
            confirm  =  messagebox.askquestion("Confirmar...","Se  va  a  proceder  a
configurar el nuevo perfil de acceso. ¿Desea continuar?")
            if confirm == "yes":
                rolName = request.form.get("rol_name")
                vlanId = request.form.get("vlan_id")
                networkId = request.form.get("network")

# Control de errores en los datos introducidos. Se verifica que el nombre del perfil
de acceso no sea numérico, que el ID de vlan sí sea numérico y además esté comprendido
entre los valores 1 y 4094, y que el ID de red tenga el formato correcto y que además
termine en .0

                if rolName.isnumeric() == True:
                    messagebox.showinfo("Error", "El nombre del perfil de acceso no
                                        debe ser numérico")
                    return redirect('/admin/new-rol')

                if vlanId.isnumeric() == False:
                    messagebox.showinfo("Error", "El ID de VLAN debe ser un valor
                                        numérico en el rango 1-4094")
                    return redirect('/admin/new-rol')
                vlanId = int(vlanId)
                if vlanId < 1 or vlanId > 4094:
                    messagebox.showinfo("Error", "El ID de VLAN debe ser un valor
                                        numérico en el rango 1-4094")
                    return redirect('/admin/new-rol')
                try:
                    ipaddress.ip_network(networkId)
                except ValueError:
                    messagebox.showinfo("Error", "Formato de ID incorrecto. se debe
                                        introducir un rango en formato IPv4")
                    return redirect('/admin/new-rol')
                networkId_len = len(networkId)
                if networkId[networkId_len - 1] != "0" or networkId[networkId_len -
2] != ".":
                    messagebox.showinfo("Error","El rango de ID de red debe terminar
en .0")
                    return redirect('/admin/new-rol')

# Verificar si el rol ya existe, en cuyo caso se cancela el proceso

                nac_packetfence = paramiko.SSHClient()
                nac_packetfence.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                nac_packetfence.connect("10.0.100.10", port=22, username="root",
                                        password="pass")
```

```
                stdin, stdout, stderr = nac_packetfence.exec_command('cat
                                        /usr/local/pf/conf/roles.conf')
                check_rol = str(stdout.readlines())
                if rolName in check_rol:
                    messagebox.showinfo("Error", "El rol que está intentando crear ya
                                        existe en el servidor NAC.")
                    nac_packetfence.close()
                    return redirect('/admin/new-rol')
                nac_packetfence.close()

# Con los datos validados, se ejecutan las funciones que llevarán a cabo cada una de
las configuraciones necesarias en los diferentes dispositivos. Cada resultado se
almacena en una variable, que contendrá el resultado de la ejecución de la función. Si
la función ha finalizado correctamente, se almacenará un mensaje de éxito, mientras que
si se ha producido algún error, se almacenará el motivo por el cual se generó dicho
error. (ver script add_new_rol.py)

                resultado_switch  =  str(administracion.add_new_rol.switchConf(vlanId,
                                        rolName))
                resultado_firewall =
str(administracion.add_new_rol.firewallConf(vlanId, rolName, networkId))
                resultado_ad = str(administracion.add_new_rol.adConf(rolName))
                resultado_nac = str(administracion.add_new_rol.nacRolConf(rolName))

# Con las funciones ejecutadas, se muestra el resultado de todas ellas en una nueva
web, donde las variables son pasadas como parámetros para poder ser mostradas en formato
HTTP (ver código del fichero new_rol_result.html)

                return render_template("new_rol_result.html",
                                       switch=resultado_switch,
                                       firewall=resultado_firewall, ad=resultado_ad,
                                       nac=resultado_nac)

            else:
                return redirect('/admin')
        return render_template("new_rol.html")
    return redirect('/')



#-----------------------MENÚ Y FUNCIONES DE MONITORIZACION-------------------------

@app.route('/monitor')
def monitor():
    if check_access == "True":
        return render_template('monitor.html')
    return redirect('/')

@app.route('/monitor/cUsers')
def cUsers():
    if check_access == "True":
        connectedusers = monitorizacion.users.connectedUsers()
        return render_template('cUsers_result.html', connected=connectedusers)
    return redirect('/')


@app.route('/monitor/dUsers')
def dUsers():
    if check_access == "True":
        deniedusers = monitorizacion.users.deniedUsers()
        return render_template('dUsers_result.html', denied=deniedusers)
    return redirect('/')

@app.route('/monitor/blocked')
def blockedAccess():
```

```
    if check_access == "True":
        result = monitorizacion.blocked_access.blockedAccess()
        return render_template('blocked_access_result.html', blocked_access=result)
    return redirect('/')


#--------------------MENÚ Y FUNCIONES DE TROUBLESHOOTING---------------------------

@app.route('/tshoot')
def tshoot():
    if check_access == "True":
        return render_template('tshoot.html')
    return redirect('/')


@app.route('/tshoot/sniffer', methods=["GET", "POST"])
# Al acceder a esta ruta, se mostrará el formulario creado en la página sniffer.html,
el cual solicitará al usuario el número de segundos que durará la captura de paquetes.
Una vez introducidos, se lleva a cabo el control de los datos introducidos (debe ser
un número entero comprendido entre 1 y 10), y si estos son correctos, se solicita al
usuario la ubicación para almacenar el fichero descargado. Estos datos (segundos y
ubicación) son enviados como parámetros a la función sniffer, ubicada en el fichero
sniffer.py, la cual llevará a cabo la conexión SSH con el servidor NAC, realizará la
captura y se almacenará el directorio local seleccionado por el usuario. Tras ello, si
la descarga se ha hecho correctamente, se preguntará al usuario si desea abrir el
fichero, mientras que, si se ha producido algún error, se notificará al usuario.

def sniffer():
    if check_access == "True":
        if request.method == "POST":
            seconds = str(request.form.get("seconds"))
            if seconds.isnumeric() == False:
                messagebox.showinfo("Error", "El tiempo introducido debe ser un valor
                                    numérico en el rango [1-10]")
                return redirect('/tshoot/sniffer')
            seconds = int(seconds)
            if seconds < 1 or seconds > 10:
                messagebox.showinfo("Error", "El tiempo introducido debe ser un valor
                                    numérico en el rango [1-10]")
                return redirect('/tshoot/sniffer')
            folder_to_download = tkinter.filedialog.askdirectory(title="Seleccione la
                                 carpeta donde se almacenará el fichero")
            result = str(troubleshooting.sniffer.sniffer(seconds,folder_to_download))
            if 'pcap' in result:
                open = messagebox.askquestion("Continuar...", "El fichero se ha
                                              descargado con éxito. ¿Desea abrirlo?")
                if open == "yes":
                    os.startfile(result)
                    return redirect('/tshoot/sniffer')
                else:
                    return redirect('/tshoot/sniffer')
            else:
                messagebox.showinfo("Error", "Se ha producido un error al conectar
                                    con el servidor. Por favor, compruebe la
                                    conectividad y los permisos de la carpeta
                                    seleccionada y vuelva a intentarlo.")
        return render_template('sniffer.html')
    return redirect('/')


