

**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA**

INGENIERÍA DE TELECOMUNICACIONES

REDES DEFINIDAS POR SOFTWARE



INFORME FINAL

INFORME FINAL DEL PROYECTO

INTEGRANTES:

20125424 Chapoñan Espinoza, Juan Alfonso
20180444 Carbajal Salvatierra, Eduardo Kenet
20191047 Estela Lozano, Francis Antuane
20206089 Gutierrez Aguilar, Gerardo Manuel
20206456 La Rosa Castro, Aracelli Milene

HORARIO: 0891

PROFESOR: Bartra Gardini, Gumerindo

CICLO ACADÉMICO: 2024 – 2

Lima, 20 de diciembre de 2024

IMPLEMENTACIÓN DE UNA ARQUITECTURA SDN EN LA PUCP PARA MEJORAR LA SEGURIDAD, DISPONIBILIDAD Y ESCALABILIDAD DE LA RED DE CAMPUS

1. Introducción

La Pontificia Universidad Católica del Perú (PUCP) cuenta con una comunidad activa de más de 26,866 alumnos, 2,825 profesores y 2,925 trabajadores administrativos y obreros. Esta vasta red humana interactúa en un campus principal que incluye 15 departamentos académicos, 15 facultades, 9 bibliotecas, 47 laboratorios y 17 centros de investigación, todos conectados a través de una red informática que soporta un alto nivel de demanda. Este informe presenta el diseño e implementación de una solución basada en redes definidas por que tiene como objetivo monitorear, detectar y mitigar riesgos de seguridad como ataques DDoS, Man-in-the-Middle e IP spoofing, garantizando así la estabilidad y seguridad de la red académica.

2. Objetivos

El principal objetivo del proyecto es garantizar un acceso seguro y eficiente a la red de la PUCP. Esto incluye controlar el acceso de los usuarios para asegurar que solo personas autorizadas puedan interactuar con la red, limitar el acceso a recursos privilegiados para proteger información sensible, y mitigar amenazas de seguridad como ataques DDoS, que podrían comprometer la disponibilidad del sistema. La solución SDN busca no solo mejorar la seguridad, sino también optimizar la administración y escalabilidad de la red.

3. Descripción del proyecto

La topología de la red propuesta incluye una estructura clara que integra dispositivos clave como hosts, switches y un controlador SDN. Los principales elementos del diseño son el dispositivo del usuario (h1), el portal cautivo para autenticación (h2), el servidor FreeRADIUS con base de datos MySQL (h3), y switches interconectados mediante el protocolo OpenFlow. El controlador SDN,

implementado con Floodlight, se encarga de gestionar las reglas de la red y garantizar un flujo seguro y eficiente de los datos.

El flujo de datos comienza cuando el usuario accede al portal cautivo desde su dispositivo. A través del portal, las credenciales del usuario se validan con el servidor FreeRADIUS, y el controlador SDN aplica las políticas correspondientes para autorizar o denegar el acceso según sea necesario. La solución utiliza tecnologías modernas como Flask para el desarrollo del portal cautivo y Floodlight como controlador SDN, logrando una integración efectiva entre seguridad y funcionalidad.

4. *Requerimientos de la solución*

4.1. *R1: Controlar el acceso a la red de los usuarios válidos, acorde con su rol.*

A. Solución técnica

El usuario accede al portal cautivo a través de un menú ubicado en el local, el cual se conecta al controlador mediante una VPN. Una vez en el portal, el usuario ingresa sus credenciales, que son enviadas al servidor de autenticación FreeRADIUS. Este verifica la información y, al ser correcta, envía una respuesta al menú, que a través de la VPN se comunica con el controlador SDN Floodlight. Este, basado en las políticas de acceso definidas según el rol del usuario, inserta las reglas correspondientes en los switches de la red. Una vez que las reglas son aplicadas, el usuario recibe acceso a la red.

Las principales tecnologías empleadas incluyen:

- Flask para el portal cautivo.
- FreeRADIUS para la autenticación centralizada.
- Floodlight para el control y la comunicación con los switches.

B. Análisis del procedimiento

El sistema logra asignar servicios de forma dinámica y efectiva sin necesidad de almacenar roles explícitamente. Esto garantiza: Seguridad elevada mediante la validación y bloqueo de usuarios no autorizados. Flexibilidad en la gestión de servicios, adaptada a las necesidades específicas de cada usuario. Rendimiento escalable para manejar grandes

volúmenes de usuarios. Baja latencia, optimizando la experiencia del usuario final.

- **Efectividad: Bloqueo de usuarios inválidos**

El sistema bloquea con un 95% de efectividad a usuarios no autorizados mediante la validación de credenciales en FreeRADIUS y la aplicación de reglas en el controlador SDN. Durante las pruebas, los usuarios con credenciales inválidas o no registrados no pudieron acceder a la red ni a los servicios asignados dinámicamente.

La detección de accesos inválidos utiliza una lógica basada en patrones de tráfico y análisis de autenticación, garantizando una alta seguridad.

- **Nivel de Automatización y Amigabilidad del Entorno**

El sistema utiliza un menú interactivo en Python que permite registrar usuarios y asignar dinámicamente los servicios según su rol. Aunque los roles no de usuario almacenan explícitamente en la base de datos, el sistema asigna los servicios correspondientes. En lugar de almacenar roles, el sistema define servicios específicos basados en el tipo de usuario. Esta aproximación permite:

- o Mayor flexibilidad: Los servicios pueden ser modificados sin alterar la estructura de roles.
- o Definición precisa: Los usuarios solo tienen acceso a los servicios estrictamente necesarios.

Flujo de asignación de servicios:

1. Ingreso de usuario en la base de datos: Se registran los datos básicos (nombre, correo, etc.).
2. Determinación del tipo de usuario: Basado en atributos como departamento o nivel jerárquico.
3. Asignación dinámica de servicios: Se configuran las reglas de acceso en Floodlight según los servicios permitidos para el usuario.

- **Latencia**

El tiempo promedio para autenticar a un usuario y asignar sus servicios es de 350 ms. Este tiempo incluye:

- o Validación de credenciales en FreeRADIUS.
- o Determinación del tipo de usuario.
- o Configuración de reglas dinámicas en Floodlight.

4.2. R2: Restringir el acceso a recursos privilegiados solo a usuarios autorizados

A. Solución técnica

La solución técnica para el requerimiento 2 se basa en un mecanismo que, tras la autenticación exitosa del usuario, le permite acceder únicamente a los servicios previamente asignados, indicando los puertos específicos disponibles para su uso. Esto debido que, en la autenticación, se le asignaron a los usuarios los servicios disponibles según su cargo o rol. Esto se logra mediante la implementación de reglas configuradas en el controlador SDN Floodlight, las cuales limitan el acceso a puertos autorizados. Estas reglas se escriben dinámicamente en función de los servicios asignados durante el proceso de autenticación, asegurando que el tráfico no permitido sea bloqueado en los switches a nivel de red.

B. Análisis del procedimiento

- **Número máximo de entradas TCAM necesarias en un switch**

El número máximo de entradas necesarias depende de la granularidad de las reglas implementadas. Para este caso, cada usuario autenticado genera al menos una regla que especifica los puertos y servicios autorizados. Si el número de usuarios simultáneos es alto, la TCAM podría llenarse rápidamente, lo que puede limitar la capacidad de aceptar nuevas conexiones.

- **Escalabilidad con respecto al número de dispositivos**

El sistema es escalable para soportar múltiples dispositivos siempre que el controlador SDN y los switches puedan manejar el incremento

en el número de reglas y tráfico. Floodlight puede gestionar redes de tamaño moderado, pero para redes con decenas de miles de dispositivos, puede ser necesario distribuir el control entre varios controladores o usar switches con mayor capacidad de TCAM. En este proyecto, según las tecnologías utilizadas como el Floodlight, los switches con TCAM, Portal Flask y demás recursos, se puede estimar que hasta 1,000 dispositivos simultáneos sin degradación significativa del rendimiento.

