

Universidad de Murcia

Grado en Ingeniería Informática

4º Curso
Curso 2017/2018

LEGO

Práctica Final

Profesor: GABRIEL LOPEZ MILLAN
Profesor: OSCAR CANOVAS REVERTE
Fecha: 17/12/2017
Convocatoria: Enero

Valiantsin Kivachuk: valiantsin.kivachuk@um.es



1. Introducción	3
2. Docker	3
2.1 Docker Compose	5
2.2 Estructura del proyecto	10
2.3 up.sh	11
2.4 Topología de red	17
2.4.1 Bridge driver	17
2.4.2 Macvlan driver	17
2.4.3 DD-WRT	19
2.4.3.1 VLAN	19
2.4.3.2 WiFi	22
2.4.3.3 DHCP	23
2. Diseño	24
SNMP	25
x509v3	25
LDAP	30
RADIUS	31
Asterisk	32
Owncloud	33
3. Servicios	35
3.1 SNMP	35
3.1.1 Definiciones	35
3.1.1.1 SNMPv1	35
3.1.1.2 SNMPv2	36
3.1.1.3 SNMPv3	36
3.1.2 Archivos de configuración	36
3.1.3 Despliegue	39
3.1.4 Trazas NET-SNMP	41
3.1.4.1 SNMPGET	42
3.1.4.2 SNMPWALK	43
3.1.4.3 SNMPPBULKWALK	45
3.1.4.4 SNMPTTRANSLATE	46
3.1.4.5 SNMPPGETNEXT	48
3.1.4.6 SNMPTABLE	49
3.1.4.7 SNMPPBULKGET	51
3.1.4.8 SNMPTRAP	52
3.1.4.9 SNMPv3	53
3.1.4.9.1 SNMPv3 Cleartext	54
3.1.4.9.2 SNMPv3 Auth	55
3.1.4.9.3 SNMPv3 Auth + Crypt	56

3.2 Asterisk	59
3.2.2 Archivos de configuración	59
3.2.X Despliegue	61
3.2.3 Trazas Asterisk	64
3.2.3.1 SIP REGISTER	64
3.2.3.2 SIP INVITE	67
3.3 Owncloud	71
3.3.1 Archivos de configuración	71
3.3.2 Despliegue	73
3.4 FreeRADIUS	76
3.4.1 Archivos de configuración	76
3.4.2 Despliegue	79
3.4.3 Trazas RADIUS	81
3.4.3.1 Identity Request	82
3.4.3.2 Identity Response	82
3.4.3.3 EAP Request	83
3.4.3.4 EAP Response	84
3.4.3.4 Radius	85
3.5 OpenLDAP	88
3.5.1 Archivos de configuración	88
3.5.2 Despliegue	91
3.5.3 Trazas OpenLDAP	93
3.5.3.1 LDAPSEARCH	93
3.5.3.2 LDAPADD	96
3.5.3.3 LDAPMODIFY	100
3.5.3.4 LDAPDELETE	103
3.6 OCSP	106
3.6.1 Archivos de configuración	106
3.6.2 Despliegue	107
3.6.3 Trazas OCSP	108
4 Mejoras pendientes	114
1. Migrar a DNS en Docker	114
2. TLS over SNMP	115
3. PKI	115
5. Manual de uso	115
6. Bibliografía	118

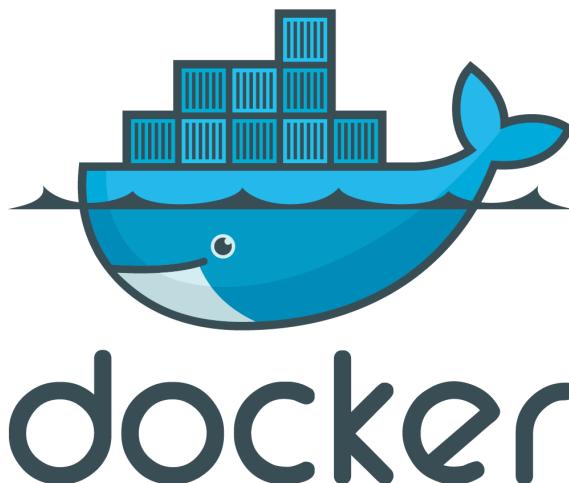
1. Introducción

La práctica descrita a continuación consiste en la configuración y puesta en marcha del conjunto de servicios telemáticos y servicios vistos en la asignatura de STA y TCI.

Esta configuración y puesta en marcha se ha realizado sobre el escenario de red propuesto en el Proyecto LEGO, y sobre el cual se ha ido trabajando a lo largo de las asignaturas.

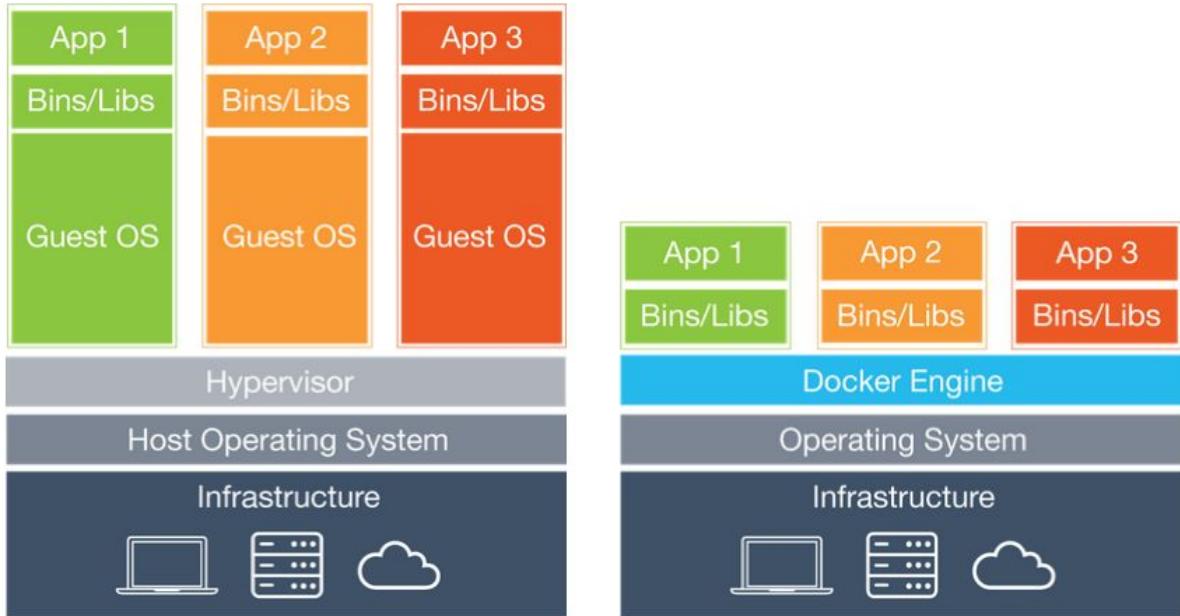
Cabe destacar que la implementación se ha hecho completamente virtualizado, a través de la tecnología *Docker* con pequeños cambios para simular la topología LEGO.

2. Docker



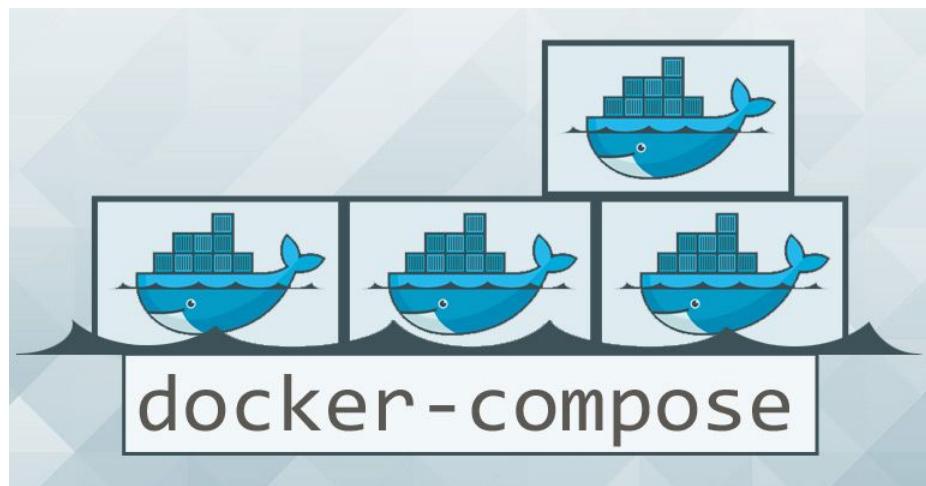
Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo.

Uno de los principales motivos para realizar este proyecto en Docker fue el ahorro de recursos para mantener una gran cantidad de servicios operativos frente a las tradicionales máquinas virtuales.

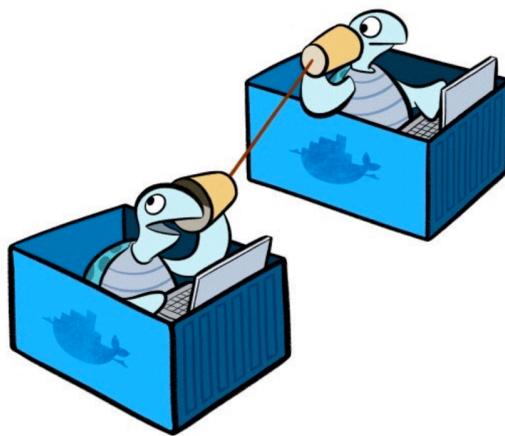


De esta forma, podemos tener un *servicio* (máquina Ubuntu, Asterisk, etc) funcionando de forma virtualizada en nuestro ordenador sin la sobrecarga de un sistema operativo.

Para el despliegue de los servicios, se hace uso de **Docker Compose**. Docker Compose es un mecanismo permite utilizar varias imágenes y comunicarlas, para obtener los requisitos necesarios para hacer funcionar nuestra infraestructura.



Además, Docker permite establecer la comunicación entre los distintos contenedores para definir las topologías de forma eficiente.