@app.route('/tshoot/logs', methods=["GET", "POST"])
# Cuando un usuario acceda a esta ruta, se mostrará una lista desplegable con las
diferentes opciones de log a descargar (fichero logs_download.html). Con la opción
seleccionada, se solicitará indicar la carpeta donde se descargará el fichero y se
```

llamará a la función getLogs, pasando dicha ubicación como parámetro. Gracias a ello, esta función descargará el fichero en el equipo local (ver código del script logs_download.py). Una vez descargado, se preguntará al usuario si desea abrir el fichero. En caso positivo, se abrirá, y en caso negativo, se volverá a la pantalla de descarga de logs. Si se produce algún error, se mostrará un aviso de conexión fallida al usuario.

```
def getlogs():
    if check_access == "True":
        if request.method == "POST":
            log_to_download = str(request.form.get("id_log"))
            folder_to_download = tkinter.filedialog.askdirectory()
            file_to_download =
str(troubleshooting.logs_download.getLogs(log_to_download,folder_to_download))
            if 'log' in file_to_download:
                open = messagebox.askquestion("Continuar...","El fichero se ha
descargado con éxito. ¿Desea abrirlo?")
                if open == "yes":
                    os.startfile(file_to_download)
                    return redirect('/tshoot/logs')
                else:
                    return redirect('/tshoot/logs')
            else:
                messagebox.showinfo("Error", "Se ha producido un error al conectar
con el servidor. Por favor, "
                                             "compruebe la conectividad y los
permisos de la carpeta seleccionada y vuelva a intentarlo.")
                return redirect('/tshoot/logs')
        return render_template('logs_download.html')
    return redirect('/')
```

Anexo 7: Código de la página de validación y menús principales

7.1. Ruta "/": Página de validación - Fichero *login.html*

```
#Contenido del fichero login.html, el cual contiene el formulario de validación de
usuario que será cargado al acceder a la ruta "/" de la aplicación.
<!DOCTYPE html>
<html>
  <head>
    <title>Acceso a la aplicación</title>
    <link        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
  </head>
  <body>
    <form action="{{ url_for('inicioSesion')}}" method="post">
      <h1>Administración NAC de TFGDanielPerez</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <label for="uname"><strong>Nombre de usuario</strong></label>
        <input type="text" placeholder="Usuario" name="uname" required>
        <label for="psw"><strong>Contraseña</strong></label>
        <input type="password" placeholder="Contraseña" name="psw" required>
        </div>
      </div>
      <button type="submit"><b>Iniciar sesión</b></button>
    </form>
  </body>
</html>
```

7.2. Ruta "/main": Menú principal - Fichero *main.html*

```
#Contenido del fichero main.html, el cual contiene el menú principal de la aplicación,
y que será mostrado una vez el usuario se haya validado. Este fichero también se carga
desde la ruta "/" del fichero nacApp.py
<!DOCTYPE html>
<html>
<head>
    <title>Menú principal</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
  </head>
  <body>
  <form>
      <h1>Menú principal</h1>
      <div class="formcontainer">
           <a href="/admin"><button type="button"><b>Administración</b> </button></a>
          <a href="/monitor"><button type="button"> <b> Monitorización </b> </button> </a>
          <a href="/tshoot"><button type="button"><b>Troubleshooting</b> </button></a>
      </div>
      <center><a href="/">Volver | </a><a href="/"> Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

7.3. Ruta "/admin": Menú de funciones de administración - Fichero *administration.html*

```
#Contenido del fichero administration.html, el cual contiene las opciones de
administración. Será cargado al acceder a la ruta "/admin" de la aplicación.
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Menú funciones de administracion</title>
    <link        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
</head>
 <body>
 <form>
     <h1>Menú principal</h1>
     <div class="formcontainer">
         <a href="/admin"><button style="background-color: #0080FF;"
type="button"><b>Administración</b></button></a>
         <a href="/admin/new-switch"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Crear nuevo Switch</b></button></a>
         <a href="/admin/new-rol"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Crear nuevo Rol</b></button></a>
         <a href="/monitor"><button type="button"><b>Monitorización</b></button></a>
         <a href="/tshoot"><button type="button"><b>Troubleshooting</b></button></a>
     </div>
      <center><a href="/main">Volver | </a><a href="/"> Cerrar sesión </a></center>
   </form>
 </body>
</html>
```

7.4. Ruta “/monitor”: Menú de funciones de monitorización - Fichero *monitor.html*

```
#Contenido del fichero monitor.html, el cual contiene las opciones de monitorización.
Será cargado al acceder a la ruta “/monitor” de la aplicación.
<!DOCTYPE html>
<html>
<head>
    <title>Menú funciones de monitorización</title>
    <link        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
  </head>
  <body>
  <form>
      <h1>Menú principal</h1>
      <div class="formcontainer">
          <a href="/admin"><button type="button"><b>Administración</b></button></a>
          <a href="/monitor"><button style="background-color: #0080FF;"
type="button"><b>Monitorización</b></button></a>
          <a href="/monitor/cUsers"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Usuarios conectados</b></button></a>
          <a href="/monitor/dUsers"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Usuarios denegados</b></button></a>
          <a href="/monitor/blocked"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Accesos bloqueados</b></button></a>
          <a href="/tshoot"><button type="button"><b>Troubleshooting</b></button></a>
      </div>
      <center><a href="/main">Volver | </a><a href="/"> Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

7.5. Ruta “/tshoot”: Menú de funciones de troubleshooting - Fichero *tshoot.html*

```
#Contenido del fichero tshoot.html, el cual contiene las opciones de troubleshooting.
Será cargado al acceder a la ruta “/tshoot” de la aplicación.
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Menú funciones de troubleshooting</title>
    <link           href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
  </head>
  <body>
  <form>
      <h1>Menú principal</h1>
      <div class="formcontainer">
          <a href="/admin"><button type="button"><b>Administración</b></button></a>
          <a href="/monitor"><button type="button"><b>Monitorización</b></button></a>
          <a href="/tshoot"><button style="background-color: #0080FF;"
type="button"><b>Troubleshooting</b></button></a>
          <a href="/tshoot/sniffer"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Captura de paquetes</b></button></a>
          <a href="/tshoot/logs"><button style="background-color: #A4A4A4; color: black"
type="button"><b>Descarga de logs</b></button></a>
      </div>
      <center><a href="/main">Volver | </a><a href="/"> Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

Anexo 8: Código y ficheros de la función Agregar un nuevo Switch

Estos ficheros son los utilizados al acceder a la ruta *"/admin/new-switch"* de la aplicación. (ver también el código de esta ruta en el fichero *nacApp.py*).