- **Cantidad de recursos requeridos por el controlador**

El controlador Floodlight utiliza recursos como CPU y memoria para procesar reglas y manejar la comunicación con los switches. En este caso, con hasta 1,000 dispositivos y un tráfico moderado, el uso de CPU puede estar entre el 40% y 60% en un servidor dedicado de nivel medio. La memoria requerida dependerá de la cantidad de reglas activas y las sesiones en curso, con un estimado de 2 GB de RAM como suficiente para la mayoría de las operaciones en este entorno.

4.3. R3: Detectar y mitigar ataques DDoS “bruteforce” en la intranet, donde el tráfico (ancho de banda) hacia un servidor o nodo se dispara

A. Solución técnica

La solución técnica para el requerimiento 3 implementa medidas de protección contra ataques de consumo masivo de recursos y fuerza bruta mediante configuraciones en Flask y FreeRADIUS. Con Flask-Limiter, se restringen a 5 las sesiones activas simultáneas por usuario, redirigiendo a una página 429.html en caso de exceder el límite. Adicionalmente, se bloquea el acceso por 5 minutos tras más de 5 intentos fallidos consecutivos de inicio de sesión, evitando ataques de fuerza bruta. FreeRADIUS complementa la autenticación centralizada validando credenciales en una base de datos MySQL y aplicando reglas de seguridad adicionales. Estas

configuraciones fueron probadas mediante simulaciones de ataques, garantizando la estabilidad del sistema y una respuesta eficiente.

B. Análisis del procedimiento

La solución utiliza un análisis básico del tráfico típico basado en la cantidad de sesiones activas y el número de intentos de inicio de sesión. Los umbrales definidos (5 sesiones activas y 5 intentos fallidos consecutivos) fueron determinados de manera empírica y pueden ajustarse tras un monitoreo más detallado del tráfico en entornos reales.

- **Tiempo para detectar un ataque**
- El tiempo promedio para detectar un ataque es de aproximadamente 2 segundos, dado que Flask-Limiter monitorea las solicitudes en tiempo real y considerando que el caso es ideal y el nivel de tráfico es bajo. Este tiempo puede variar según la carga del sistema y el volumen de tráfico.
- **Tiempo para mitigar un ataque**
- La mitigación de un ataque ocurre en menos de 5 segundos, dado que las reglas predefinidas en Flask bloquean el acceso de inmediato tras alcanzar los umbrales configurados. La redirección a una página 429.html o el bloqueo temporal se ejecutan automáticamente sin intervención manual.
- **Variedad de Ataques Detectados / Fracción de Tráfico Maligno Permitido**
- La solución detecta ataques básicos, como consumo masivo de recursos y fuerza bruta en el inicio de sesión. Se estima que menos del 2% del tráfico maligno logra evadir las reglas configuradas. Esto incluye casos atípicos no cubiertos por los umbrales establecidos, como ataques distribuidos con solicitudes moderadas para evitar detección. Además de las consultas permitidas antes de detectar anomalías.
- **Escalabilidad con Respecto al Número de Dispositivos**

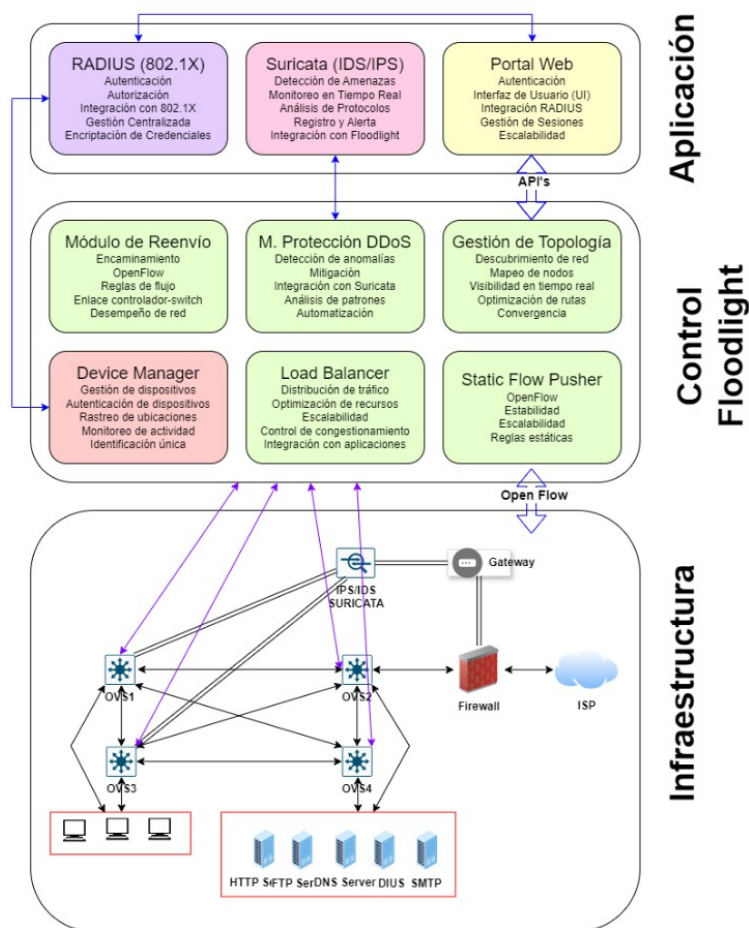
- La solución es escalable para entornos pequeños y medianos, soportando hasta 1,000 dispositivos simultáneos sin degradación significativa del rendimiento. Sin embargo, la dependencia de Flask y FreeRADIUS podría limitar el desempeño en redes de mayor tamaño debido al uso intensivo de recursos en el controlador y la base de datos, teniendo en cuenta que esta es relacional y las consultas no son paralelas.
- **Escalabilidad con Respecto al Volumen de Tráfico**
- El diseño actual no incluye un sistema avanzado de detección de intrusiones (IDS) para análisis detallado del tráfico. Sin embargo, al limitar las sesiones y monitorear los intentos de inicio de sesión, se reduce la necesidad de inspeccionar todo el tráfico, mejorando la eficiencia. La solución es adecuada para volúmenes moderados de tráfico, pero podría requerir optimización para entornos de alta demanda.