2.1 Docker Compose

Es el esqueleto de nuestro proyecto. En él se definen las máquinas y su relación entre ellas.

```
version: '3'
services:

# Org1 #####
um_ca:
    build: ./um/ca/
    hostname: ca
    domainname: um.es
    volumes:
    - um-certs:/certs
    network_mode: none

um_crl:
    image: halverneus/static-file-server
    hostname: crl
    domainname: um.es
    environment:
    - PORT=80
    volumes:
    - lego_ca:/web
    networks:
    red1:
        aliases:
        - crl.um.es
        ipv4_address: 192.168.251.99

um_snmp:
    build: ./um/snmp/
    hostname: snmp
    domainname: um.es
    networks:
    red1:
        aliases:
        - snmp.um.es
        ipv4_address: 192.168.251.100

um_db:
```

```

image: postgres:10-alpine
hostname: db
domainname: um.es
environment:
- POSTGRES_PASSWORD=owncloud
networks:
red1:
    aliases:
    - db.um.es
    ipv4_address: 192.168.251.4

um_radius:
    build: ./um/radius/
    hostname: radius
    domainname: um.es
    volumes:
    - um-certs-radius:/etc/freeradius/keys/
depends_on:
- "um_snmp"
- "um_opendap"
networks:
red1:
    aliases:
    - radius.um.es
    ipv4_address: 192.168.251.3

um_owncloud:
    build: ./um/owncloud
    hostname: owncloud
    domainname: um.es
    volumes:
    - um-certs-owncloud:/etc/apache2/keys/
depends_on:
- "um_db"
- "um_opendap"
- "um_crl"
- "um_snmp"
links:
- um_db:owncloud-db
ports:
- "80:80"
- "443:443"
environment:
- OWNCLLOUD_DB_HOST=db.um.es
- OWNCLLOUD_DB_TYPE=pgsql
- OWNCLLOUD_DB_NAME=owncloud
- OWNCLLOUD_DB_USERNAME=postgres
- OWNCLLOUD_DB_PASSWORD=owncloud
- OWNCLLOUD_ADMIN_USERNAME=owncloud
- OWNCLLOUD_ADMIN_PASSWORD=owncloud
- OWNCLLOUD_VOLUME_CERTS=/etc/apache2/keys/
networks:
red1:
    aliases:
    - owncloud.um.es
    ipv4_address: 192.168.251.5

um_opendap:
    build: ./um/openldap/
    command: --loglevel info
    hostname: ldap
    domainname: um.es
depends_on:
- "um_snmp"
environment:

```

```

- LDAP_ORGANISATION=Universidad de Murcia
- LDAP_DOMAIN=um.es
- LDAP_ADMIN_PASSWORD=um_password
- HOSTNAME=ldap.um.es #Bug
- LDAP_TLS=true
- LDAP_TLS_CRT_FILENAME=ldap-cert.pem
- LDAP_TLS_KEY_FILENAME=ldap-key.pem
- LDAP_TLS_CA_CRT_FILENAME=um.pem
- LDAP_TLS_ENFORCE=false
volumes:
- um-certs-ldap:/container/service/slapd/assets/certs
networks:
red1:
    aliases:
    - ldap.um.es
    ipv4_address: 192.168.251.6

um_pc1:
    build: ./um/pc1/
    depends_on:
    - "um_snmp"
networks:
red1:
    ipv4_address: 192.168.251.20

um_voip: #VoIP
    build: ./um/voip/
    hostname: voip
    domainname: um.es
    depends_on:
    - "um_snmp"
    - "um_opendLDAP"
volumes:
- um-certs-voip:/certs
- lego_ca:/ca_ssl
networks:
red1:
    aliases:
    - voip.um.es
    ipv4_address: 192.168.251.2

# Org2 #####
upm_ca:
    build: ./um/ca/
    hostname: ca
    domainname: upm.es
    volumes:
    - upm-certs:/certs
    network_mode: none

upm_crl:
    image: halverneus/static-file-server
    hostname: crl
    domainname: upm.es
    environment:
    - PORT=80
    volumes:
    - lego_ca:/web
    networks:
    red2:
        aliases:
        - crl.upm.es
        ipv4_address: 192.168.252.99

```

```

upm_snmp:
    build: ./upm/snmp/
    hostname: snmp
    domainname: upm.es
    networks:
        red2:
            aliases:
                - snmp.upm.es
            ipv4_address: 192.168.252.100

upm_voip: #VoIP
    build: ./upm/voip/
    hostname: voip
    domainname: upm.es
    depends_on:
        - "upm_opendap"
        - "upm_snmp"
    volumes:
        - upm-certs-voip:/certs
        - lego_ca:/ca_ssl
    networks:
        red2:
            aliases:
                - voip.upm.es
            ipv4_address: 192.168.252.2

upm_radius:
    build: ./upm/radius/
    hostname: radius
    domainname: upm.es
    volumes:
        - upm-certs-radius:/etc/freeradius/keys/
    depends_on:
        - "upm_snmp"
    networks:
        red2:
            aliases:
                - radius.upm.es
            ipv4_address: 192.168.252.3

upm_opendap:
    build: ./upm/opendap/
    command: --loglevel info
    hostname: ldap
    domainname: upm.es
    depends_on:
        - "upm_snmp"
    environment:
        - LDAP_ORGANISATION=Universidad Politecnica de Madrid
        - LDAP_DOMAIN=upm.es
        - LDAP_ADMIN_PASSWORD=upm_password
        - HOSTNAME=ldap.upm.es #Bug
        - LDAP_TLS=true
        - LDAP_TLS_CRT_FILENAME=ldap-cert.pem
        - LDAP_TLS_KEY_FILENAME=ldap-key.pem
        - LDAP_TLS_CA_CRT_FILENAME=upm.pem
        - LDAP_TLS_ENFORCE=false
    volumes:
        - upm-certs-ldap:/container/service/slapd/assets/certs
    networks:
        red2:
            aliases:
                - ldap.upm.es
            ipv4_address: 192.168.252.6

```

```

# PERSISTENCE #####
volumes:
    lego_ca:
        external:
            name: "lego_ca"

    um-certs:
        external:
            name: "um-certs"
    um-certs-voip:
        external:
            name: "um-certs-voip"
    um-certs-owncloud:
        external:
            name: "um-certs-owncloud"
    um-certs-ldap:
        external:
            name: "um-certs-ldap"
    um-certs-radius:
        external:
            name: "um-certs-radius"

    upm-certs:
        external:
            name: "upm-certs"
    upm-certs-voip:
        external:
            name: "upm-certs-voip"
    upm-certs-owncloud:
        external:
            name: "upm-certs-owncloud"
    upm-certs-ldap:
        external:
            name: "upm-certs-ldap"
    upm-certs-radius:
        external:
            name: "upm-certs-radius"

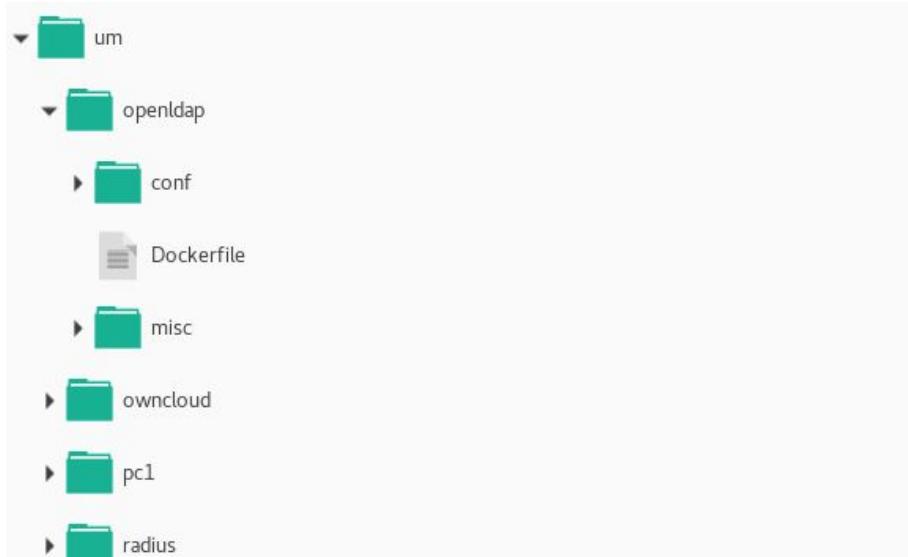
# NETWORKS #####
networks:
    red1:
        driver: bridge
        driver_opts:
            com.docker.network.bridge.enable_icc: "true"
        ipam:
        config:
        - subnet: 192.168.251.0/24

    red2:
        driver: bridge
        driver_opts:
            com.docker.network.bridge.enable_icc: "true"
        ipam:
        config:
        - subnet: 192.168.252.0/24

```

2.2 Estructura del proyecto

Para poder desplegar todos los servicios de todas las organizaciones, el proyecto se estructura de forma jerárquica en los directorios:



Para cada organización, tenemos un carpeta. Dentro de cada organización, cada carpeta representa el servicio que posee. Dentro del servicio, encontramos 3 elementos principales:

1. **conf**: Directorio con los archivos de configuración que se insertaran en el servicio antes de despegarlo
2. **misc**: Directorio con diversos scripts/configuraciones para levantar el servicio.
3. **Dockerfile**: Núcleo del servicio. Define cómo se despliega el servicio y sus dependencias.

De esta forma, tal como se muestra en la imagen, para el servicio *OpenLDAP* de la Organización **UM**, desplegamos el servicio con el siguiente Dockerfile:

```
FROM vk496/openldap
LABEL maintainer="valiantsin.kivachuk@um.es"

ENV DEBIAN_FRONTEND noninteractive

#TODO ffix duplicated sourcelist
RUN echo "deb http://deb.debian.org/debian stretch main non-free contrib" >> /etc/apt/sources.list && \
echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
        apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
        apt-get update && apt-get --no-install-recommends -y install apt-fast && apt-fast
install -y \
net-tools \
```

```

traceroute \
netcat \
dnsutils \
iputils-ping \
curl \
nano \
supervisor \
snmpd snmp snmp-mibs-downloader \
&& rm -rf /var/lib/apt/lists/*
COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf
COPY conf/snmpd.conf /etc/snmp/snmpd.conf

#Extend LDAP Schema
COPY conf/um.ldif /container/service/slapd/assets/config/bootstrap/ldif/custom/

CMD ["/usr/bin/supervisord"]

```

Tal como indica el archivo, usamos la imagen **vk496/openldap**, a la cual instalamos varios paquetes, copiamos los archivos de configuración a la máquina y lanzamos el servicio **supervisord**, quien levantará el servicio.

Es importante remarcar la importancia del programa **Supervisord**, presente en cada uno de los servicios. Un servicio en Docker se lanza con un propósito, y se termina cuando deja de ejecutarse. Para evitar que el servicio se apague cuando el programa principal del servicio se cierre por cualquier motivo, se hace uso de *Supervisord*.

Supervisord no es más que un framework encargado de lanzar diversos programas que definamos en su configuración. En caso de que se cierra, el contenedor de Docker seguirá activo porque *supervisor* se estará ejecutando. Esto se consigue con la siguiente entrada en los **Dockerfile**:

```
CMD ["/usr/bin/supervisord"]
```

2.3 up.sh

El “pequeño” script **up.sh**, empezó siendo pequeño, pero a medida que ha avanzado el proyecto, se ha hecho bastante complejo. Es por ello, que se merece una sección en este documento.