8.1. Formulario HTML – Fichero *new_switch.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formulario para agregar un nuevo Switch</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">

  </head>
  <body>
    <form action="{{ url_for('newSwitch')}}" method="post">
      <h1>Configurar un nuevo Switch en NAC_TFG</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <label for="ip_switch"><strong>IPv4 del Switch</strong></label>
        <input type="text" placeholder="Ipv4" name="ip_switch" required>
        </div>
      </div>
      <button type="submit"><b>Agregar Switch</b></button>
      <center><a href="/admin">Volver | </a><a href="/"> Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

8.2. Script Phyton – Fichero *add_new_switch.py*

```
import time
import paramiko
import jinja2.exceptions

def new_switch(ip_add):
"""Se establece la conexión SSH con la IP del Switch recibida por parámetro y se configura todo lo necesario para incluir al dispositivo en el entorno NAC. Es decir, se configuran y propagan todas las VLAN, se define el servidor Radius sobre el cual se autenticarán los clientes y se aplica el tipo de autenticación en 8021.x y MAB en todos los puertos destinados a usuarios. NOTA: La pausa de 0,5 segundos entre el envío de comandos al Switch ha sido necesaria incluirla porque de los contrario el Switch no era capaz de ejecutar dichos comandos a la velocidad que los enviaba el script"""
    try:
        switch = paramiko.SSHClient()
        switch.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        switch.connect("ip_add", port=22, username="admin", password=("pass"))
        switch_channel = switch.invoke_shell()
        # CREAR VLANS
        time.sleep(0.5)
        switch_channel.send('set vlan create 10\n')
        time.sleep(0.5)
        switch_channel.send('set vlan name 10 "RRHH"\n')
        time.sleep(0.5)
        switch_channel.send('set vlan create 20\n')
        time.sleep(0.5)
        switch_channel.send('set vlan name 20 "VENTAS"\n')
        time.sleep(0.5)
        switch_channel.send('set vlan create 30\n')
        time.sleep(0.5)
```

```
        switch_channel.send('set vlan name 30 "FINANZAS"\n')
        time.sleep(0.5)
        switch_channel.send('set vlan create 40\n')
        time.sleep(0.5)
        switch_channel.send('set vlan name 40 "IMPRESORAS"\n')
        time.sleep(0.5)
        switch_channel.send('set vlan egress 10,20,30,40 ge.1.24 tagged\n')
        time.sleep(0.5)
        switch_channel.send('set port alias ge.1.24 "ENLACE_FIREWALL"\n')
        time.sleep(1)
        # COMUNICACION CON EL SERVIDOR RADIUS
        switch_channel.send('set radius enable\n')
        time.sleep(0.5)
        switch_channel.send('set radius server 1 10.0.100.10 1812\n')
        time.sleep(0.5)
        switch_channel.send('Passw0rd\n')
        time.sleep(0.5)
        switch_channel.send('Passw0rd\n')
        time.sleep(0.5)
        switch_channel.send('set radius realm network-access all\n')
        time.sleep(0.5)
        switch_channel.send('set radius interface lookback 1\n')
        #CONFIGURAR AUTENTICACIÓN
        time.sleep(0.5)
        switch_channel.send('set dot1x enable\n')
        time.sleep(0.5)
        switch_channel.send('set eapol enable\n')
        time.sleep(0.5)
        switch_channel.send('set macauthentication enable\n')
        time.sleep(0.5)
        switch_channel.send('set multiauth mode multi\n')
        time.sleep(0.5)
        switch_channel.send('set multiauth precedence dot1x mac\n')
        time.sleep(0.5)
        switch_channel.send('set multiauth port mode auth-reqd ge.1.1-20\n')
        time.sleep(0.5)
        switch_channel.send('set multiauth port numusers 1\n')
        time.sleep(1)
        switch_channel.close()
        switch.close()
        return str('¡¡ Configuración del Switch finalizada con éxito !!')

    except:
        return str('Se ha producido un error durante la conexión al Switch. Por favor, revisar la
                    configuración de SSH en el Switch. Se ha cancelado el proceso.\n')
        quit()
```

```
def add_switch_to_nac(ip_add):
"""Se establece la conexión SSH con el servidor NAC y se agrega el nuevo y se agrega el nuevo Switch
a su configuración, lo cual se lleva a cabo agregando las líneas necesarias en el fichero switches.conf.
Tras ello, se reinicia el servicio Radius para que los cambios se apliquen. La IP del nuevo Switch es
recibida por parámetro cuando se llama a la función (ver código del fichero nacApp.py)""
    try:
        nac = paramiko.SSHClient()
        nac.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        nac.connect("10.0.100.10", port=22, username="root", password=("pass"))
        nac_channel = nac.invoke_shell()
        #-------AGREGAR SWITCH AL FICHERO DE CONFIGURACIÓN----------
        nac.exec_command("printf '\n["+ip_add+"']\nVentasVlan=20\nFinanzasVlan=30\nImpresorasVlan=40"
"\ngroup=default\nisolationVlan=1\nradiusSecret=Passw0rd\nRRHHVlan=10"
"\ndescription=Nuevo_Switch\nregistrationVlan=1\ncliPwd=Passw0rd"
"\nclieEnablePwd=Passw0rd\ncliTransport=SSH\ncliUser=admin\n' >> /usr/local/pf/conf/switches.conf")
        nac.exec_command("/etc/init.d/freeradius restart")
        nac.close()
        return str('¡¡ El servidor NAC ha sido configurado con éxito !!')
    except:
```

```
        return str('Se ha producido un error en el servidor NAC. Por favor, comprobar la
                    configuración SSH y los datos de conexión.')
        quit()
```

9.3. Plantilla HTML – Fichero *new_switch_result.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Resultado de un nuevo Switch</title>
    <link        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
  </head>
  <body>
    <form>
      <h1>Configurando el nuevo Switch...</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <p> <strong> Resultado de la configuración del Switch: </strong></p>
        <p> {{ switch }} </p><br>
        <p> <strong> Resultado de la configuración del servidor NAC: </strong></p>
        <p> {{ nac }} </p>
        </div>
      </div>
      <center><a href="/admin">Volver | </a><a href="/">  Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

Anexo 9: Código y ficheros de la función Agregar un nuevo Rol

Estos ficheros son los utilizados al acceder a la ruta *"/admin/new-rol"* de la aplicación. (ver también el código de esta ruta en el fichero *nacApp.py*).