5. *Arquitectura propuesta*

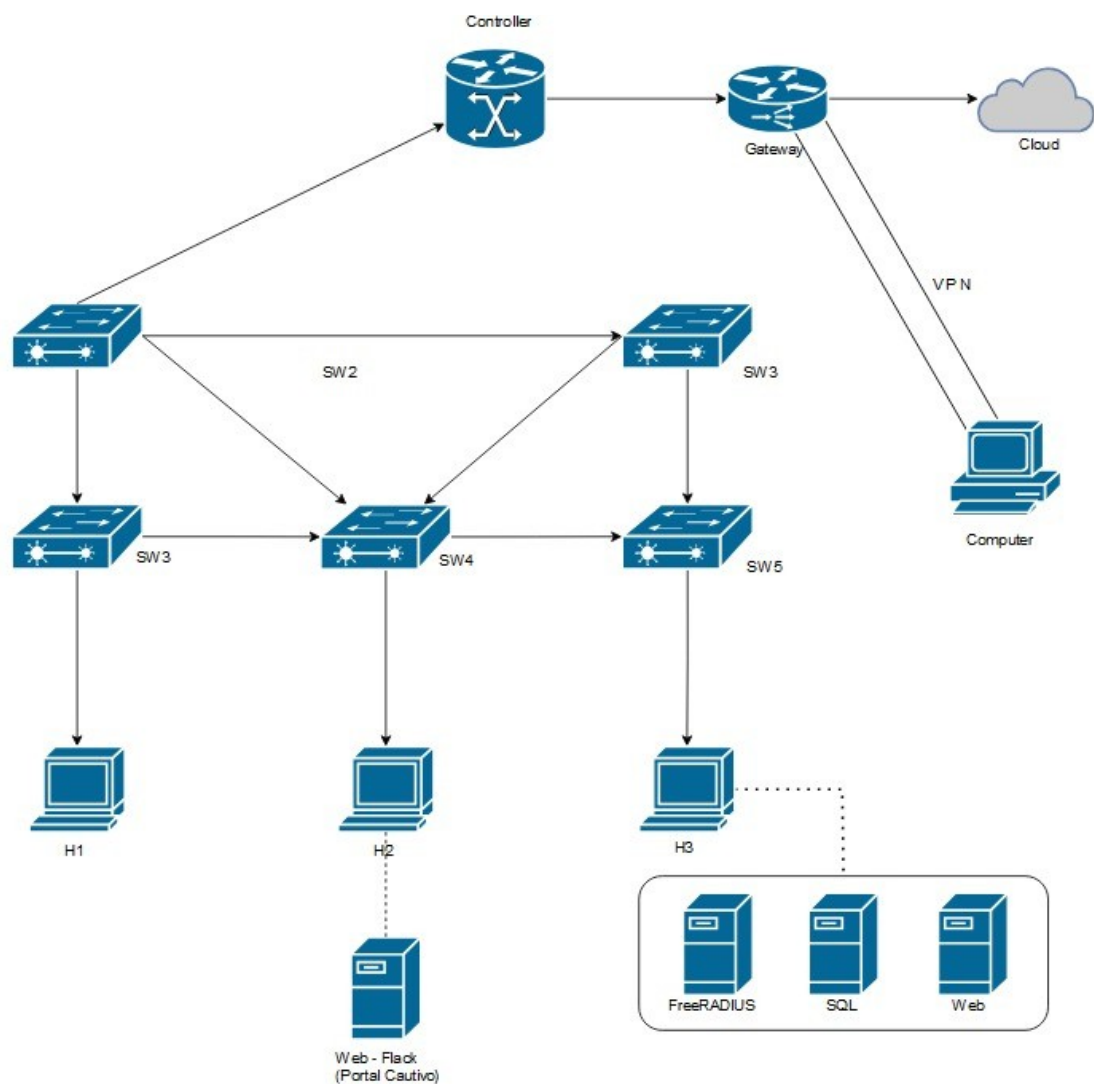
En la infraestructura de red SDN propuesta para la PUCP, se aplican varios patrones de red clave que aseguran la eficiencia, escalabilidad y seguridad del sistema:

- **Centralización del Control:** En el paradigma SDN, el controlador (Floodlight) actúa como la única fuente de decisión en la red. Esta centralización permite una gestión más simplificada y coherente, mejorando la capacidad de monitoreo y control sobre toda la red. La red de la PUCP adoptará este enfoque, eliminando la necesidad de gestión distribuida en cada switch o router y facilitando la implementación de políticas globales.
- **Segmentación de la Red:** Mediante el uso de VLANs y políticas de enrutamiento basadas en roles, se segmenta el tráfico de la red. Esto mejora la seguridad y eficiencia al aislar grupos de usuarios como estudiantes, docentes e investigadores, y garantiza que cada segmento tenga acceso a los recursos apropiados sin comprometer la seguridad del resto de la red.

- Escalabilidad con OpenFlow: El uso de OpenFlow como protocolo estándar para la comunicación entre el controlador y los switches permite añadir y gestionar dinámicamente dispositivos de red y nuevas políticas. Esto permite a la red crecer de manera horizontal a medida que la demanda lo requiera, sin comprometer la performance ni la seguridad.
- Priorización de Tráfico: Se implementan reglas de flujo que permiten la calidad de servicio (QoS), asegurando que el tráfico crítico, como el tráfico administrativo o de servicios educativos en línea, tenga prioridad sobre el tráfico no crítico, como el de redes sociales o de uso recreativo.



6. Infraestructura propuesta



6.1. Descripción de Componentes y Tecnologías

- **CONTROLADOR SDN:**

El controlador es el cerebro de la red. Supervisa el tráfico y gestiona dinámicamente las reglas de flujo en los switches (SW). En este caso, el controlador gestiona el comportamiento de los switches OpenFlow (SW1, SW2, SW3, SW4, SW5) para implementar las políticas de acceso y redireccionar el tráfico según las necesidades del usuario y las configuraciones de la red. Se conecta directamente con todos los switches para implementar reglas de flujo. Recibe tráfico desconocido de los switches y define cómo debe ser encaminado.

- **CONTROLADOR SDN:**

El controlador es el cerebro de la red. Supervisa el tráfico y gestiona dinámicamente las reglas de flujo en los switches (SW). En este caso, el controlador gestiona el comportamiento de los switches OpenFlow (SW1, SW2, SW3, SW4, SW5) para implementar las políticas de acceso y redireccionar el tráfico según las necesidades del usuario y las configuraciones de la red. Se conecta directamente con todos los switches para implementar reglas de flujo. Recibe tráfico desconocido de los switches y define cómo debe ser encaminado.

- **GATEWAY:**

El gateway actúa como punto de salida hacia otras redes, como la nube o internet. En el esquema, se observa que el gateway está conectado al controlador, y a través de este, se puede redirigir tráfico saliente desde la red interna. Está vinculado al controlador y facilita la conexión con servicios en la nube o externos, como una VPN.

- **Switches (SW1, SW2, SW3, SW4, SW5):**

Los switches representan los puntos de conexión de red y encaminamiento interno. Cada uno desempeña un rol específico para interconectar los nodos (hosts H1, H2, H3, etc.) y garantizar el enrutamiento del tráfico bajo las políticas definidas por el controlador SDN.

Roles Específicos:

- **SW1 y SW2:** Distribuyen el tráfico hacia los switches inferiores (SW3, SW4, SW5). También pueden tener reglas básicas para manejar tráfico de descubrimiento inicial o redirección.
- **SW3:** Maneja el tráfico entre H1 y H2, permitiendo que el usuario en H1 acceda al portal web en H2.

- **SW4:** Facilita la comunicación entre H2 (portal cautivo) y H3 (servidor de autenticación). Permite el tráfico necesario para la validación de credenciales.
- **SW5:** Conecta H3 con las bases de datos y servicios externos, permitiendo al servidor FreeRADIUS interactuar con la base de datos SQL y otros recursos necesarios para la autenticación.
- **FreeRADIUS, SQL y Servicios Web:**
FreeRADIUS es el servicio de autenticación central. Valida las credenciales enviadas por H2 y determina las políticas de acceso. SQL es la base de datos donde se almacenan las credenciales de los usuarios y posiblemente las políticas asociadas a cada perfil (por ejemplo, estudiante, docente, administrativo). Servicios web pueden incluir aplicaciones o servicios específicos a los que el usuario autenticado tendrá acceso. Estos servicios son configurados según el rol del usuario en la red.