Iremos describiendo los bloques más importantes dentro de este inicializador. Para empezar, tenemos:

```

#!/bin/bash
set -e
trap 'exit 130' INT #Exit if trap Ctrl+C

#Software necesario

```

```

software=( sudo bash docker-compose sysctl docker iptables grep awk basename cat cut
sed)

for i in "${software[@]}"; do
    if ! hash $i 2>/dev/null; then
        echo -e "missing $i"
        exit 1
    fi
done

#Root
[ "$UID" -eq 0 ] || exec sudo bash "$0" "$@"

```

Aquí definimos que el script debe salir en caso de que algún comando falle. Definimos una lista de comandos que deben estar presentes en el sistema y se relanza a sí mismo como root.

Esto último se hace así porque los privilegios de administrador son necesarios cuando se despliega el proyecto completamente. Sin embargo, desplegar el proyecto en limpio puede durar entre 30-120min, por lo que la sesión de sudo caducaría (no todos los comandos deben ejecutarse como root)

```

help_usage() {
    echo "Invalid option: -$OPTARG" >&2
    echo -e "Use:\n$0 -b\n$0 -m iface1 -m iface2" >&2
}

while getopts ":m: :b" opt; do
    case $opt in
        b)
            echo "Installing bridge driver"
            base=$(sed '/# NETWORKS #####/Q' docker-compose.yml)
            echo "$base" >docker-compose.yml
            cat .docker-compose.BRIDGE >>docker-compose.yml
        ;;
        m)
            multi+=("$OPTARG")
            mode_macvlan=1
        ;;
        *)
            help_usage
            exit 1
        ;;
    esac
done

if [ $mode_macvlan ]; then
    if [ ${#multi[@]} -ne 2 ]; then
        help_usage
        exit 1
    fi

    shift ${((OPTIND -1))}

    echo "Installing macvlan driver - Review parent interfaces..."
    base=$(sed '/# NETWORKS #####/Q' docker-compose.yml)"

```

```

echo "$base" >docker-compose.yml
cat .docker-compose.MACVLAN >>docker-compose.yml

sed -i "s/enp1s0.3/${multi[0]}/g" docker-compose.yml
sed -i "s/enp1s0.4/${multi[1]}/g" docker-compose.yml

fi

driver=$(sed -n -e '/# NETWORKS #/,${p}' docker-compose.yml | grep -i driver: -m1 | cut -d: -f2 | sed 's/ //g')

if [ $driver == "macvlan" ]; then
    color='\033[0;31m'
elif [ $driver == "bridge" ]; then
    color='\033[0;33m'
else
    echo "Strange network driver. Exiting...."
    exit 1
fi

echo -e "Using ${color}${driver}\033[0m network driver"
sleep 1.5

```

A continuación, hacemos uso de `getopts` para parsear argumentos. Esencialmente, el script se puede lanzar de dos formas:

```

./up.sh -b para topología en modo Bridge
./up.sh -m eth0.3 -m eth0.4 para topología en modo Macvlan

```

Para conseguir eso, usaremos plantillas de la sección de red del Docker Compose. Así, dependiendo del modo, se usará la plantilla en el archivo `docker-compose.yml` con las correspondientes variables que le pase el usuario.

Después, se despliega la infraestructura PKI de ambas organizaciones:

```

domains=(
    "um:192.168.251"
    "upm:192.168.252"
)

for domain in "${domains[@]}"; do
    dom=$(echo $domain | cut -d: -f1) #Get domain
    ip_range=$(echo $domain | cut -d: -f2) #Get IP range

    servicios=(
        "distribution:$ip_range.99"
        "owncloud:$ip_range.5"
        "voip:$ip_range.2"
        "ldap:$ip_range.6"
        "radius:$ip_range.3"
    )

    #GENERAL SSL ARGUMENTS
    LEGO_CA_KEY="$dom-key.pem"
    LEGO_CA_CERT="$dom.pem"
    LEGO_CA_SUBJECT="$dom"

```

```

LEGO_CA_EXPIRE="10000"
LEGO_SSL_CONFIG="openssl.cnf"
LEGO_SSL_CRL="http://distribution.$dom.es/$dom.crl,http://$ip_range.99/$dom.crl"
LEGO_SSL_OCSP="http://$ip_range.99:8080,http://distribution.$dom.es:8080"
LEGO_SSL_SIZE="2048"
LEGO_SSL_EXPIRE="360"

if ! sudo -u $SUDO_USER docker volume inspect $dom-certs 2>/dev/null >/dev/null;
then
    sudo -u $SUDO_USER docker volume create $dom-certs #All certs

#Generate CA
sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs \
    -e CA_KEY="$LEGO_CA_KEY" \
    -e CA_CERT="$LEGO_CA_CERT" \
    -e CA SUBJECT="$LEGO_CA_SUBJECT" \
    -e CA_EXPIRE="$LEGO_CA_EXPIRE" \
    -e SSL_SIZE="4096" \
    -e SSL_KEY="revoked_key.pem" \
    -e SSL_CSR="revoked_key.csr" \
    -e SSL_CERT="revoked_cert.pem" \
    -e SSL_CRL="$LEGO_SSL_CRL" \
    -e SSL_OCSP="$LEGO_SSL_OCSP" \
vk496/omgwtfssl

sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs vk496/omgwtfssl cat
$dom.pem > $dom.pem #Get CA outside Docker

#Revoke default SSL key
echo "REVOKE default server cert & keys"
sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs vk496/omgwtfssl bash -c
"openssl ca -config openssl.cnf -revoke revoked_cert.pem -keyfile $LEGO_CA_KEY -cert
$LEGO_CA_CERT"
echo "OK"

for servicio in "${servicios[@]}"; do
    subdom=$(echo $servicio | cut -d: -f1) #Get domain of service
    ip_serv=$(echo $servicio | cut -d: -f2) #Get ip of service

    LEGO_SSL_KEY="$subdom-key.pem"
    LEGO_SSL_CSR="$subdom-key.csr"
    LEGO_SSL_CERT="$subdom-cert.pem"
    LEGO_SSL SUBJECT="$ip_serv"
    LEGO_SSL_DNS="$subdom.$dom.es"
    LEGO_SSL_IP="$ip_serv"

    sudo -u $SUDO_USER docker volume create $dom-certs-$subdom #Service certs

#Generate cert
sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs \
    -e CA_KEY="$LEGO_CA_KEY" \
    -e CA_CERT="$LEGO_CA_CERT" \
    -e CA SUBJECT="$LEGO_CA_SUBJECT" \
    -e SSL_KEY="$LEGO_SSL_KEY" \
    -e SSL_CSR="$LEGO_SSL_CSR" \
    -e SSL_CERT="$LEGO_SSL_CERT" \
    -e SSL_SIZE="$LEGO_SSL_SIZE" \
    -e SSL_EXPIRE="$LEGO_SSL_EXPIRE" \
    -e SSL SUBJECT="$LEGO_SSL SUBJECT" \
    -e SSL_DNS="$LEGO_SSL_DNS" \
    -e SSL_IP="$LEGO_SSL_IP" \
    -e SSL_CRL="$LEGO_SSL_CRL" \
    -e SSL_OCSP="$LEGO_SSL_OCSP" \
vk496/omgwtfssl

```

```

        if [[ $subdom == "distribution" ]]; then #distribution must have special
extension. Resign
            echo "Special resign for distribution service"
            sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs \
                vk496/omgwtfssl bash -c "openssl ca -batch -config
${LEGO_SSL_CONFIG} -keyfile ${LEGO_CA_KEY} -cert ${LEGO_CA_CERT} -outdir . -in
${LEGO_SSL_CSR} -out ${LEGO_SSL_CERT} -extensions v3_OCSP -create_serial -extfile
${LEGO_SSL_CONFIG} -days ${LEGO_SSL_EXPIRE} -notext -preserveDN"
            fi

            #Aislate the keys from CA and full chain CA
            sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs -v
$dom-certs-$subdom:/service_certs vk496/omgwtfssl bash -c "cp /certs/{$subdom*,$dom.pem}
/service_certs && cat /certs/{$subdom-cert.pem,$dom.pem} >
/service_certs/$subdom-cert-full.pem && openssl dhparam -out /service_certs/dh 1024"
            done
            fi

            if ! sudo -u $SUDO_USER docker volume inspect $dom-certs-user 2>/dev/null
>/dev/null; then
                sudo -u $SUDO_USER docker volume create $dom-certs-user #Public CA keys

                #Generate User certificates
                sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs -v
$dom-certs-user:/certs_user vk496/omgwtfssl bash -ce "\"
                    openssl genrsa -out /certs_user/user-key.pem $LEGO_SSL_SIZE > /dev/null &&
\
                    openssl req -new -key /certs_user/user-key.pem -out
/certs_user/user-key.csr -subj \"CN=user1@$dom.es\" -config ${LEGO_SSL_CONFIG} >
/dev/null && \
                    openssl ca -batch -config ${LEGO_SSL_CONFIG} -keyfile ${LEGO_CA_KEY} -cert
${LEGO_CA_CERT} -outdir . -in /certs_user/user-key.csr -out /certs_user/user-cert.pem
-extensions usr_cert -create_serial -extfile ${LEGO_SSL_CONFIG} -days ${LEGO_SSL_EXPIRE}
-notext -preserveDN"

                    sudo -u $SUDO_USER docker run --rm -v $dom-certs-user:/certs vk496/omgwtfssl cat
user-cert.pem > $dom-user-cert.pem #Get cert outside docker
                    sudo -u $SUDO_USER docker run --rm -v $dom-certs-user:/certs vk496/omgwtfssl cat
user-key.pem > $dom-user-key.pem #Get keys outside docker

                fi
            done

        if ! sudo -u $SUDO_USER docker volume inspect lego_ca 2>/dev/null >/dev/null; then
            sudo -u $SUDO_USER docker volume create lego_ca #Public CA keys

            for domain in "${domains[@]}"; do
                dom=$(echo $domain | cut -d: -f1) #Get domain

                #Prepare public CA certs
                sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs -v lego_ca:/public_ca
vk496/omgwtfssl bash -c "cp /certs/$dom.pem /public_ca && cp /certs/$dom.pem
/public_ca/\$(openssl x509 -in /certs/$dom.pem -noout -hash).0"

                #Generate CRL
                sudo -u $SUDO_USER docker run --rm -v $dom-certs:/certs -v
$dom-certs-distribution:/distribution vk496/omgwtfssl bash -c "openssl ca -genCRL
-config /certs/openssl.cnf -keyfile /certs/$dom-key.pem -cert $dom.pem -out
/distribution/$dom.crl && cp .db.pem{,.attr} /distribution/"

            done
        fi
    
```

Para dicho cometido, se utiliza la imagen **vk496/omgwtfssl**, un fork de *paulczar/omgwtfssl* con modificaciones para nuestro proyecto. Destacar que para cada servicio, se utiliza un volumen distinto, para tener un aislamiento entre los distintos certificados.