9.1. Formulario HTML – Fichero *new_rol.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formulario para agregar un nuevo Switch</title>
    <link        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">
</head>
  <body>
    <form action="{{ url_for('newRol')}}" method="post">
      <h1>Configurar un nuevo Rol en NAC_TFG</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <label     for="rol_name"><strong>Nombre     del     nuevo     perfil     de
acceso</strong></label>
        <input type="text" placeholder="Nombre" name="rol_name" required>
        <label for="vlan_id"><strong>ID de VLAN</strong></label>
        <input type="text" placeholder="Vlan" name="vlan_id" required>
        <label for="network"><strong>ID de Red</strong></label>
        <input type="text" placeholder="Id de red" name="network" required>
        </div>
      </div>
      <button type="submit"><b>Crear nuevo perfil de acceso</b></button>
      <center><a href="/admin">Volver | </a><a href="/">  Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

9.2. Script Phyton – Fichero *add_new_rol.py*

```
import time
import paramiko
from paramiko.client import SSHClient
import ipaddress
import ldap3
from ldap3 import Server, Connection, ALL
import ldap3.core.exceptions

def switchConf(id_vlan, rol_name):
"""Se establece la conexión SSH con el Switch y se configuran la nueva VLAN, propagándola también
por los enlaces troncales. El ID de VLAN y el nombre de rol son recibidos como parámetros al llamar
a la función (ver la llamada a la función en el fichero nacApp.py)"""
    try:
        switch = paramiko.SSHClient()
        switch.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        switch.connect("172.16.0.10", port=22, username="admin", password="pass")
        switch_channel = switch.invoke_shell()
        time.sleep(0.5)
        switch_channel.send('set vlan create '+str(id_vlan)+'\n')
        time.sleep(0.5)
        switch_channel.send('set vlan name '+str(id_vlan)+' "'+str(rol_name)+'"\n')
        time.sleep(0.5)
        switch_channel.send('set vlan egress '+str(id_vlan)+' ge.1.24 tagged\n')
```

```
        switch_channel.close()
        switch.close()
        return str('¡¡ El Switch se ha configurado con éxito!!\n')
    except:
        return str('Se ha producido un error. Por favor, revisar la configuración de SSH en el
                    Switch. Se ha cancelado el proceso.\n')
        quit()
```

def firewallConf(id_vlan,rol_name,network):

```
"""Se establece la conexión mediante SSH con el firewall y posteriormente se configura la interfaz
que actuará como puerta de enlace para la nueva VLAN y el rango DHCP para los clientes del nuevo
perfil de acceso. Cabe destacar que del rango de red recibido por parámetro se crean nuevas
variables, una para crear la puerta de enlace (IP terminada en .1), y otras dos para configurar el
rango DHCP (IPs terminadas en .10 y .254)). El ID de VLAN y el nombre del nuevo perfil de acceso
también son recibidos por parámetros al llamar a la función (ver código del fichero nacApp.py"""
    try:
        firewall=paramiko.SSHClient()
        firewall.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        firewall.connect("10.0.100.1", port=22, username="admin", password="pfsense")
        #AGREGAR INTERFAZ
        interface_ip = network[:-1]+"1"
        start_dhcp_range = network[:-1]+"10"
        fin_dhcp_range = network[:-1]+"254"
        firewall.exec_command("8\n")
        time.sleep(0.5)
        firewall.exec_command("ifconfig ue0."+str(id_vlan)+" create\n")
        firewall.exec_command("ifconfig ue0."+str(id_vlan)+" "+interface_ip+" netmask 255.255.255.0
vlan "+str(id_vlan)+" vlandev ue0 description "+rol_name+"\n")
        firewall.exec_command("ifconfig ue0."+str(id_vlan)+" up\n")
        time.sleep(2)
        #CONFIGURAR DHCP
        firewall.exec_command( "printf 'class \"s_ue0."+str(id_vlan)+"\"  {\n\tmatch pick-first-
             value (option dhcp-client-identifier, hardware);\n"
            "}\nsubnet "+network+" netmask 255.255.255.0 {\n "
            "\t pool  {\n\t\toption domain-name-servers 8.8.8.8,4.4.4.4;\n\n\t\trange
              "+start_dhcp_range+" "+fin_dhcp_range+";\n\t}"
            "\n\n\toption routers "+interface_ip+";\n\toption domain-name
                \"TFGdanielperez.es\";\n\t"
            "option domain-name-servers 8.8.8.8,4.4.4.4;\n\tping-check true;\n\n}\n' >>
              /var/dhcpd/etc/dhcpd.conf")
        firewall.close()
        return str('¡¡ El Firewall se ha configurado con éxito !!\n')
    except:
        return str('Se ha producido un error. Por favor, revisar la configuración de SSH en el
                    Firewall. Se ha cancelado el proceso.\n')
        quit()
```

def adConf(rol_name):

```
"""Se establece la conexión con el servidor LDAP, haciendo uso para ello de la librería 'ldap3'. Una
vez establecida la conexión, se crea un nuevo grupo en el AD, el cual será utilizado para agregar a
los usuarios a los que se aplicará el nuevo perfil de acceso.
El nombre de rol es recibido por parámetro."""
    try:
        ad_server = ldap3.Server("10.0.100.5", use_ssl=True, get_info=ALL)
        ad_connection = ldap3.Connection(ad_server,
         'CN=Administrador,CN=users,DC=TFGdanielperez,DC=es','pass',auto_bind=True)
        new_group='CN='+rol_name+',CN=users,DC=TFGdanielperez,DC=es'
        tipo_objeto='group'
        atributos={'cn':rol_name,
                   'description': 'Grupo para el departamento '+rol_name,
                   'grouptype':'-2147483644',
                   'sAMAccountName': rol_name
                   }
        ad_connection.add(new_group,tipo_objeto,atributos)
        time.sleep(1)
        ad_connection.closed
```

```
        return str('¡¡ El grupo en el directorio activo se ha configurado con éxito !!\n')
    except:
        return str("Se ha producido un error en la creación del grupo. Se ha cancelado el proceso.")
        quit()

def nacRolConf(rol_name):
"""Se establece la conexión SSH con el servidor NAC y se agrega el nuevo rol de acceso, lo cual se
lleva a cabo mediante la modificación del fichero roles.conf. El nombre del rol es recibido por
parámetro"""
    try:
        nac_packetfence=paramiko.SSHClient()
        nac_packetfence.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        nac_packetfence.connect("10.0.100.10", port=22, username="root", password="pass")
        #AGREGAR ROL
        nac_packetfence.exec_command("printf
         '\n["+rol_name+"]\ninherit_vlan=disabled\ninherit_web_auth_url=disabled\n"
         "notes=Rol para "+rol_name+"\nmax_nodes_per_pid=1\ninherit_role=disabled\n'"
          ">> /usr/local/pf/conf/roles.conf")
        nac_packetfence.close()
        return str('¡¡ El nuevo perfil de acceso en el servidor NAC se ha configurado con éxito
                        !!\n')
    except:
        return str('Se ha producido un error. Por favor, revisar la configuración de SSH en el
                    servidor NAC. Se ha cancelado el proceso.\n')
        quit()
```