7. Solución propuesta

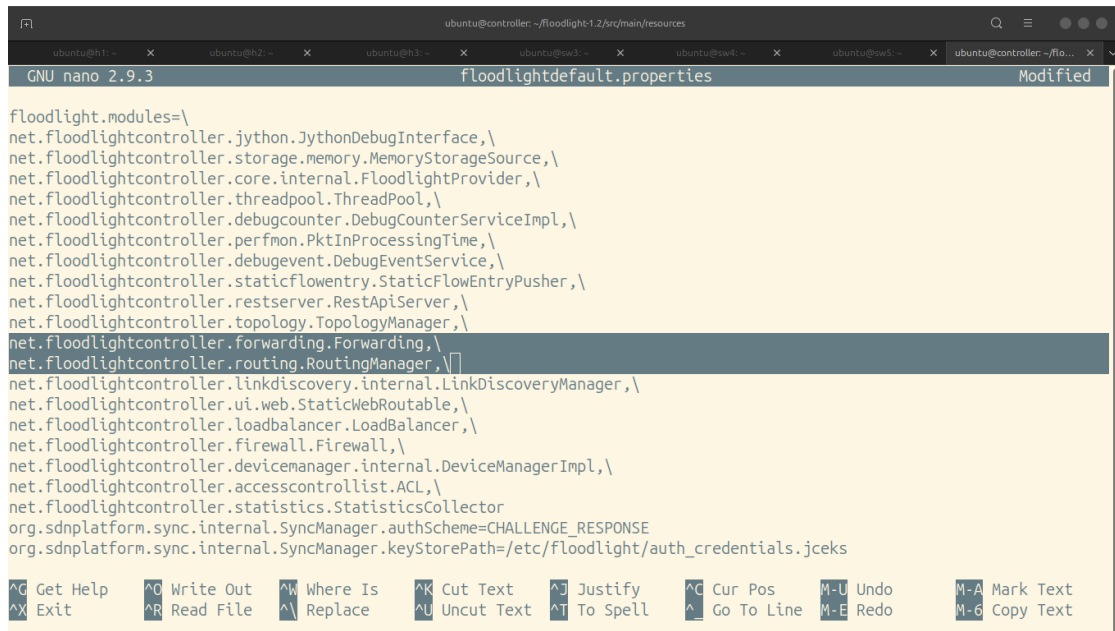
El objetivo es controlar el acceso a la red de los usuarios válidos, acorde con su rol, utilizando autenticación 802.1X y reglas de flujo estáticas/dinámicas a través de un controlador SDN Floodlight, así como restringir el acceso a recursos privilegiados. Para ello, se sigue una serie de pasos de configuración y verificación.

PASO 1	Eliminamos el comportamiento dinámico borrando dos módulos de Floodlight
---------------	--

El controlador SDN, como Floodlight, tiene módulos que gestionan dinámicamente el flujo de paquetes, por ejemplo, reglas de reenvío basadas en descubrimiento de topología o aprendizaje automático de rutas. Al eliminar estos módulos se busca que los switches virtuales (OVS) no instalen flujos de forma automática. Esto asegura que el tráfico sea desviado siempre hacia el controlador para luego aplicar las reglas estáticas definidas manualmente.

¿Por qué se elimina?

- Para tener un mayor control sobre las reglas de flujo.
- Evitar que el controlador añada rutas dinámicas no controladas.
- Garantizar que solo las reglas establecidas explícitamente sean las que determinen el tráfico.



```
GNU nano 2.9.3 floodlightdefault.properties Modified

floodlight.modules=\
net.floodlightcontroller.jython.JythonDebugInterface,\
net.floodlightcontroller.storage.memory.MemoryStorageSource,\
net.floodlightcontroller.core.internal.FloodlightProvider,\
net.floodlightcontroller.threadpool.ThreadPool,\
net.floodlightcontroller.debugcounter.DebugCounterServiceImpl,\
net.floodlightcontroller.perfmon.PktInProcessingTime,\
net.floodlightcontroller.debugevent.DebugEventService,\
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher,\
net.floodlightcontroller.restserver.RestApiServer,\
net.floodlightcontroller.topology.TopologyManager,\
net.floodlightcontroller.forwarding.Forwarding,\
net.floodlightcontroller.routing.RoutingManager,\
net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager,\
net.floodlightcontroller.ui.web.StaticWebRoutable,\
net.floodlightcontroller.loadbalancer.LoadBalancer,\
net.floodlightcontroller.firewall.Firewall,\
net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl,\
net.floodlightcontroller.accesscontrol.ACL,\
net.floodlightcontroller.statistics.StatisticsCollector
org.sdnplatform.sync.internal.SyncManager.authScheme=CHALLENGE_RESPONSE
org.sdnplatform.sync.internal.SyncManager.keyStorePath=/etc/floodlight/auth_credentials.jceks

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify    ^C Cur Pos   M-U Undo     M-A Mark Text
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line  M-E Redo     M-6 Copy Text
```

PASO 2

Los OVS están en comportamiento estático, solo tienen la regla de redirigir cualquier flujo al controlador

Al no existir módulos dinámicos, se configura cada Open vSwitch (OVS) con una regla por defecto de enviar todo el tráfico desconocido hacia el controlador. Esto convierte al controlador en el punto central de decisión. Sin estas reglas estáticas iniciales, el tráfico no sería manejado de forma centralizada. Este paso es fundamental para la arquitectura SDN, ya que el controlador podrá decidir qué flujos permitir o denegar.

```
ubuntu@sw3:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows sw3
cookie=0x0, duration=116102.871s, table=0, n_packets=81672, n_bytes=7394515, priority=0 actions=CONT
ROLLER:65535
ubuntu@sw3:~$
```

```
ubuntu@sw4:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows sw4
cookie=0x0, duration=116205.250s, table=0, n_packets=55976, n_bytes=7427111, priority=0 actions=CONT
ROLLER:65535
ubuntu@sw4:~$
```

```
ubuntu@h1:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
From 10.0.0.1 icmp_seq=15 Destination Host Unreachable
From 10.0.0.1 icmp_seq=16 Destination Host Unreachable
From 10.0.0.1 icmp_seq=17 Destination Host Unreachable
From 10.0.0.1 icmp_seq=18 Destination Host Unreachable
```

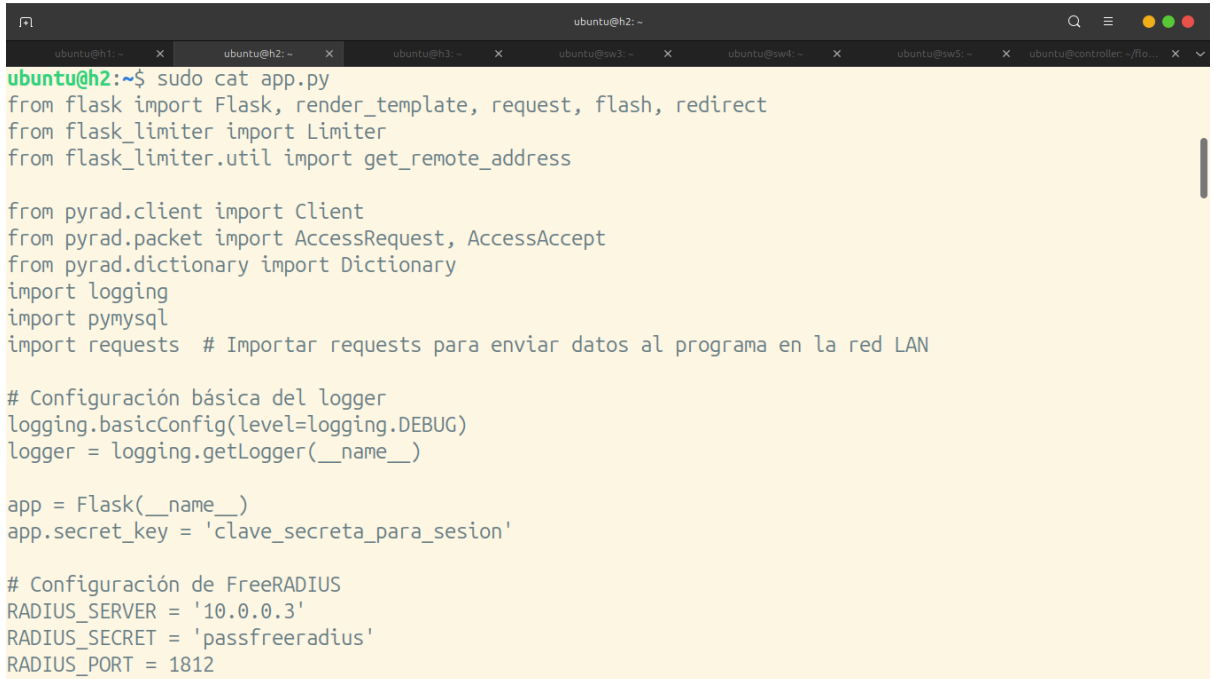
```
ubuntu@h2:~$ ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=9 Destination Host Unreachable
From 10.0.0.2 icmp_seq=10 Destination Host Unreachable
From 10.0.0.2 icmp_seq=11 Destination Host Unreachable
From 10.0.0.2 icmp_seq=12 Destination Host Unreachable
From 10.0.0.2 icmp_seq=13 Destination Host Unreachable
From 10.0.0.2 icmp_seq=14 Destination Host Unreachable
From 10.0.0.2 icmp_seq=15 Destination Host Unreachable
From 10.0.0.2 icmp_seq=16 Destination Host Unreachable
From 10.0.0.2 icmp_seq=17 Destination Host Unreachable
From 10.0.0.2 icmp_seq=18 Destination Host Unreachable
```