Un volumen es una unidad de persistencia que se puede conectar a un contenedor. La idea es tener una forma de guardar algo de un contenedor para que no se pierda en el reinicio del servicio.

Inicialmente, se despliegan los CAs autofirmados en su volumen y se copia sus certificados a otro volumen “público” para cualquier servicio.

Después, se revoca un certificado de prueba, siendo esto útil para posteriores pruebas.

Después, para cada servicio, se genera su correspondiente certificado en su volumen. Remarcar que para el servicio de *redistribution*, el .csr se vuelve a firmar manualmente para poder utilizar una extensión distinta a la de los demás certificados.

A continuación, se generan certificados de usuario y se guardan en el archivo host, para utilizarlos cómodamente en pruebas fuera del entorno virtualizado.

Por último, se generan los CRLs de cada organización y se trasladan al volumen del servicio de distribución para publicarlos en cada una de las organizaciones.

```
sudo -u $SUDO_USER docker-compose up --build -d #Deploy services
sudo sysctl -w net.ipv4.ip_forward=1 #Enable IP Forwarding

#Get list of virtual interfaces
com=
while read -r line; do
    if [[ $line ]]; then
        com="${com}\|$line"
    fi
done < <(sudo -u $SUDO_USER docker network ls | grep "$(basename "$(pwd)")" |grep -v default| awk '{print $1}')

#Disable iptables for communicate between virtual interfaces
if [[ $com ]]; then
    while read -r line; do
        echo "iptables ${line/-A DOCKER-ISOLATION/-D DOCKER-ISOLATION}"
        eval "iptables ${line/-A DOCKER-ISOLATION/-D DOCKER-ISOLATION}"
    done < <(eval "iptables-save | grep \"DOCKER-ISOLATION\" | grep -i \"\|-j DROP\" |
grep -v \"docker0\" | grep \"${com:2:-1}\\"")
fi
```

Finalmente, se despliega el proyecto con docker compose. Cuando termine, realizamos pequeños cambios para que el modo Bridge funcione correctamente.

Concretamente, habilitamos el IP Forwarding para que nuestra máquina host haga de enrutador entre los distintos contenedores (los hosts virtualizados representados como servicios en este proyecto).

Además, eliminamos algunas reglas de firewall concretas para permitir plena comunicación entre los hosts de ambas redes.

2.4 Topología de red

Uno de los requisitos del proyecto es el despliegue de las organizaciones en redes aisladas, con un router entre ambas que enrute el tráfico.

A pesar de poder definir en el Docker Compose ambas topologías y qué equipos van a pertenecer a ellas, simular una auténtica red no es posible de forma nativa: Docker parte del principio de que los servicios deben estar aislados unos de otros, a no ser que se indique una relación concreta y puntual.

2.4.1 Bridge driver

Todo contenedor está conectado por defecto a una interfaz de red **docker0**, que puede estar en *modo bridge* con otras interfaces definidas por nosotros (para dar salida a Internet).

Formalmente, en Docker esto se hace definiendo una red e indicando el driver para esa red (*bridge* en este caso). Tras levantar las interfaces, se usan **IPTables** para aislar los contenedores a través de **DOCKER-ISOLATION**:

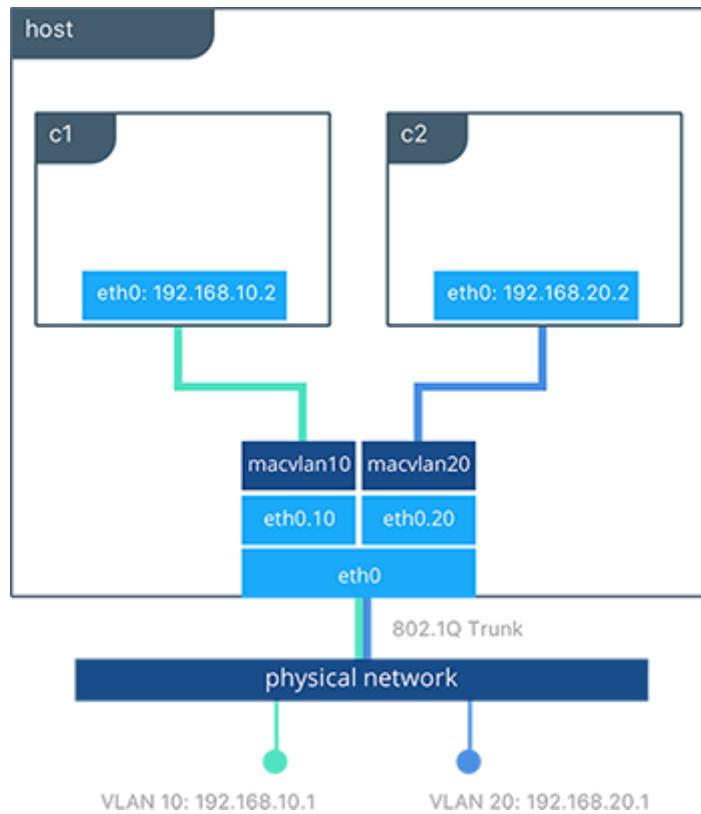
```
[vk496@vk496-pc lego]$ sudo iptables-save | grep DOCKER-ISOLATION
:DOCKER-ISOLATION - [0:0]
-A FORWARD -j DOCKER-ISOLATION
-A DOCKER-ISOLATION -i br-e544ca652aed -o br-74c1f5302aa9 -j DROP
-A DOCKER-ISOLATION -i br-74c1f5302aa9 -o br-e544ca652aed -j DROP
-A DOCKER-ISOLATION -i docker0 -o br-74c1f5302aa9 -j DROP
-A DOCKER-ISOLATION -i br-74c1f5302aa9 -o docker0 -j DROP
-A DOCKER-ISOLATION -i docker0 -o br-e544ca652aed -j DROP
-A DOCKER-ISOLATION -i br-e544ca652aed -o docker0 -j DROP
-A DOCKER-ISOLATION -j RETURN
```

Nuestro script **up.sh** se encarga de eliminar todas las reglas que afecten a nuestros servicios, para tener plena libertad de comunicación entre ambas topologías de red. Es indispensable ejecutarlo sobre **Linux** para que esto sea posible.

2.4.2 Macvlan driver

Conforme se avanza en el proyecto, llega un momento en el que es necesario poner en escena un router. Desde el router, podemos crear dos redes VLAN para simular la separación. Sin embargo, hay que hacerlo funcionar con toda la infraestructura virtual. Aquí entra el driver *macvlan*.

A grandes rasgos, lo que conseguimos es conectar las interfaces virtuales del router con interfaces virtuales de los contenedores, haciendo posible “conectarlas” par a par:



En esta imagen podemos ver más claramente cómo funciona. La *VLAN 10* del router se conecta a la *macvlan10* de Docker. De esta forma, la red *192.168.10.0/24* es “compartida” entre ambos, siendo **completamente transparente para los hosts**.

En nuestro caso, el script **up.sh** recibe un parámetro *-b* o *-m* para usar el driver bridge o macvlan respectivamente. Los parámetros los coge de los archivos ocultos *.docker-compose.MACVLAN* y *.docker-compose.BRIDGE*.

Cabe destacar que para el uso de macvlan, debemos configurar previamente nuestro router y equipo host. A continuación, cambiar la interfaz **enp1s0.X** por las que corresponda en el apartado *parent*:

```
# NETWORKS #####
networks:
    red1:
        driver: macvlan
        driver_opts:
            parent: enp1s0.3
        ipam:
            config:
                - subnet: 192.168.251.0/24

    red2:
        driver: macvlan
        driver_opts:
            parent: enp1s0.4
        ipam:
            config:
                - subnet: 192.168.252.0/24
```

2.4.3 DD-WRT

Para poder utilizar el modo macvlan, es requisito crear redes VLAN, además de otros elementos avanzados, que el firmware stock de la gran mayoría de routers domésticos no dispone.

Para ello, se ha optado por utilizar el firmware **DD-WRT** debido a su simplicidad para poner todo en marcha mediante interfaz gráfica. Las partes críticas de configuración para el proyecto LEGO son las siguientes:

2.4.3.1 VLAN

VLAN	Port					Assigned To Bridge
	W	1	2	3	4	
0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	LAN
1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None
2	<input type="checkbox"/>	None				
3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	None
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None
5	<input type="checkbox"/>	None				
6	<input type="checkbox"/>	None				

Dependiendo del puerto Ethernet, estaremos dentro de una red concreta. Así pues:

- Ethernet 1: Red del router (**192.168.2.0/24**)
- Ethernet 2: Universidad de Murcia (**192.168.251.0/24**)
- Ethernet 3: Universidad Politecnica de Madrid (**192.168.252.0/24**)
- Ethernet 4: VLAN de ambas organizaciones (**192.168.251.0/24** y **192.168.252.0/24**)

Además de la definición de la VLAN (vlan3 y vlan4), es necesario poner dichas interfaces en modo bridge con sus respectivas interfaces WiFi:

The screenshot shows a network configuration interface with the following sections:

- VLAN Tagging**: A section for managing VLAN tags, currently empty.
- Bridging**:
 - Create Bridge**:
 - Bridge 0: IP Address 192.168.251.1, Subnet Mask 255.255.255.0
 - Bridge 1: IP Address 192.168.252.1, Subnet Mask 255.255.255.0
 - Assign to Bridge**:
 - Assignment 0: Interface eth1, Prio 63
 - Assignment 1: Interface wlan0.1, Prio 63
 - Assignment 2: Interface eth1, Prio 63
 - Assignment 3: Interface wlan0, Prio 63
- Current Bridging Table**:

Bridge Name	STP enabled	Interfaces
br251	yes	eth1 wlan3
br252	yes	wlan0.1 wlan4
br0	no	wlan0