9.3. Plantilla HTML – Fichero *new_rol_result.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Resultado de agregar un nuevo Rol</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">
  </head>
  <body>
    <form>
      <h1>Configurando el nuevo perfil de acceso...</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <p> <strong> Resultado de la configuración del Switch: </strong></p>
        <p> {{ switch }} </p><br>
        <p> <strong> Resultado de la configuración del Firewall: </strong></p>
        <p> {{ firewall }} </p><br>
        <p> <strong> Resultado de la configuración del Directorio Activo: </strong></p>
        <p> {{ ad }} </p><br>
        <p> <strong> Resultado de la configuración de servidor NAC: </strong></p>
        <p> {{ nac }} </p>
        </div>
      </div>
      <center><a href="/admin">Volver | </a><a href="/">  Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

Anexo 10: Código y ficheros de las funciones Ver usuarios conectados y Ver usuarios denegados

Estos ficheros son los utilizados al acceder a la ruta *"/monitor/cUsers"* y *"/monitor/dUsers"* de la aplicación. (ver también el código de estas rutas en el fichero *nacApp.py*). Se adjuntan en el mismo anexo ya que ambos hacen uso del mismo script, mientras que la plantilla es prácticamente idéntica.

10.1. Script Phyton – Fichero *users.py*

```
import mysql.connector

def connectedUsers():
#Esta función se conectará a la base de datos del servidor NAC y realizará una consulta SQL sobre la tabla node, realizando un filtro sobre el campo "status". Si su valor es igual a "reg", significa que se trata de un equipo/usuario conectado, por lo que se muestran los datos asociados al mismo.
    try:
        database = mysql.connector.connect(host="10.0.100.10", user="pf", password="O4KA0ujh3}}QmkMR", database="pf")
        datos = database.cursor()
        datos.execute("SELECT user_agent,computername,mac,status FROM node WHERE status='reg'")
        resultado = datos.fetchall()
        return (resultado)
        database.close
        quit()
    except mysql.connector.Error as error_type:
        return ('Se ha producido el siguiente error al conectar a la base de datos. Por favor, compruebe los datos de conexión'), error_type.msg
        database.close
        quit()

def deniedUsers():
#Esta función es exactamente igual a la anterior. Simplemente se modifica el filtro para obtener el valor "unreg", que hace referencia a equipos/usuarios denegados.
    try:
        database = mysql.connector.connect(host="10.0.100.10", user="pf", password="O4KA0ujh3}}QmkMR", database="pf")
        datos = database.cursor()
        datos.execute("SELECT user_agent,computername,mac,status FROM node WHERE status='unreg'")
        resultado = datos.fetchall()
        return (resultado)
        database.close
        quit()
    except mysql.connector.Error as error_type:
        return ('Se ha producido el siguiente error al conectar a la base de datos. Por favor, compruebe los datos de conexión'), error_type.msg
        database.close
        quit()
```

10.2. Plantilla HTML – Ficheros *cUsers_result.html* y *dUsers_result.html*

cUsers_result.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Listado de usuarios conectados</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">
  </head>
```

```
  <body>
    <form>
      <h1>Listado de usuarios conectados...</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <p> <strong> Datos de los usuarios conectados:<br><br> (Equipo, MAC, Usuario, Estado)
</strong></p>
        <p>
                    {% for line in connected %}
                    {{line}}<br></p>
                    {% endfor %}
        <p></p><br>
        </div>
      </div>
      <center><a href="/monitor">Volver | </a><a href="/"> Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

dUsers_result.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Listado de usuarios denegados</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">
  </head>
  <body>
    <form>
      <h1>Listado de usuarios denegados...</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <p> <strong> Datos de los usuarios denegados:<br><br> (Equipo, MAC, Usuario, Estado)
</strong></p>
        <p>
                    {% for line in denied %}
                    {{line}}<br></p>
                    {% endfor %}
        <p></p><br>
        </div>
      </div>
      <center><a href="/monitor">Volver | </a><a href="/">  Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

Anexo 11: Código y ficheros de la función Ver accesos bloqueados

Estos ficheros son los utilizados al acceder a la ruta *"/monitor/blocked"* de la aplicación. (ver también el código de esta ruta en el fichero *nacApp.py*).

11.1. Script Phyton – Fichero *blocked_access.py*

```
import paramiko
import time
import paramiko
import time

def blockedAccess():
#Se capturan los ficheros log del firewall y se almacenan en la variable output, la cual es
devuelta al fichero nacApp para mostrar su contenido al usuario.
    firewall = paramiko.SSHClient()
    firewall.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    firewall.connect("10.0.100.1", port=22, username="admin", password="pfsense")
    stdin, stdout, stderr = firewall.exec_command('cat /var/log/filter.log* | filterparser.php |
grep -wv ff02 | grep block')
    output = str(stdout.readlines())
    output = output.split("\\n',")
    firewall.close()
    return output
```

11.2. Plantilla HTML – Fichero *blocked_access_result.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Listado de accesos bloqueados</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">
  </head>
  <body>
    <form>
      <h1>Listado de accesos bloqueados...</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <p> <strong> Accesos bloqueados: </strong></p>
        <p>
                    {% for line in blocked_access %}
                    {{line}}<br></p>
                    {% endfor %}
        <p></p><br>
        </div>
      </div>
      <center><a href="/monitor">Volver | </a><a href="/"> Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

Anexo 12: Código y ficheros de la función Captura de paquetes

Estos ficheros son los utilizados al acceder a la ruta *"/tshoot/sniffer"* de la aplicación. (ver también el código de esta ruta en el fichero *nacApp.py*).