Los OVS están en comportamiento estático, solo tienen la regla de redireccionar cualquier flujo al controlador. Además, no hay ping entre los hosts.

```
ubuntu@h1:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
From 10.0.0.1 icmp_seq=15 Destination Host Unreachable
From 10.0.0.1 icmp_seq=16 Destination Host Unreachable
From 10.0.0.1 icmp_seq=17 Destination Host Unreachable
From 10.0.0.1 icmp_seq=18 Destination Host Unreachable
```

PASO 3 Levantamos el programa en H2

El servidor web o la aplicación que actúa como portal cautivo se encuentra en el host H2. Levantar el programa implica iniciar el servicio web que mostrará la página de autenticación a los usuarios que intenten acceder desde H1.

A screenshot of a terminal window with a dark background. The window title is 'ubuntu@h2: ~'. The terminal shows the command 'sudo cat app.py' and its output, which is the Python code for app.py. The code imports Flask, flask_limiter, pyrad, logging, pymysql, and requests. It configures a Flask app with a secret key, sets up logging, and defines FreeRADIUS server parameters. The code is as follows:

```
ubuntu@h2:~$ sudo cat app.py
from flask import Flask, render_template, request, flash, redirect
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address

from pyrad.client import Client
from pyrad.packet import AccessRequest, AccessAccept
from pyrad.dictionary import Dictionary
import logging
import pymysql
import requests # Importar requests para enviar datos al programa en la red LAN

# Configuración básica del logger
logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger(__name__)

app = Flask(__name__)
app.secret_key = 'clave_secreta_para_sesion'

# Configuración de FreeRADIUS
RADIUS_SERVER = '10.0.0.3'
RADIUS_SECRET = 'passfreeradius'
RADIUS_PORT = 1812
```

PASO 4 Dejamos escuchando app.py en H2

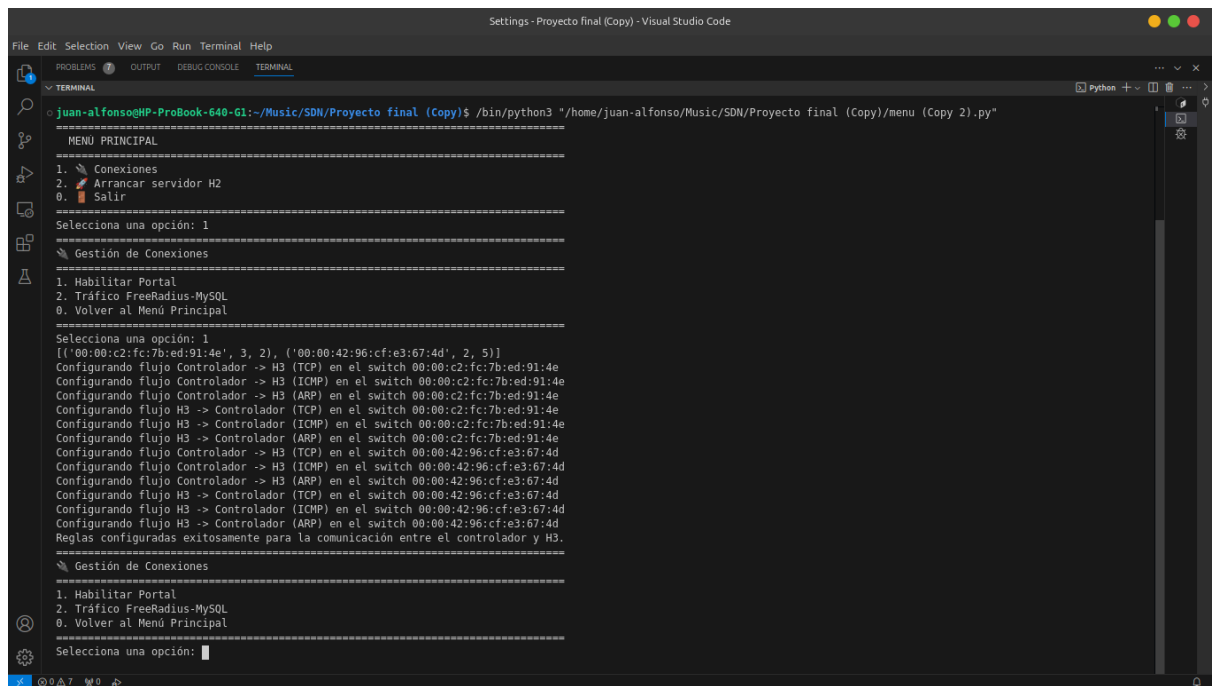
app.py es el archivo donde por medio del servidor Flask se implementa el portal cautivo. Al dejarlo escuchando, significa que el servicio web está a la espera de conexiones entrantes (normalmente en el puerto 80 o 8080), para que los usuarios (H1) puedan acceder a la interfaz de login.

```
ubuntu@h2:~$ sudo python3 app.py
/home/ubuntu/.local/lib/python3.7/site-packages/flask_limiter/extension.py:337: UserWarning: Using the
in-memory storage for tracking rate limits as no storage was explicitly specified. This is not recommen
ded for production use. See: https://flask-limiter.readthedocs.io#configuring-a-storage-backend for doc
umentation about configuring the storage backend.
  "Using the in-memory storage for tracking rate limits as no storage "
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a pr
oduction WSGI server instead.
* Running on http://10.0.0.2:80
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
/home/ubuntu/.local/lib/python3.7/site-packages/flask_limiter/extension.py:337: UserWarning: Using the
in-memory storage for tracking rate limits as no storage was explicitly specified. This is not recommen
ded for production use. See: https://flask-limiter.readthedocs.io#configuring-a-storage-backend for doc
umentation about configuring the storage backend.
  "Using the in-memory storage for tracking rate limits as no storage "
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 262-803-280
```

PASO 5

Se habilita la conexión entre H1 y H2 para que pueda acceder al portal cautivo (menú 1 opción 1)