Port Setup

Port Setup

WAN Port Assignment	vlan1
Network Configuration eth0	<input type="radio"/> Unbridged <input checked="" type="radio"/> Default
Network Configuration eth1	<input type="radio"/> Unbridged <input checked="" type="radio"/> Default
Network Configuration etheripo	<input type="radio"/> Unbridged <input checked="" type="radio"/> Default
Network Configuration vlan0	<input type="radio"/> Unbridged <input checked="" type="radio"/> Default
Network Configuration vlan3	<input checked="" type="radio"/> Unbridged <input type="radio"/> Default
MTU	1500
Multicast forwarding	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Masquerade / NAT	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
IP Address	192 . 168 . 251 . 1
Subnet Mask	255 . 255 . 255 . 0
Network Configuration vlan4	<input checked="" type="radio"/> Unbridged <input type="radio"/> Default
MTU	1500
Multicast forwarding	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Masquerade / NAT	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
IP Address	192 . 168 . 252 . 1
Subnet Mask	255 . 255 . 255 . 0
Network Configuration wl0.1	<input type="radio"/> Unbridged <input checked="" type="radio"/> Default

DHCPD

Multiple DHCP Server

Add

2.4.3.2 WiFi

Wireless Physical Interface wl0

Wireless Mode	AP	
Wireless Network Mode	Mixed	
Wireless Network Name (SSID)	LEGO_UM	
Wireless Channel	Auto	
Wireless SSID Broadcast	<input checked="" type="radio"/> Enable <input type="radio"/> Disable	
Sensitivity Range (ACK Timing)	2000	(Default: 2000 meters)
Network Configuration	<input type="radio"/> Unbridged <input checked="" type="radio"/> Bridged	

Virtual Interfaces

Wireless Network Name (SSID)	LEGO_UPM
Wireless SSID Broadcast	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
AP Isolation	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Network Configuration	<input type="radio"/> Unbridged <input checked="" type="radio"/> Bridged

Wireless Security wl0

Security Mode	WPA2 Enterprise	
WPA Algorithms	TKIP	
Radius Auth Server Address	192.168.251.3	
Radius Auth Server Port	1812	(Default: 1812)
Radius Auth Shared Secret	*****	<input type="checkbox"/> Unmask
Key Renewal Interval (in seconds)	3600	

Virtual Interfaces wl0.1 SSID [LEGO_UPM]

Security Mode	WPA2 Enterprise	
WPA Algorithms	TKIP	
Radius Auth Server Address	192.168.252.3	
Radius Auth Server Port	1812	(Default: 1812)
Radius Auth Shared Secret	*****	<input type="checkbox"/> Unmask
Key Renewal Interval (in seconds)	3600	

Además de la WiFi principal, crearemos otra para que cada organización tenga su punto de acceso con la configuración RADIUS pertinente.

2.4.3.3 DHCP

DNSMasq

DNSMasq Enable Disable

Local DNS Enable Disable

Additional DNSMasq Options

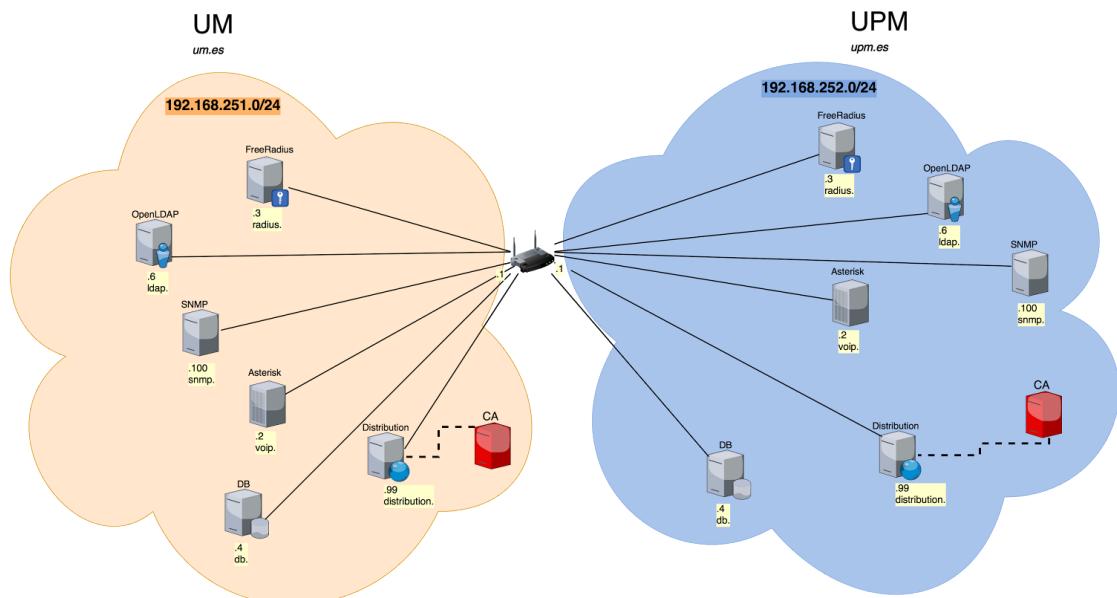
```
# Enables DHCP on br1
interface=br251
dhcp-option=br251,3,192.168.251.1
dhcp-option=br251,6,192.168.251.1
dhcp-option=br251,121,192.168.252.0/24,192.168.251.1
dhcp-range=br251,192.168.251.120,192.168.251.250,255.255.255.0,24h

# Enables DHCP on br1
interface=br252
dhcp-option=br252,3,192.168.252.1
dhcp-option=br252,6,192.168.252.1
dhcp-option=br252,121,192.168.251.0/24,192.168.252.1
dhcp-range=br252,192.168.252.120,192.168.252.250,255.255.255.0,24h
```

Por último, estableceremos manualmente dos servidores DHCP para cada red.

2. Diseño

El escenario planteado en esta práctica es el de dos universidades que se encuentran en el territorio de España: *Universidad de Murcia* y *Universidad Politécnica de Madrid*. La topología final resultante es la que se muestra a continuación:



La Universidad de Murcia y la Universidad Politécnica de Madrid estarán bajo el nombre de dominio **um.es** y **upm.es** respectivamente.

Para la **UM**, se usará un rango de red 192.168.251.0/24 con dominio **um.es**, mientras que para la **UPM** el tramo es 192.168.252.0/24 y dominio **upm.es**.

Para el despliegue virtualizado, se usa un router WRT54G con firmware **DD-WRT** para dar soporte VLAN y desplegar el escenario.



SNMP

Cada Universidad tendrá definido un esquema de usuarios en uno de sus hosts (**192.168.25X.100, snmp.x**):

- **admin**: Usuario con permiso de lectura. Sus consultas utilizan **SHA** (MD5 es más sencillo computacionalmente) con **DES**.

Las consultas van protegidas por motivos de seguridad. Las operaciones de **admin** requieren de contraseña (modo *priv*) para cualquier consulta.

Todos los agentes escucharán en el *IPv4:0.0.0.0:161* excepto el manejador, quien no tendrá habilitado la escucha a la red pública (sólo en local).

Por otro lado, los agentes tendrán habilitados traps al manejador de su universidad, para algunos parámetros genéricos:

1. Que el proceso *snmpd* esté activo y sólo haya 1
2. Debe haber al menos 10% de espacio libre
3. La carga del sistema no puede superar 12 8 5

En caso de no cumplirse alguna de estas condiciones, se enviará un trap al manejador.

x509v3

Cada una de las universidades tendrá su propia infraestructura PKI, que constará de una CA como la siguiente:

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      d8:e0:ec:c4:70:80:f3:c6
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=um
    Validity
      Not Before: Dec 17 20:52:15 2017 GMT
      Not After : May  4 20:52:15 2045 GMT
    Subject: CN=um
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
        Public-Key: (4096 bit)
          Modulus:
            00:e1:ff:4a:e3:1d:30:68:f1:61:02:92:37:ea:d6:
            1d:0f:64:7e:18:b3:3f:7e:68:bd:74:c9:df:ed:64:
            86:1e:c1:a3:5d:d1:be:25:11:c6:4e:dc:60:13:2e:
```

```
9c:64:04:7b:44:17:1f:35:bb:9e:1e:b8:c1:14:c6:  
3a:0f:c9:c3:15:7b:59:51:25:11:91:1a:d2:a2:47:  
16:73:a9:1f:d8:f9:66:23:ee:cc:1c:af:dd:d0:7b:  
59:80:82:e3:45:95:86:44:b8:ce:e1:0c:c8:07:b9:  
60:96:c9:54:b7:68:60:04:ff:87:3f:05:75:e4:f1:  
67:a4:ce:b6:52:3b:3d:c8:df:e0:75:ec:aa:c9:31:  
2c:1b:35:97:05:c3:a2:2d:5d:5a:c2:11:d8:0c:db:  
75:c0:d7:c3:93:a0:e1:0c:cd:3f:81:37:c3:ca:0f:  
bb:4e:3b:01:8c:5e:a0:67:10:d6:5c:57:23:67:d0:  
bd:3b:7a:4d:53:a3:32:1b:50:25:e7:0e:0a:0d:24:  
d5:29:6c:42:5a:ee:b1:5b:3b:14:da:c6:06:61:b3:  
5d:23:6a:36:92:3a:e0:06:c8:e2:87:21:c0:5d:dd:  
23:2a:bb:fe:a5:0e:56:31:35:ba:c3:f2:45:05:4f:  
9b:70:a5:00:ae:5c:53:50:9c:50:ae:f0:26:3b:21:  
07:23:36:b6:8c:04:61:90:13:76:71:09:93:90:82:  
eb:47:32:1d:06:e8:b1:dc:e8:36:20:20:2d:1a:b9:  
e2:d5:d8:9a:ac:ae:2c:7a:2a:e1:6b:c4:8e:81:83:  
9e:3e:20:44:e9:26:84:da:7c:a3:29:24:61:c8:d6:  
09:f4:48:ce:80:f8:7e:88:cb:7c:c2:51:cd:39:e6:  
0c:51:ca:a3:38:d6:65:67:a8:47:b4:6f:51:c4:87:  
63:95:fe:f6:4c:ce:32:09:7b:55:a4:e4:61:be:93:  
5b:23:93:b6:ca:a1:a5:29:b7:18:61:56:03:9e:bf:  
99:06:d9:ea:76:64:91:19:e0:d6:a9:cf:f2:0d:b9:  
69:c5:45:af:f9:e7:89:15:94:9c:d4:fc:32:24:e4:  
bc:c6:29:9a:c2:c8:b3:83:91:8f:de:68:32:78:69:  
79:c3:2b:1e:48:ee:08:a6:2b:0d:c6:a0:19:69:91:  
0b:3a:45:29:79:8c:c0:3a:af:e4:a4:a1:18:0e:c9:  
f5:9e:5b:27:8e:1f:59:a4:99:15:e2:f5:79:ef:8c:  
c2:5e:f5:03:42:34:be:a5:32:77:27:85:40:ac:50:  
0d:58:74:89:fa:a0:04:f5:3c:c1:8c:a9:a0:c5:fc:  
9e:fc:0c:39:63:6f:2b:52:4f:90:cf:71:91:b6:af:  
ae:0b:b1  
Exponent: 65537 (0x10001)  
X509v3 extensions:  
X509v3 Basic Constraints: critical  
CA:TRUE  
X509v3 Key Usage: critical  
Digital Signature, Certificate Sign, CRL Sign  
X509v3 CRL Distribution Points:  
Full Name:  
URI:http://distribution.um.es/um.crl  
  