12.1. Formulario HTML – Fichero *sniffer.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formulario para agregar un nuevo Switch</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">
  </head>
  <body>
    <form action="{{ url_for('sniffer')}}" method="post">
      <h1>Capturar paquetes de autenticación</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <label for="seconds"><strong>Tiempo de captura de paquetes</strong></label>
        <input type="text" placeholder="Número de segundos..." name="seconds" required>
        </div>
      </div>
      <button type="submit"><b>Inciar captura</b></button>
      <center><a href="/tshoot">Volver | </a><a href="/">  Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

12.2. Script Phyton – Fichero *sniffer.py*

```
import os
import time
import paramiko
import promptlib

def sniffer(seconds, directorio):
"""Lo primero que se hace es establecer la conexión SSH con el servidor NAC, para posteriormente
ejecutar el comando que se encargar de realizar la captura de paquetes, la cual será almacenada en
el fichero denominado como captura_auth.pcap. La captura durará los segundos que haya introducido
el usuario, los cuales son recibidos como parámetro. Una vez realizada, se almacena en el
directorio indicado por el usuario, también es recibido como parámetro."""
    try:
        nac_packetfence = paramiko.SSHClient()
        nac_packetfence.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        nac_packetfence.connect("10.0.100.10", port=22, username="root", password="pass")
        nac_packetfence.exec_command("timeout "+str(seconds)+" tcpdump -i any port 1812 -w
                                     captura_auth.pcap")
        time.sleep(seconds)
        file="captura_auth.pcap"
        directorio = directorio + "/captura_auth.pcap"
        sftp_for_download = nac_packetfence.open_sftp()
        sftp_for_download.get('/root/' + file, directorio)
        sftp_for_download.close()
        nac_packetfence.close()
        return str(directorio)
    except:
        return str("\nSe ha producido un error de permisos al intentar copiar el fichero. "
              "Por favor, compruebe los permisos de la carpeta seleccionada y vuelva a
intentarlo.")
```

Anexo 13: Código y ficheros de la función Descarga de logs

Estos ficheros son los utilizados al acceder a la ruta *"/tshoot/logs"* de la aplicación. (ver también el código de esta ruta en el fichero *nacApp.py*).

13.1. Formulario HTML – Fichero *logs_download.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formulario para agregar un nuevo Switch</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet">
  </head>
  <body>
    <form action="{{ url_for('getlogs')}}" method="post">
      <h1>Descarga de logs</h1>
      <div class="formcontainer">
      <hr/>
      <div class="container">
        <label for="log_select"><strong>Log</strong></label>
        <select name="id_log" id="log">
            <option value="1">Log del servidor Radius (packetfence)</option>
            <option value="2">Log general de packetfence (packetfence)</option>
            <option value="3">Log del servidor Apache (packetfence)</option>
            <option value="4">Log de filtros del servidor NAC (packetfence)</option>
            <option value="5">Log del servidor DHCP (firewall)</option>
            <option value="6">Log de reglas del firewall (firewall)</option>
            <option value="7">Log de accesos administrativos al firewall (firewall)</option>
        </select>
        <input type="submit" value="Descargar">
        </div>
      </div>
      <center><a href="/tshoot">Volver | </a><a href="/">  Cerrar sesión </a></center>
    </form>
  </body>
</html>
```

13.2. Script Phyton – Fichero *logs_download.py*

```
import os
import time
import paramiko
import promptlib

def getLogs(opcion,directorio):
# Se comprueba la opción que ha seleccionado el usuario (recibida por parámetro) y en base a la
misma el script se conectará vía SSH al servidor NAC o al firewall, desde donde descargará el log
en la carpeta indicada por el usuario, también recibida por parámetro.
    file=""
    try:
        if opcion>="1" and opcion<="4":
            nac_packetfence = paramiko.SSHClient()
            nac_packetfence.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            nac_packetfence.connect("10.0.100.10", port=22, username="root", password="pass")
            sftp_for_download=nac_packetfence.open_sftp()
            if opcion =="1":
                file="radius.log"
                directorio=directorio+"/nac_radius.log"
            if opcion =="2":
                file="packetfence.log"
                directorio=directorio+"/nac_packetfence.log"
            if opcion =="3":
```

```
            file="httpd.apache"
            directorio=directorio+"/nac_httpd.apache.log"
        if opcion == "4":
            file = "pffilter.log"
            directorio = directorio + "/nac_pffilter.log"

        sftp_for_download.get('/usr/local/pf/logs/'+file,directorio)
        sftp_for_download.close()
        nac_packetfence.close()
        return str(directorio)

    if opcion >="5" and opcion<="7":
        firewall = paramiko.SSHClient()
        firewall.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        firewall.connect("10.0.100.1", port=22, username="admin", password="pfsense")
        sftp_for_download=firewall.open_sftp()
        if opcion =="5":
            file="dhcpd.log"
            directorio=directorio+"/firewall_dhcpd.log"
        if opcion =="6":
            file="filter.log"
            directorio=directorio+"/firewall_filter.log"
        if opcion =="7":
            file="auth.log"
            directorio=directorio+"/firewall_auth.log"