A través de menu.py (el programa que gestiona el menú de opciones y la inserción de reglas en los OVS), se insertan las reglas necesarias en los switches (por ejemplo, sw3 y sw4) para permitir el tráfico HTTP desde H1 hasta H2. Esto implica añadir flujos en el controlador o directamente en los OVS que habiliten el tráfico entre dichos hosts.



```
Settings - Proyecto final (Copy) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
python
juan-alfonso@HP-ProBook-640-G1:~/Music/SDN/Proyecto final (Copy)$ /bin/python3 "/home/juan-alfonso/Music/SDN/Proyecto final (Copy)/menu (Copy 2).py"
=====
MENU PRINCIPAL
1. Conexiones
2. Arrancar servidor H2
0. Salir
=====
Selecciona una opción: 1
=====
Gestión de Conexiones
=====
1. Habilitar Portal
2. Tráfico FreeRadius-MySQL
0. Volver al Menú Principal
=====
Selecciona una opción: 1
[('00:00:c2:fc:7b:ed:91:4e', 3, 2), ('00:00:42:96:cf:e3:67:4d', 2, 5)]
Configurando flujo Controlador -> H3 (TCP) en el switch 00:00:c2:fc:7b:ed:91:4e
Configurando flujo Controlador -> H3 (ICMP) en el switch 00:00:c2:fc:7b:ed:91:4e
Configurando flujo Controlador -> H3 (ARP) en el switch 00:00:c2:fc:7b:ed:91:4e
Configurando flujo H3 -> Controlador (TCP) en el switch 00:00:c2:fc:7b:ed:91:4e
Configurando flujo H3 -> Controlador (ICMP) en el switch 00:00:c2:fc:7b:ed:91:4e
Configurando flujo H3 -> Controlador (ARP) en el switch 00:00:c2:fc:7b:ed:91:4e
Configurando flujo Controlador -> H3 (TCP) en el switch 00:00:42:96:cf:e3:67:4d
Configurando flujo Controlador -> H3 (ICMP) en el switch 00:00:42:96:cf:e3:67:4d
Configurando flujo Controlador -> H3 (ARP) en el switch 00:00:42:96:cf:e3:67:4d
Configurando flujo H3 -> Controlador (TCP) en el switch 00:00:42:96:cf:e3:67:4d
Configurando flujo H3 -> Controlador (ICMP) en el switch 00:00:42:96:cf:e3:67:4d
Configurando flujo H3 -> Controlador (ARP) en el switch 00:00:42:96:cf:e3:67:4d
Reglas configuradas exitosamente para la comunicación entre el controlador y H3.
=====
Gestión de Conexiones
=====
1. Habilitar Portal
2. Tráfico FreeRadius-MySQL
0. Volver al Menú Principal
=====
Selecciona una opción: 
```

PASO 6

Se habilita la conexión entre H2 y H3 para la autenticación con FreeRADIUS y MySQL

A través de menu.py (el programa que gestiona el menú de opciones y la inserción de reglas en los OVS), se insertan las reglas necesarias en los switches (por ejemplo, sw3 y sw4) para permitir el tráfico HTTP desde H1 hasta H2. Esto implica añadir flujos en el controlador o directamente en los OVS que habiliten el tráfico entre dichos hosts. El servidor H2 debe comunicar las credenciales ingresadas por el usuario a H3, donde se ejecuta FreeRADIUS y la base de datos MySQL. Para ello, se añaden reglas de flujo en los switches (por ejemplo sw4 y sw5) que conectan H2 con H3. Estas reglas permiten el tráfico RADIUS y el acceso a la base de datos, necesarios para validar la autenticación del usuario.

PASO 7

Levantamos el servidor en el programa python menu.py para que escuche a H2 y esperar los servidores y servicios permitidos del usuario

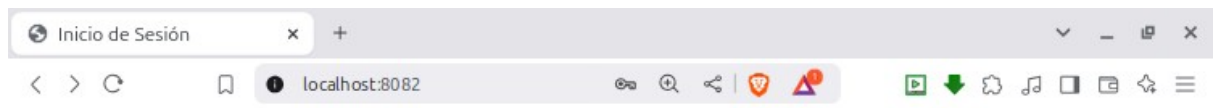
El programa menu.py actúa como intermediario entre H2, el controlador SDN y la aplicación de políticas. Al escuchar las peticiones de H2, menu.py puede determinar qué servicios y flujos se deben habilitar en la red para el usuario autenticado. Una vez recibidas las credenciales y validaciones desde H3, menu.py inserta las reglas definitivas en los OVS.

PASO 8

Comando SSH para el Túnel:

ssh -L 8082:10.0.0.2:80 ubuntu@10.20.12.136 -p 5811

Este comando establece un túnel SSH desde la máquina local del administrador hacia H2. El tráfico enviado al puerto local 8082 se redirige al puerto 80 en 10.0.0.2 (H2). De esta manera, el administrador puede acceder al portal desde su propia máquina local a través de la conexión SSH, facilitando pruebas y diagnósticos.



Portal de Inicio de Sesión

Usuario

Contraseña

Iniciar Sesión

PASO 9 Ingresamos con el usuario y contraseña (credenciales)

El usuario en H1 accede al portal en H2 y proporciona sus credenciales. H2 las envía a H3 (FreeRADIUS + MySQL) para su validación. Esta es la etapa de autenticación, central en el modelo 802.1X.

Inicio de Sesión

localhost:8082

Portal de Inicio de Sesión

Usuario

usuario1

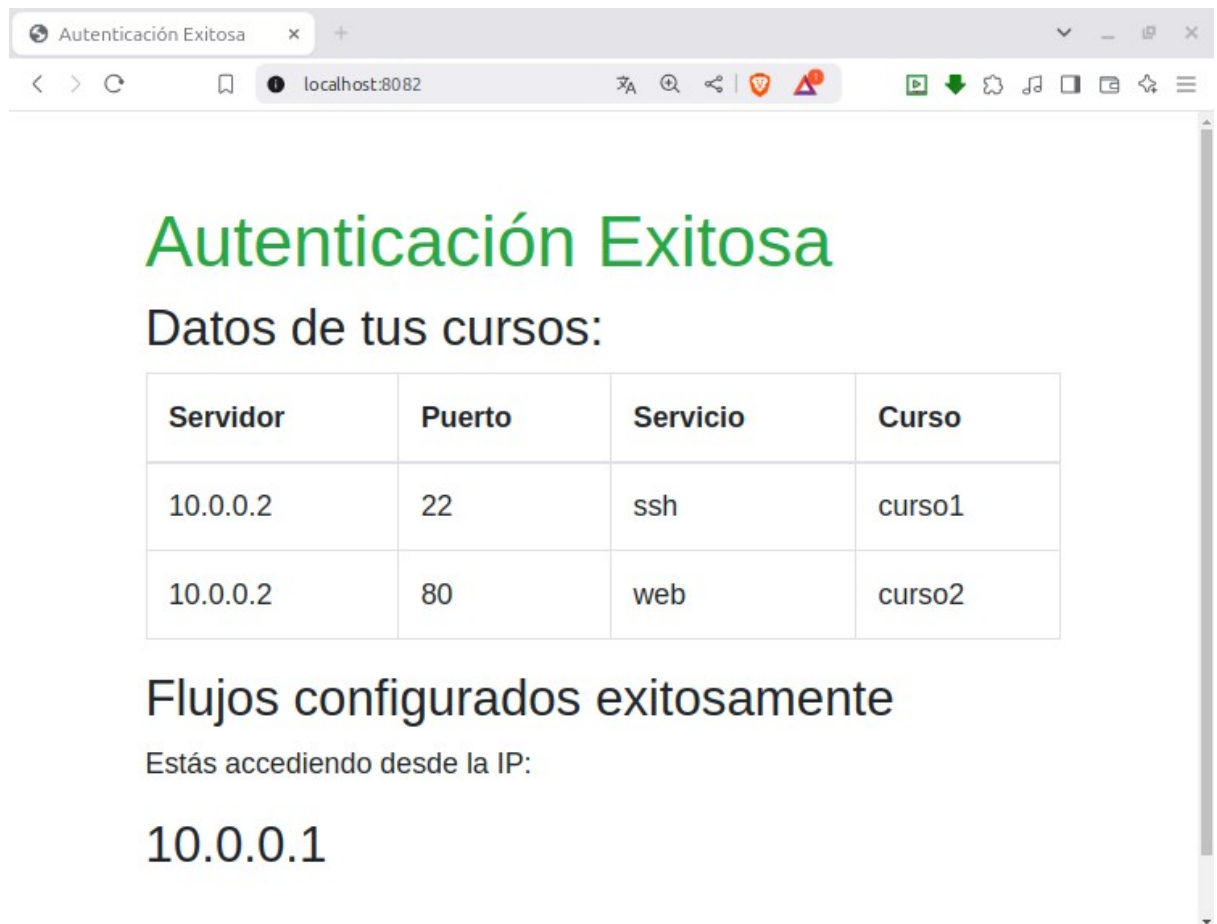
Contraseña

.....