Full Name:  
URI:http://192.168.251.99/um.crl  
  
Authority Information Access:  
OCSP - URI:http://192.168.251.99:8080  
OCSP - URI:http://distribution.um.es:8080
```

```
Signature Algorithm: sha256WithRSAEncryption
55:ce:86:f6:2f:02:a8:ca:b3:8a:65:a2:ef:45:5d:5c:11:4f:
29:32:c1:ca:ea:9c:41:96:13:c4:39:1c:f4:96:ee:72:cf:7a:
a6:15:4d:e0:d2:58:16:18:fd:c5:a6:78:9f:e5:9d:43:6c:d9:
76:9b:f6:69:50:23:85:26:4c:87:79:ae:d9:fe:48:d2:36:61:
0f:08:3d:7d:f4:c9:7a:6d:5e:1c:58:15:90:0b:45:a2:c9:21:
0c:c8:db:09:1c:f4:3c:fe:31:ea:7a:2a:48:df:60:d0:3d:21:
46:e8:04:86:3e:8f:d4:40:50:21:a5:ee:d9:90:ec:ac:f0:52:
b0:00:9f:a2:43:83:74:24:48:e2:65:6b:de:a5:20:73:52:b1:
6d:92:0e:85:e9:95:af:b5:d8:c4:41:5c:c2:9d:aa:59:e6:4b:
67:10:d8:a5:90:02:b6:5d:14:81:a7:9a:40:4b:d8:d3:c5:f0:
a5:12:f4:5b:60:55:d1:05:bd:9b:9d:5d:d8:bf:38:90:16:e8:
cf:ab:35:b3:f2:73:e8:e6:66:90:d3:56:98:4e:c9:5f:b7:78:
ca:8e:fd:fc:40:0c:5f:36:7e:4a:1c:98:1c:d7:f7:c5:16:ee:
f0:e2:99:e8:5e:9d:7a:e3:44:05:6a:77:18:96:30:05:53:0b:
af:c9:e0:a5:0a:69:74:be:25:44:53:98:a3:57:25:a6:fc:93:
45:d0:a5:51:06:72:87:ef:77:f5:e0:b7:ef:5b:a6:2a:9f:6b:
97:78:22:d4:14:d7:3c:6f:1b:8a:6a:3b:a0:53:56:b3:2b:67:
2f:b8:90:f5:56:21:9c:11:80:72:a4:98:49:f1:09:19:18:47:
cf:a7:af:79:42:dd:1e:71:e7:c2:b7:62:1d:2f:de:c7:6d:c2:
5b:01:1d:80:cc:50:a0:4a:13:02:9b:0d:bf:c7:1e:b8:22:3a:
d2:4f:66:7e:7f:b7:58:14:66:6b:22:0b:3d:ba:a1:11:04:c2:
2a:05:39:48:9e:86:aa:af:db:e0:98:f6:cb:05:dc:0a:17:e9:
8c:d1:36:8b:7a:f6:77:05:09:dc:a9:50:29:f2:88:02:54:d4:
01:8d:1b:e7:85:0b:3f:cd:55:c9:e6:7e:bb:7f:3d:86:06:cf:
a4:98:91:57:70:19:c0:61:f1:87:64:7c:65:97:ad:34:f3:c2:
df:31:68:71:8a:fb:4e:be:cd:6d:69:1d:63:b4:a6:54:21:65:
ec:fe:6f:12:8b:87:06:33:1c:fd:81:81:f8:eb:43:ec:83:b9:
f4:cf:8d:8d:83:6e:59:88:bc:cd:a7:4a:d0:fd:c8:9d:c1:65:
71:33:ab:4f:7a:a1:8c:16
```

Cabe remarcar el campo de la CRLs y OCSPs, presente tanto en el certificado de la CA como el de lo emitidos por él.

El CA tendrá una validez de 10.000 días. Así pues, en cada universidad, el CA emite los siguientes certificados para diversos servicios:

- Owncloud
- Asterisk
- OpenLDAP
- RADIUS

Además, el equipo donde se encuentran las claves privadas del CA está aislado completamente de la red. Este equipo generará periódicamente las CRLs y las enviará a un equipo encargado de publicarlas para todo el mundo.

Podemos ver un ejemplo de esos certificados a continuación:

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      aa:dd:00:f7:a3:40:da:aa
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=um
    Validity
      Not Before: Dec 17 20:52:49 2017 GMT
      Not After : Dec 12 20:52:49 2018 GMT
    Subject: CN=192.168.251.2
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
        Modulus:
          00:b0:bf:7d:ff:09:30:ee:de:08:24:38:ba:5c:b0:
          e3:5c:d0:51:6c:c0:bf:1b:27:5a:ce:0a:c9:63:b3:
          4c:ee:a2:bd:3f:e0:86:c1:73:40:f1:0f:83:62:27:
          fd:df:43:80:80:3a:28:04:8a:c2:87:a2:88:a2:10:
          7a:c7:63:68:2a:f5:55:35:aa:5d:de:04:47:5d:eb:
          cd:04:59:10:a1:3b:47:08:31:e4:6e:d6:99:f6:eb:
          2e:86:c5:8e:55:ad:a1:10:79:9c:bd:89:4a:81:c6:
          ae:7f:5f:bc:f0:d6:14:c3:36:a8:c7:53:fd:c5:4c:
          c9:cd:e4:28:9e:7d:24:4d:90:9c:46:89:3f:76:35:
          62:03:ab:76:25:52:4d:99:24:8e:93:c9:c6:3c:15:
          55:a8:96:ab:df:85:e9:fe:d5:d2:06:49:f4:f9:44:
          75:1a:59:88:d7:24:5e:6e:c7:f1:f9:08:41:30:9a:
          55:b6:dc:b7:1d:e4:25:d0:c4:e4:dd:f7:3c:eb:25:
          5c:c4:58:7d:ff:58:5a:2c:7a:db:b9:70:b7:86:8b:
          0e:f5:6a:8d:dd:15:39:c0:d1:80:e3:9a:c6:f6:56:
          8b:a2:da:d3:88:5b:6b:6f:ea:f7:67:54:97:12:02:
          18:d4:a0:50:92:3e:e6:81:7b:cf:09:4f:33:6b:fd:
          2f:fb
        Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    X509v3 Key Usage:
      Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
      TLS Web Server Authentication
    X509v3 CRL Distribution Points:
      Full Name:
        URI:http://distribution.um.es/um.crl
      Full Name:
        URI:http://192.168.251.99/um.crl
    Authority Information Access:

```

```

OCSP - URI:http://192.168.251.99:8080
OCSP - URI:http://distribution.um.es:8080

X509v3 Subject Alternative Name:
DNS:voip.um.es, DNS:192.168.251.2, IP
Address:192.168.251.2
Signature Algorithm: sha256WithRSAEncryption
4b:3b:3b:d0:80:ac:cc:ae:a5:da:6a:1e:68:c3:20:42:db:43:
3d:b7:21:92:ff:16:3a:86:68:a2:ac:78:98:d2:ab:0f:8e:d3:
e3:c8:88:7e:ce:f0:2f:7b:28:d9:06:bd:7c:9d:f5:2f:cd:a6:
d5:e2:52:6f:de:15:5b:7a:26:d5:72:14:ef:eb:ba:28:3e:97:
3b:eb:ff:37:87:23:8d:f0:4d:ea:fa:5b:6c:26:e6:2f:d8:1a:
18:d4:9c:46:21:a4:d7:bf:59:0a:95:53:76:cc:83:b1:e5:c7:
a5:2e:68:0e:51:50:5b:af:c6:f2:e0:be:50:ac:06:9e:5d:59:
db:9d:98:0c:c1:3f:46:c8:b5:88:e9:89:67:53:16:c6:c6:5c:
3d:dc:25:90:17:ef:fd:30:0b:be:49:e1:86:4e:93:6d:cf:d1:
85:84:27:5f:a6:f6:4e:3a:3e:6d:ed:e5:52:93:f6:2f:06:f8:
5a:20:ff:65:ac:6b:5c:29:38:0d:28:eb:65:a4:0f:7b:2d:74:
c9:80:fd:14:a0:81:63:1b:fd:df:19:83:e3:6f:21:be:50:ef:
38:22:98:9e:39:8f:16:30:72:28:1d:7c:f2:8b:93:74:d4:49:
dc:f5:0e:48:da:df:a7:8a:d3:d6:7e:2c:a3:48:0a:06:25:af:
7a:07:25:8e:aa:1f:4c:e0:3a:3e:fb:60:92:02:5a:42:d0:a3:
7b:5f:e3:5b:25:37:2a:52:cf:e6:9f:16:bc:d8:5d:e1:15:2e:
43:be:7e:fc:04:52:c0:5f:f8:bd:ca:de:c0:31:3b:ae:e3:26:
2c:b8:ee:5b:32:7b:73:e8:f3:3b:fc:70:93:3f:6e:9b:9d:f3:
b6:91:3d:41:96:b6:3d:74:70:61:31:cf:56:f5:e0:8c:73:a5:
15:9f:1c:43:01:6c:6d:79:af:41:46:4d:5e:e0:df:b8:df:27:
81:16:a7:d5:01:7e:bd:e4:05:1b:2f:5f:42:f1:f5:0c:ee:e5:
e2:df:af:46:89:cb:61:56:e3:3c:b2:34:f0:33:c6:b7:93:4b:
2b:25:19:12:dc:6e:8f:e0:b8:06:33:6b:d4:ed:04:28:ed:3f:
32:0c:d1:d6:36:96:96:b4:97:9f:bf:79:37:f7:ec:2a:0e:83:
58:50:b2:3c:f6:43:75:32:3c:9c:89:8e:55:40:51:e6:21:c0:
91:5c:38:08:49:29:42:2f:81:43:42:58:c3:bc:93:77:90:18:
68:bc:87:b6:e4:da:45:6f:ee:8c:ad:98:d5:6b:8c:48:23:8e:
64:75:0e:4d:69:a0:01:2a:d5:f6:5d:87:e8:a1:91:5b:6e:13:
b0:5b:16:9e:9e:dc:a5:1b

```

Estos certificados tienen una validez de un años. Destacar que usamos de CN la IP del servicio (debido a las restricciones de Asterisk). Sin embargo, incluimos el nombre de dominio en el campo **X509v3 Subject Alternative Name**, para mantener la compatibilidad con navegadores y otros servicios.