        sftp_for_download.get('/var/log/'+file,directorio)
        sftp_for_download.close()
        firewall.close()
        return str(directorio)
except:
    return str("\nSe ha producido un error de permisos al intentar copiar el fichero. "
          "Por favor, compruebe los permisos de la carpeta seleccionada y vuelva a
          intentarlo.")
    sftp_for_download.close()
    nac_packetfence.close()
    quit()
```

Anexo 14: Pruebas con la aplicación de gestión centralizada

15.1. Página de validación de usuario y acceso a rutas sin haber iniciado sesión.

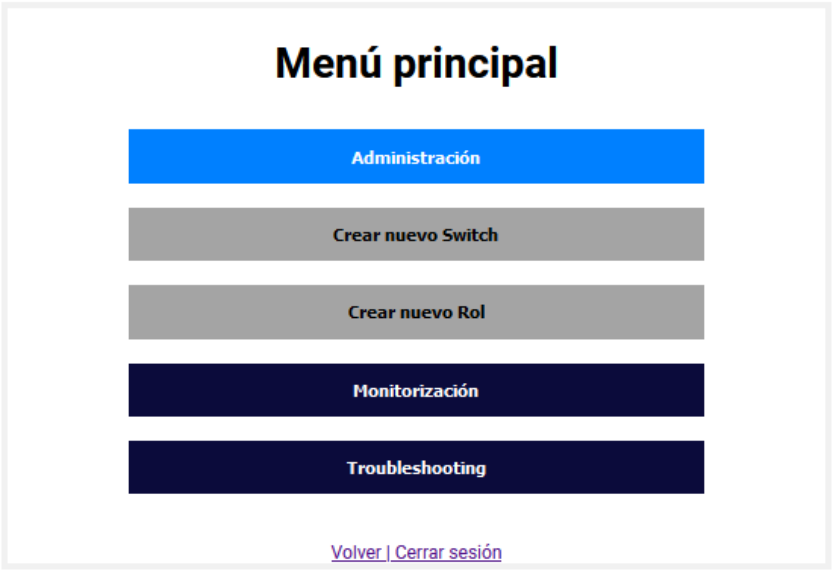
Al acceder a la aplicación se cargará el contenido de la ruta “/”, la cual mostrará el siguiente formulario de inicio de sesión…

Si, sin haber iniciado sesión, intentamos acceder a cualquier otra ruta de la aplicación, como por ejemplo a *“/main*”, automáticamente se redirige al usuario nuevamente a “/”, obligándole a iniciar sesión.

15.2. Validación correcta y navegación por menús

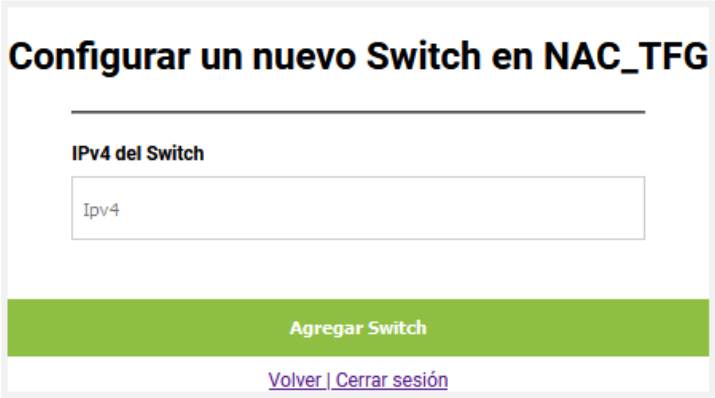
Si el usuario inicia sesión correctamente (*admin/admin*) se le redirigirá a la ruta “*/main*”, la cual contiene el menú principal, siendo el siguiente:

Mientras que, si navega por cualquiera de las opciones, se mostrarán las utilidades asociadas a cada una de ellas. Por ejemplo, para Administración…

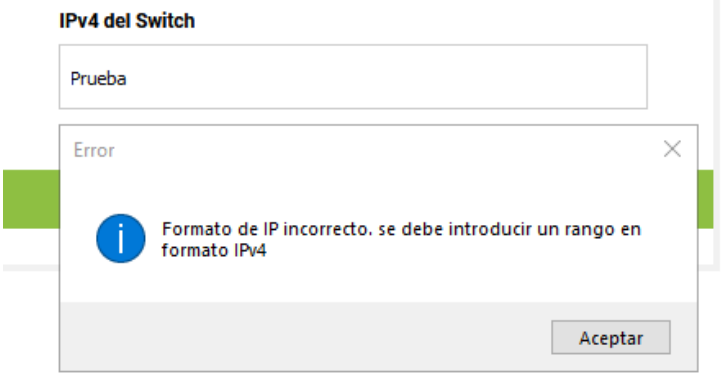


15.3. Crear un nuevo Switch

Al acceder a esta función, se mostrará un formulario solicitando la IP del Switch que se desea configurar y agregar al entorno NAC…



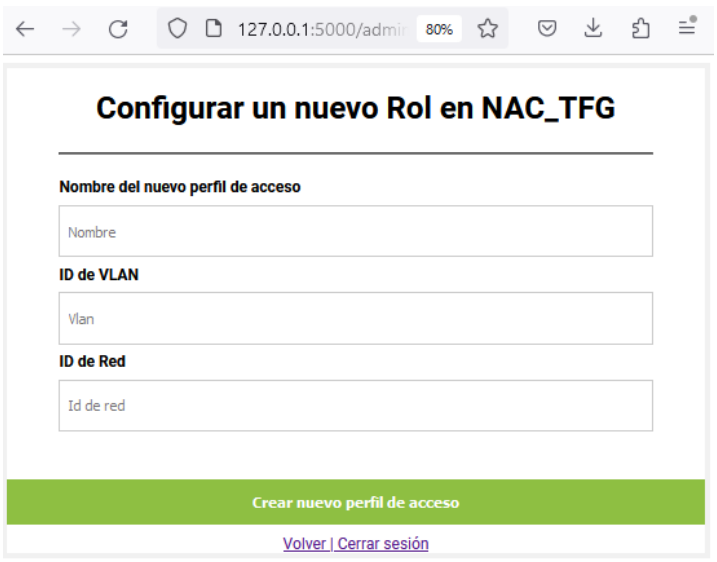
Si el dato introducido no corresponde al formato de una IPv4 mostrará el siguiente error y, tras aceptarlo, volverá a mostrar el formulario...

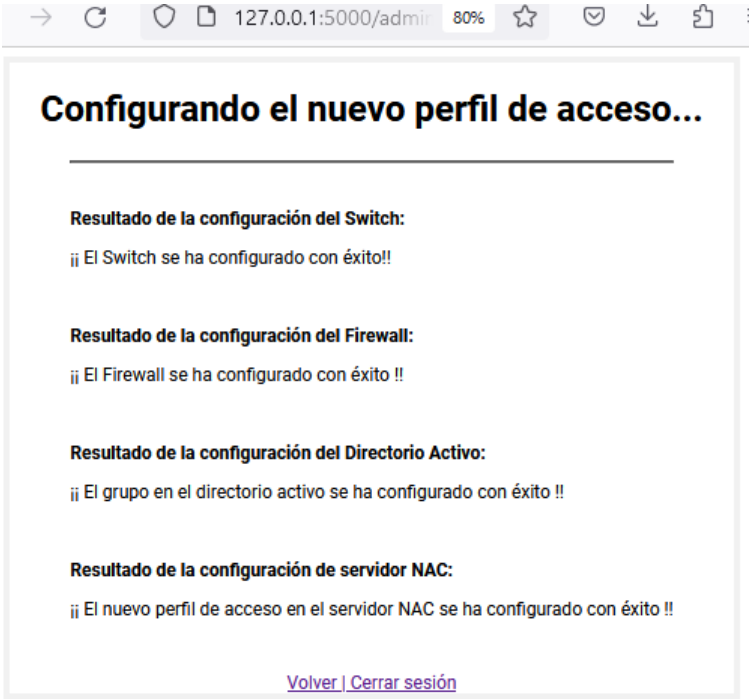