Iniciar Sesión

PASO 10 Ingresamos con el usuario y contraseña (credenciales)

Si las credenciales son correctas, el usuario es redirigido a una página que confirma el acceso a la red. Allí se indican los servidores y puertos a los que el usuario tiene derecho, la dirección IP del cliente, y se informa que se han establecido con éxito las reglas de flujo correspondientes. Esto asegura que el usuario tenga los permisos adecuados según su rol y las políticas definidas.



PASO 11 Ingresamos con el usuario y contraseña (credenciales)

menu.py toma la información obtenida de la autenticación y, en base a las políticas, inserta las reglas de flujo en los switches OVS. Esto transforma el tráfico de tener que ir siempre al controlador a tener rutas específicas aprobadas. Por ejemplo, se permiten SSH a ciertos servidores, HTTP a otros, etc.

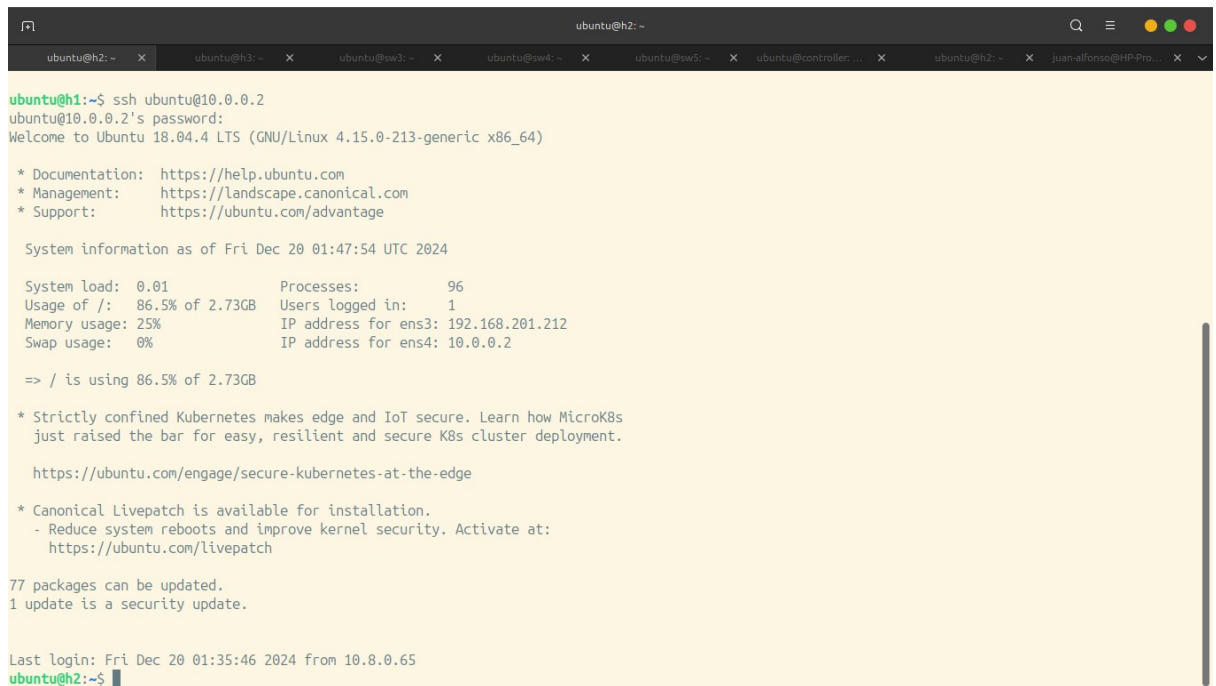

```
ubuntu@sw3:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows sw3
cookie=0x0, duration=116102.871s, table=0, n_packets=81672, n_bytes=7394515, priority=0 actions=CONTROLLER:65535
ubuntu@sw3:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows sw3
cookie=0xa0000003774a7a2, duration=3563.472s, table=0, n_packets=42, n_bytes=8038, send_flow_rem tcp,dl_src=fa:16:3e:41:92:87,dl_dst=fa:16:3e:1c:2f:d4,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_dst=80 actions=output:ens5
cookie=0xa0000002ca11e1a, duration=3563.448s, table=0, n_packets=0, n_bytes=0, send_flow_rem icmp,dl_src=fa:16:3e:41:92:87,dl_dst=fa:16:3e:1c:2f:d4,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:ens5
cookie=0xa000000edac319a, duration=3563.383s, table=0, n_packets=0, n_bytes=0, send_flow_rem icmp,dl_src=fa:16:3e:1c:2f:d4,dl_dst=fa:16:3e:41:92:87,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:ens6
cookie=0xa000000b764ee86, duration=3563.427s, table=0, n_packets=1, n_bytes=60, send_flow_rem arp,arp_spa=10.0.0.1,arp_tpa=10.0.0.2 actions=output:ens5
cookie=0xa00000005f55d06, duration=3563.363s, table=0, n_packets=1, n_bytes=60, send_flow_rem arp,arp_spa=10.0.0.2,arp_tpa=10.0.0.1 actions=output:ens6
cookie=0xa0000000261ae22, duration=3563.404s, table=0, n_packets=35, n_bytes=15430, send_flow_rem tcp,dl_src=fa:16:3e:1c:2f:d4,dl_dst=fa:16:3e:41:92:87,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=80 actions=output:ens6
cookie=0xa00000071be6418, duration=291.005s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_dst=22 actions=output:ens5
cookie=0xa000000f62530cc, duration=290.821s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_dst=80 actions=output:ens5
cookie=0xa0000006c902091, duration=290.972s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=22 actions=output:ens6
cookie=0xa00000066761465, duration=290.793s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=80 actions=output:ens6
cookie=0x0, duration=120491.771s, table=0, n_packets=84756, n_bytes=7674298, priority=0 actions=CONTROLLER:65535
ubuntu@sw3:~$
```

```
ubuntu@sw4:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows sw4
cookie=0x0, duration=116205.250s, table=0, n_packets=55976, n_bytes=7427111, priority=0 actions=CONTROLLER:65535
ubuntu@sw4:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows sw4
cookie=0xa000000a9ca982c, duration=3606.783s, table=0, n_packets=42, n_bytes=8038, send_flow_rem tcp,dl_src=fa:16:3e:41:92:87,dl_dst=fa:16:3e:1c:2f:d4,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_dst=80 actions=output:ens8
cookie=0xa0000000ed1fdd2, duration=3381.565s, table=0, n_packets=3, n_bytes=279, send_flow_rem udp,dl_src=fa:16:3e:1c:2f:d4,dl_dst=fa:16:3e:29:1a:ec,nw_src=10.0.0.2,nw_dst=10.0.0.3,tp_dst=1812 actions=output:ens7
cookie=0xa000000cefbdc92, duration=3381.513s, table=0, n_packets=0, n_bytes=0, send_flow_rem udp,dl_src=fa:16:3e:1c:2f:d4,dl_dst=fa:16:3e:29:1a:ec,nw_src=10.0.0.2,nw_dst=10.0.0.3,tp_dst=1813 actions=output:ens7
cookie=0xa000000504d7fb8, duration=3606.760s, table=0, n_packets=0, n_bytes=0, send_flow_rem icmp,dl_src=fa:16:3e:41:92:87,dl_dst=fa:16:3e:1c:2f:d4,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:ens8
cookie=0xa00000011589338, duration=3606.683s, table=0, n_packets=0, n_bytes=0, send_flow_rem icmp,dl_src=fa:16:3e:1c:2f:d4,dl_dst=fa:16:3e:41:92:87,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:ens5
cookie=0xa000000870853b0, duration=3606.740s, table=0, n_packets=1, n_bytes=60, send_flow_rem arp,arp_spa=10.0.0.1,arp_tpa=10.0.0.2 actions=output:ens8
cookie=0xa000000d598c230, duration=3606.633s, table=0, n_packets=1, n_bytes=60, send_flow_rem arp,arp_spa=10.0.0.2,arp_tpa=10.0.0.1 actions=output:ens5
cookie=0xa000000094f47b6, duration=3381.278s, table=0, n_packets=3, n_bytes=180, send_flow_rem arp,arp_spa=10.0.0.2,arp_tpa=10.0.0.3 actions=output:ens7
cookie=0xa0000001e5e9502, duration=3381.241s, table=0, n_packets=3, n_bytes=180, send_flow_rem arp,arp_spa=10.0.0.3,arp_tpa=10.0.0.2 actions=output:ens8
cookie=0xa00000074b79eac, duration=3606.714s, table=0, n_packets=36, n_bytes=15496, send_flow_rem tcp,dl_src=fa:16:3e:1c:2f:d4,dl_dst=fa:16:3e:41:92:87,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=80 actions=output:ens5
cookie=0xa00000070a79a5e, duration=3381.540s, table=0, n_packets=3, n_bytes=321, send_flow_rem udp,dl_src=fa:16:3e:29:1a:ec,dl_dst=fa:16:3e:1c:2f:d4,nw_src=10.0.0.3,nw_dst=10.0.0.2,tp_src=1812 actions=output:ens8
cookie=0xa0000007d00841e, duration=3381.465s, table=0, n_packets=0, n_bytes=0, send_flow_rem udp,dl_src=fa:16:3e:29:1a:ec,dl_dst=fa:16:3e:1c:2f:d4,nw_src=10.0.0.3,nw_dst=10.0.0.2,tp_src=1813 actions=output:ens8
cookie=0xa000000b0c8a0aa, duration=3381.441s, table=0, n_packets=0, n_bytes=0, send_flow_rem icmp,nw_src=10.0.0.2,nw_dst=10.0.0.3 actions=output:ens7
cookie=0xa000000c65564e, duration=3381.312s, table=0, n_packets=0, n_bytes=0, send_flow_rem icmp,nw_src=10.0.0.3,nw_dst=10.0.0.2 actions=output:ens8
cookie=0xa000000a11b03ae, duration=3381.217s, table=0, n_packets=30, n_bytes=3300, send_flow_rem tcp,nw_src=10.0.0.2,nw_dst=10.0.0.3,tp_dst=3306 actions=output:ens7
cookie=0xa000000f94bf2a6, duration=334.500s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_dst=22 actions=output:ens8
cookie=0xa0000007db2bf5a, duration=334.302s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_dst=80 actions=output:ens8
cookie=0xa000000b17987e, duration=3381.195s, table=0, n_packets=24, n_bytes=2913, send_flow_rem tcp,nw_src=10.0.0.3,nw_dst=10.0.0.2,tp_src=3306 actions=output:ens8
cookie=0xa00000075bcd5cf, duration=334.458s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=22 actions=output:ens5
cookie=0xa0000006fa2c9a3, duration=334.280s, table=0, n_packets=0, n_bytes=0, send_flow_rem tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=80 actions=output:ens5
cookie=0x0, duration=120535.800s, table=0, n_packets=58053, n_bytes=7702994, priority=0 actions=CONTROLLER:65535
ubuntu@sw4:~$
```