Además, tendremos disponible un equipo de distribución (**192.168.25X.99**), quien tendrá a su cargo dos servicios para sus organizaciones:

1. Servir las CRLs en el puerto 80
2. Actuar de OCSP Responder en el puerto 8888

De esta forma, cualquier entidad de cada organización puede verificar la validez de un determinado certificado.

Por último, comentar que el certificado del OCSP Responder debe tener atributos particulares (frente a los certificados de cliente y servidor). Concretamente, se trata del atributo **X509v3 Extended Key Usage: OCSP Signing**.

LDAP

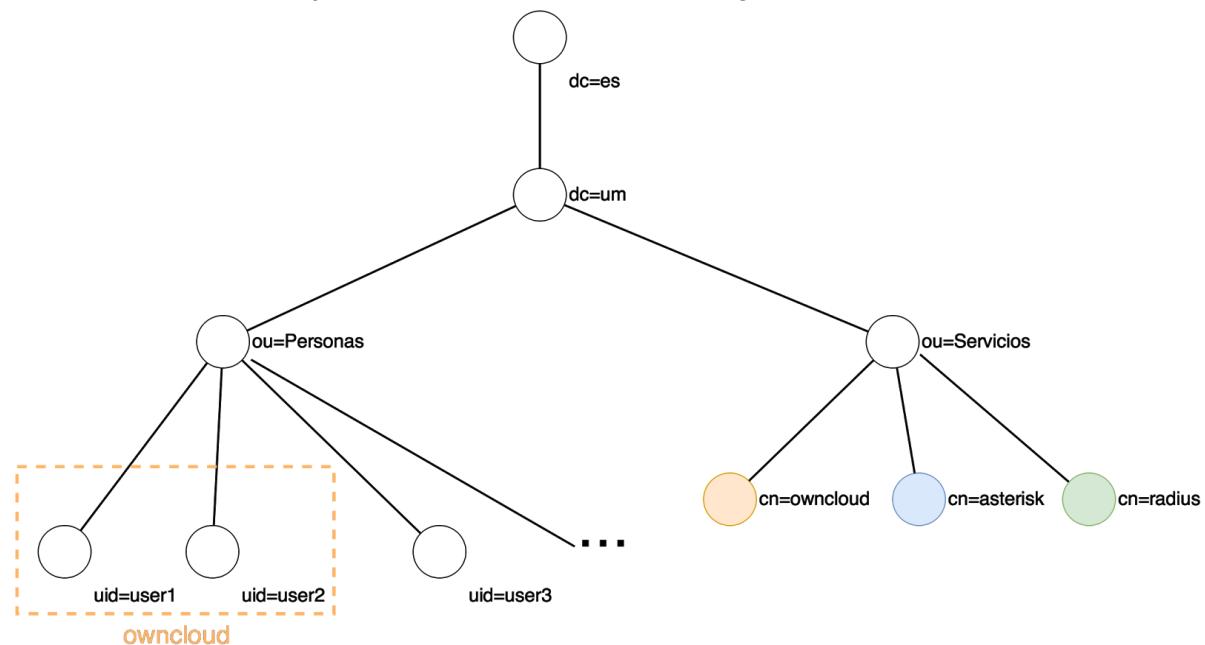
Ambas universidades tendrán un directorio activo (LDAP), ejecutándose bajo el software OpenLDAP.

Este servicio estará disponible para otros servicios como Owncloud o RADIUS para verificar la autenticación de los usuarios que intentan hacer uso de ese servicio. Todas las consultas van protegidas mediante contraseña, por lo que únicamente está disponible para servicios concretos habilitados por el administrador.

La configuración del directorio activo se divide en 3 partes:

1. La raíz y estructura
2. Los usuarios en la base de datos
3. Los grupos (servicios) a los que pertenecen

Podemos ver aquí un ejemplo de la UM, que es homólogo a la UPM:



De esta forma, además de mantener una lista de usuarios, tenemos flexibilidad con los servicios a los que tienen acceso.

Ambas organizaciones usan la entidad organizativa **dc** frente a **o** porque los servicios hacen uso del DNS, siendo esto deseable por compatibilidad de cara al futuro.

RADIUS



Además, ambas universidades dispondrán de un servidor RADIUS para la autenticación. Este servidor será encargado de dar acceso a la red de cada organización por medio inalámbrico.

Se usará el dominio *um.es* y *upm.es* para la Universidad de Murcia y Universidad Politécnica de Madrid respectivamente, teniendo cada usuario un formato como el de a continuación:

- usuario@um.es
- juan@upm.es
- etc.

Los clientes del servicio serán entidades autorizadas para dicho cometido, protegido mediante un secreto compartido. En ellas, habrá los siguientes clientes por cada organización:

1. El router de la organización, quien dará acceso a la red a los usuarios.
2. El RADIUS de la otra organización, para permitir roaming.

La autenticación de los usuarios se realizará a través del protocolo EAP. Por defecto, el servidor RADIUS autentifica a los usuarios a través de **PEAP**, quien se respalda en el directorio activo (LDAP) para verificar la autenticación mediante el usuario y la contraseña.

Por otro lado, además de PEAP, el servidor permite autenticar a cualquier usuario a través de certificados digitales mediante **EAP-TLS**. Para ello, dichos certificados deben cumplir dos condiciones:

1. Haber sido emitidos por el CA de su organización.
2. Utilizar el identificador completo del usuario en el **CN** del certificado. Esto se hace así porque RADIUS no permite verificar dicho campo en las extensiones del certificado

Podemos verificar el correcto funcionamiento de ambos métodos a través de la herramienta **rad_eap_test**:

```
$ rad_eap_test -H 192.168.251.3 -P 1812 -S um_router_password -m WPA-EAP  
-e TLS -u user1@um.es -j um-user-cert.pem -k um-user-key.pem -a um.pem  
access-accept; 0  
$ rad_eap_test -H 192.168.251.3 -P 1812 -S um_router_password -m WPA-EAP  
-e PEAP -2 MSCHAPV2 -u user1@um.es -p password1  
access-accept; 0
```

Al igual que en otros servicios, desde **LDAP** se controla qué usuarios pueden acceder a este servicio mediante la gestión de grupos. Además, desde el mismo directorio LDAP, se utilizan **atributos** para definir cómo será la conexión:

Fragmento del servidor LDAP

```
...  
AstAccountCallerID: 251001  
radiusFramedMTU: 999  
objectClass: inetOrgPerson  
objectClass: AsteriskSIPUser  
objectClass: radiusprofile  
...
```

Asterisk



Para el diseño de la infraestructura de asterisk, dispondremos de dos prefijos en base a la organización. Además, cada organización actuará bajo un contexto definido por sus siglas (um y upm).

Por un lado, el prefijo de **UM** es **251XXX** dónde **X** es un número de 0 a 9. Así pues, el número **251000** es especial, reservado para prueba de conexión con la centralita.

Por otro lado, el prefijo de UPM es **252XXX** dónde **X** es un número de 0 a 9. Así pues, el número **252000** es especial, reservado para prueba de conexión con la centralita.

Ambas organizaciones hacen uso de su servidor LDAP para autenticar a sus usuarios.

Cada una de las universidades tendrá configurada su dialpan de tal modo que fuerza a usar únicamente sus prefijos internos y, como excepción, traspasar la llamada a la otra universidad si se trata de su prefijo y no el suyo.

Respecto a los protocolos, se utilizará señalización SIP mediante **TLS** porque es una cantidad de tramas relativamente pequeña y es preferible asegurarla. Por otro lado, se utiliza el protocolo RTP (**UDP**) para el intercambio de datos de voz, ya que prevalece una latencia baja frente a asegurar transmitir todos los paquetes.

Además, el intercambio de datos de voz se realizará directamente entre los clientes, sin pasar por la centralita. Esto se define así porque:

- En las universidades no es necesario un autómata en las llamadas. La mayor parte de la comunicación se realiza entre terminales y, de forma muy escasa, con un operario de soporte. Por tanto, la detección de pulsaciones es prescindible
- La carga se reduce drásticamente. Las universidades tienen miles de personas que usan frecuentemente las llamadas, por lo que es supondría una sobrecarga considerable.

Owncloud



Cada una de las universidades tendrá un servicio de almacenamiento de archivos basado en la nube.

Este servicio funcionará junto a un directorio activo (*LDAP*) por cuestiones de administración: En caso de tener que realizar alguna modificación a un usuario, no tener que hacerlo para cada uno de los servicios que puedan tener elementos comunes (por ejemplo, la contraseña).

Además, cada usuario tendrá definido por defecto una cuota de disco duro de **1GB**, si no se especifica otra cantidad en el directorio activo.

Por otro lado, el servidor web que brinda el servicio (Apache), usará un certificado x509 emitido por su universidad junto a la lista de revocación de certificados (CRL). De esta forma, la conexión de los clientes se hará de forma segura.

Además, Owncloud delegará toda la gestión de metainformación y estado en una **base de datos** (PostgreSQL). Una base de datos no es obligatoria con este servicio. Sin embargo, cuando decenas de miles de personas hacen uso del servicio, la gestión de la metainformación a nivel de sistema de ficheros no es suficiente y sobrecarga el sistema. Es por ello, que la base de datos se convierte en un elemento crucial para que el servicio sea decente.

3. Servicios

3.1 SNMP

El **Protocolo Simple de Administración de Red** o **SNMP** es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red.

Simple Network Management Protocol (SNMP)	
Familia	Familia de protocolos de Internet
Función	Facilita el intercambio de información de administración entre dispositivos de red
Última versión	SNMPv3
Puertos	161/UDP, 162/UDP (Trap)
Ubicación en la pila de protocolos	
Aplicación	SNMP
Transporte	UDP y TCP
Red	IP (IPv4 y IPv6)
Estándares	
RFC 1157 (SNMP, 1990)	
RFC 3410 (SNMPv3, 2002)	

3.1.1 Definiciones

3.1.1.1 SNMPv1

Cuando se utiliza SNMPv1, el agente emplea un sencillo sistema de autenticación para determinar qué gestor puede acceder a qué variables de la MIB. Este esquema implica la especificación de unas determinadas políticas de acceso, relacionadas con los conceptos de **comunidad**, **modo de acceso** y **vista de la MIB**. Una **comunidad** es un conjunto de

hosts relacionados mediante un nombre de comunidad, que debe ser incluido en la petición. El **modo de acceso** especifica los accesos permitidos para una determinada comunidad, que suelen ser **none**, **read-write**, **read-only** o **write-only**. Una **vista de la MIB** define uno o más subárboles de la MIB a los que una determinada comunidad puede acceder.