Si se introduce una IP correcta se procede a configurar el Switch, y tras finalizar mostrará el resultado de la operación.

15.4. Crear un nuevo Rol

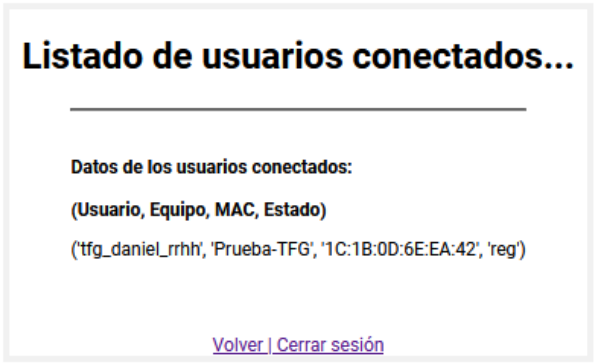
Al acceder a esta función, se mostrará un formulario que solicitará varios datos, los cuales serán verificados antes de realizar cualquier acción. Si alguno de ellos no tiene el formato correcto, se muestra un mensaje de error, mientras que de lo contrario, se realizaran las acciones oportunas y se mostrará el resultado de las mismas…

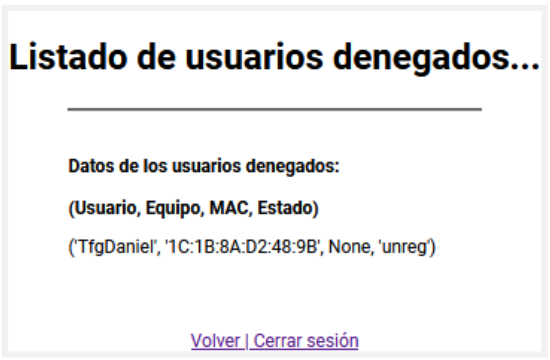




15.5. Ver usuarios conectados y Ver usuarios denegados

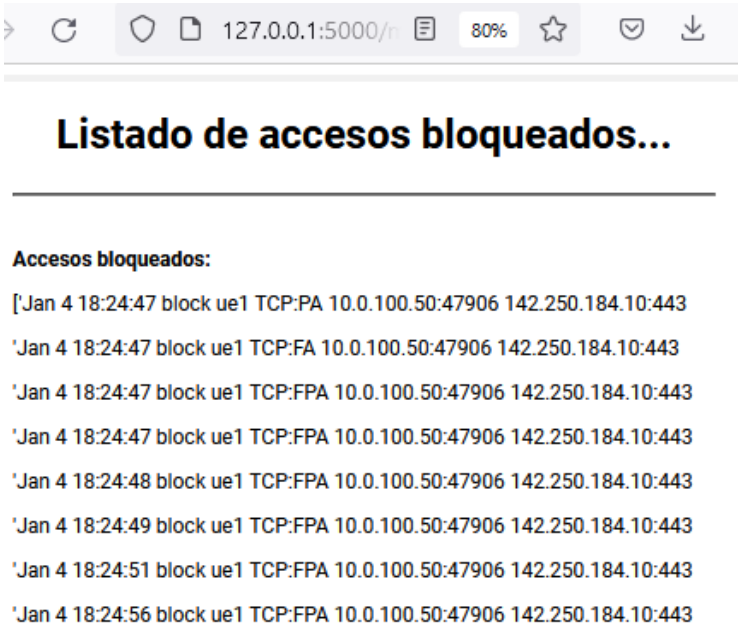
Al acceder a estas funciones se mostrará el listado de usuarios conectados y denegados por NAC. Ambos métodos son exactamente iguales, mostrando los siguientes resultados…





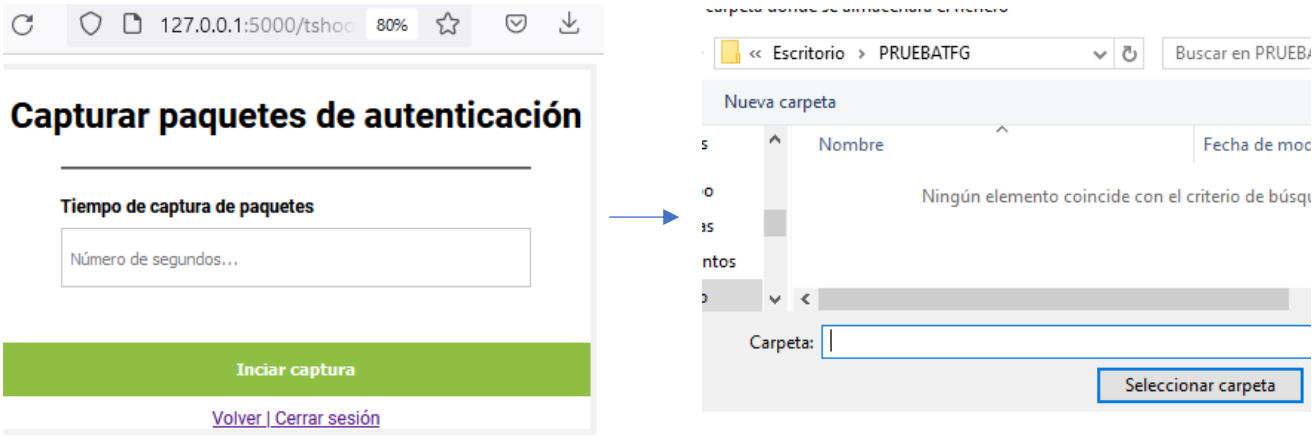
15.6. Ver accesos bloqueados

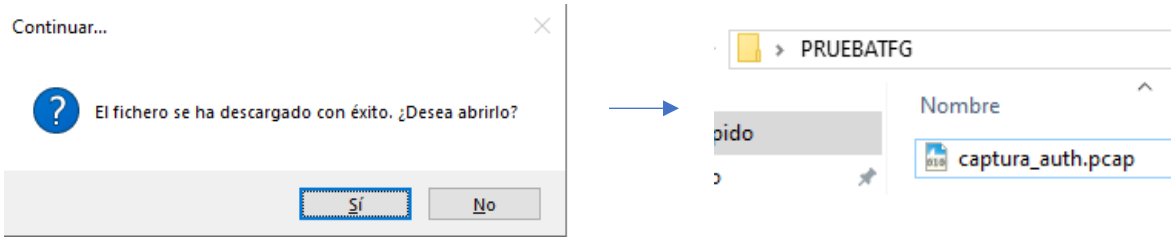
En este caso, se mostrará un listado de los accesos IPv4 bloqueados en el firewall… El resultado de su ejecución ha sido el siguiente…



15.7. Capturar paquetes de autenticación

Al acceder a esta función se mostrará un formulario solicitando el tiempo que durará la captura de paquetes, el cual se ha establecido en un máximo de 10 segundos. Tras ello, comienza la captura en el servidor NAC y posteriormente se descarga el fichero en la carpeta seleccionada por el usuario.





15.8. Descarga de logs

Por último, al acceder a esta función se mostrará una lista despegable con diferentes opciones de logs a descargar. Cuando el usuario seleccione una y pulse sobre descargar, se mostrará una ventana despegable solicitando la ubicación donde descargar el fichero. Posteriormente, el fichero se descarga y se da la posibilidad al usuario de abrirlo….