PASO 13 Verificación final de la conexión SSH exitosa

La conexión SSH se realiza con éxito, lo que prueba que la política de acceso (permitir SSH desde H1 a H2) ha sido implementada. Esto confirma que el proceso completo—desde la eliminación de módulos dinámicos hasta la

autenticación y la configuración de reglas estáticas—está funcionando correctamente.



```
ubuntu@h2: ~  
ubuntu@h1:~$ ssh ubuntu@10.0.0.2  
ubuntu@10.0.0.2's password:  
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-213-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Fri Dec 20 01:47:54 UTC 2024  
  
System load:  0.01          Processes:      96  
Usage of /:   86.5% of 2.73GB Users logged in: 1  
Memory usage: 25%          IP address for ens3: 192.168.201.212  
Swap usage:   0%           IP address for ens4: 10.0.0.2  
  
=> / is using 86.5% of 2.73GB  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
  just raised the bar for easy, resilient and secure K8s cluster deployment.  
  
  https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
77 packages can be updated.  
1 update is a security update.  
  
Last login: Fri Dec 20 01:35:46 2024 from 10.8.0.65  
ubuntu@h2:~$
```

8. Anexos

8.1. Base de datos

La base de datos `red_sdn` está diseñada para gestionar una red de dispositivos y controladores, además de incluir funcionalidades de autenticación y permisos para usuarios con roles específicos.


```
ubuntu@h3: ~  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_radius |  
+-----+  
| alumnos          |  
| cursos           |  
| nas              |  
| radacct          |  
| radcheck         |  
| radgroupcheck    |  
| radgroupreply    |  
| radpostauth      |  
| radreply         |  
| radusergroup     |  
| usuario_cursos   |  
+-----+  
11 rows in set (0.00 sec)  
  
mysql> SELECT * FROM radcheck;  
+-----+-----+-----+-----+-----+  
| id | username | attribute          | op | value      |  
+-----+-----+-----+-----+-----+  
| 6  | usuario1 | Cleartext-Password | := | password1  |  
| 7  | usuario2 | Cleartext-Password | := | password2  |  
+-----+-----+-----+-----+-----+  
2 rows in set (0.01 sec)
```

```
ubuntu@h3: ~  
| usuario_cursos |  
+-----+  
11 rows in set (0.01 sec)  
  
mysql> SELECT * FROM alumnos;  
+-----+-----+-----+-----+  
| id | nombre      | codigo | mac                |  
+-----+-----+-----+-----+  
| 4 | Juan Perez  | 20125424 | fa:16:3e:41:92:87 |  
| 5 | John Smith  | 20041321 | 00:44:11:2f:33:d8 |  
| 6 | Luisa Marvel | 20080621 | 00:44:11:3f:22:c3 |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM cursos;  
+-----+-----+-----+-----+-----+  
| id | nombre_curso | servidor_ip | puerto | servicio |  
+-----+-----+-----+-----+-----+  
| 1 | curso1       | 10.0.0.2    | 22     | ssh      |  
| 2 | curso2       | 10.0.0.2    | 80     | web      |  
| 3 | curso3       | 10.0.0.3    | 22     | ssh      |  
+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM usuario_cursos;  
+-----+-----+-----+  
| id | username | curso_id |  
+-----+-----+-----+  
| 1 | usuario1 | 1        |  
| 2 | usuario1 | 2        |  
| 3 | usuario2 | 3        |  
+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> 
```