Cuando el agente recibe una petición, verifica el nombre de comunidad junto con la IP del host desde el que se hizo la petición para determinar que el host realmente pertenezca a esa comunidad. Si esta verificación devuelve un resultado positivo, entonces comprueba que la comunidad tenga acceso a las variables de la MIB solicitadas en la petición. Si esto es correcto, el agente responderá a la petición. Si no, devolverá el mensaje de error correspondiente al host peticionario.

3.1.1.2 SNMPv2

SNMPv2 es una extensión de SNMPv1. La abstracción de SNMPv2 que permite el uso de los mismos elementos descritos para SNMPv1, es decir, los conceptos de **comunidad**, **vista de la MIB** y **modo de acceso**, es llamada SNMPv2c (*Community-based SNMPv2*), y es la que el agente de NET-SNMP implementa.

El formato de la PDU (mensaje) es el mismo, solo que se usa un nuevo número de versión. En cuanto a las operaciones de acceso a la MIB, además de las definidas en SNMPv1, esta versión define dos nuevas: **GetBulk** e **Inform**. **GetBulk** se usa para obtener de forma eficiente grandes cantidades de datos, e **Inform** se usa para el envío de notificaciones con confirmación por parte del receptor.

3.1.1.3 SNMPv3

SNMPv3 añade capacidades de seguridad y configuración remota a las versiones previas. Se introduce la arquitectura del **modelo de seguridad basado en usuarios** o USM (*User-based Security Model*), para añadir seguridad a los mensajes, y el **modelo de control de acceso basado en vistas** o VACM (*View-based Access Control Model*) para el control de acceso.

También introduce la posibilidad de configurar remotamente el agente dando distintos valores a los objetos que representan la configuración del agente. La arquitectura soporta el uso simultáneo de distintos modelos de control de acceso, seguridad y proceso de mensajes, entre los cuales hay que resaltar la **autenticación de usuarios mediante protocolos seguros** (como MD5 o SHA) y la **privacidad en la comunicación usando algoritmos de encriptación** como DES.

3.1.2 Archivos de configuración

Encontramos dos archivos de configuración principales: **snmp.conf** y **snmpd.conf**. El primero indica al demonio de SNMP que MIBS deben cargarse. Normalmente, nos interesa cargar todos los que tenga disponible el sistema, por lo que es necesario comentar esta línea:

```
sed -i 's/.*mibs :/#/' /etc/snmp/snmp.conf
```

Por otro lado, **snmpd.conf** es el núcleo de SNMP. Esencialmente, tenemos dos modelos: El que usamos en el manejador y el que usamos en los agentes. Mostaremos la información más relevante

Manejador:

```
# Listen for connections from the local system only
agentAddress udp:127.0.0.1:161

rocommunity public default -V systemonly          # Default access to basic system info
rocommunity6 public default -V systemonly         # rocommunity6 is for IPv6

sysLocation    UM_SNPMP
sysContact     Valentín <admin@um.es>           # Application + End-to-End layers
sysServices     72

#
# Process Monitoring
#
proc mountd               # At least one 'mountd' process
proc ntalkd 4              # No more than 4 'ntalkd' processes - 0 is OK
proc sendmail 10 1          # At least one 'sendmail' process, but no more than 10

# Walk the UCD-SNMP-MIB::prTable to see the resulting output
# Note that this table will be empty if there are no "proc" entries in the snmpd.conf file

#
# Disk Monitoring
#
disk   /      10000          # 10MBs required on root disk, 5% free on /var, 10% free on
all other disks
disk   /var   5%
includeAllDisks 10%

#
# System Load
#
load   12 10 5             # Unacceptable 1-, 5-, and 15-minute load averages
```

Del manejador, pocos cambios realizamos respecto a la configuración por defecto. Lo más destacable es que escuchamos únicamente en nuestra dirección local, por lo que ningún otro equipo puede realizar consultas.

Por otro lado, los agentes tienen una estructura tal que así:

```
agentAddress udp:161

createUser admin SHA "@admin_password@" DES "@admin_password@"

#####
#
# ACCESS CONTROL
#
# Acceso completo de sólo lectura
rouser      admin    priv

#####
#
# SYSTEM INFORMATION
#
sysLocation   UM_OWNCloud
sysContact     Valentin <admin@n1.net>
                #
# Process Monitoring
#
# Solo puede haber 1 servicio de SNMP ejecutandose
proc    snmpd  1  1

#
# Disk Monitoring
#
# Debe haber al menos 10% de espacio en todas las particiones del sistema
includeAllDisks  10%

# Walk the UCD-SNMP-MIB::dskTable to see the resulting output
# Note that this table will be empty if there are no "disk" entries in the snmpd.conf
file

#
# System Load
#
# Máximas cargas del sistema para 1-, 5-, 15- minutos
load    12 8 5

# Walk the UCD-SNMP-MIB::laTable to see the resulting output
# Note that this table *will* be populated, even without a "load" entry in the
snmpd.conf file

#####
#
# ACTIVE MONITORING
#
# trap2sink 192.168.251.100 public
# informsink 192.168.251.100 public
# authtrapenable 1
```

```
# Event MIB - automatically generate alerts
#
# Remember to activate the 'createUser' lines above
iquerySecName internalUser
rouser internalUser           localhost
defaultMonitors yes           # generate traps on UCD error conditions
linkUpDownNotifications yes   # generate traps on linkUp/Down
```

Tal como podemos observar, escuchamos en todas las direcciones, por lo que cualquier equipo puede realizar consultas en el puerto UDP. Por otro lado, tenemos definido únicamente un usuario (admin), con el que se podrán realizar consultas.

Además, tenemos activados traps, que irán dirigidas al manejador de nuestra respectiva universidad.

Por último, comentar que el archivo **traps_wrapper** es un simple script que imprime por pantalla los traps que recibe:

```
#!/bin/sh

read host
read ip
vars=

while read oid val
do
  if [ "$vars" = "" ]
  then
    vars="$oid = $val"
  else
    vars="$vars, $oid = $val"
  fi
done

echo trap: $1 $host $ip $vars
```

3.1.3 Despliegue

El despliegue de este servicio, se define así en el **docker-compose** de forma homóloga para cada universidad:

```
um_snmp:
  build: ./um/snmp/
  hostname: snmp
  domainname: um.es
  networks:
    red1:
      aliases:
        - snmp.um.es
      ipv4_address: 192.168.251.100
```

Definimos el directorio donde se encuentra el servicio y la dirección IP, entre otros detalles como el dominio y el hostname.

Por otra parte, el **Dockerfile** es el siguiente:

```
FROM ubuntu:16.04
LABEL maintainer="valiantsin.kivachuk@um.es"

ENV DEBIAN_FRONTEND noninteractive

#Colors
RUN bash -c 'echo -e "Dpkg::Progress-Fancy \"1\";\nAPT::Color \"1\";" > /etc/apt/apt.conf.d/99geekosupremo'

RUN echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
    apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
    apt-get update && apt-get install --no-install-recommends -y apt-fast && apt-fast
install -y \
    net-tools \
    traceroute \
    netcat \
    dnsutils \
    iputils-ping \
    curl \
    nano \
    ifupdown2 \
    bash-completion \
    supervisor \
    locales \
    software-properties-common \
    snmpd snmptrapd snmp snmp-mibs-downloader \
    wget \
    git-core \
    unzip \
    default-mta \
    libssl-dev \
    ca-certificates \
    build-essential \
    libtool \
    freeradius-utils \
&& rm -rf /var/lib/apt/lists/ \
&& rm -rf /var/cache/oracle-jdk8-installer

ENV SNMP_VERSION 5.6.2

RUN wget -P /usr/src/
https://sourceforge.net/projects/net-snmp/files/net-snmp/$SNMP_VERSION/net-snmp-$SNMP_VERSION.tar.gz && tar --directory /usr/src/ -xvf /usr/src/net-snmp-$SNMP_VERSION.tar.gz

# Configure
RUN cd /usr/src/net-snmp-$SNMP_VERSION && \
    ./configure \
        --with-security-modules=tsm \
        --with-transports="DTLSUDP TLSTCP" \
        --with-default-snmp-version=3 \
        --with-sys-contact=admin@um.es \
        --with-sys-location=Murcia \
        --with-logfile=/var/log/snmpd.log \
        --with-persistent-directory=/var/net-snmp \
        --disable-embedded-perl \
```

```

--without-perl-modules \
--disable-shared \
--enable-static \
--with-pic \
&& \
make -j $(((${getconf _NPROCESSORS_ONLN}+1))) && \
make -j $(((${getconf _NPROCESSORS_ONLN}+1))) install && \
\ 
rm -Rf /usr/src/net-snmp*

#Wrapper
COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf
COPY misc/traps_wrapper /usr/bin/traps_wrapper

RUN sed -i 's/.*mibs :/#/' /etc/snmp/snmp.conf #Enable all MIBS

COPY conf/snmpd.conf /etc/snmp/snmpd.conf
COPY conf/snmptrapd.conf /etc/snmp/snmptrapd.conf

ENTRYPOINT ["/usr/bin/supervisord"]

```

Sobre una imagen de Ubuntu 16.04, descargamos las dependencias necesarias para funcionar. A continuación, compilamos SNMP con parámetros personalizados para habilitarle soporte TLS. Por último, preparamos las wrappers, que no son nada más que scripts que se ejecutarán cuando iniciemos el contenedor e inicializarán el servicio.

Supervisord:

```

[supervisord]
nodaemon=true

[program:snmpd]
command=/etc/init.d/snmpd start
priority=100
startretries=0

[program:snmptrapd]
command=snmptrapd -f -Leo
priority=100

```

3.1.4 Trazas NET-SNMP

NET-SNMP es un conjunto de aplicaciones usado para implementar el protocolo SNMP. Esto incluye:

- Aplicaciones de línea de comandos para:
 - tomar información de dispositivos capaces de manejar el protocolo SNMP, ya sea usando peticiones simples (**snmpget**, **snmpgetnext**) o múltiples (**snmpwalk**, **snmphtable**, **snmpdelta**).
 - manipular información sobre la configuración de dispositivos capaces de manejar SNMP (**snmpset**).

- conseguir un conjunto de informaciones de un dispositivo con SNMP (**snmpdf**, **snmpnetstat**, **snmpstatus**).
- traducir entre OIDs numéricos y textuales de los objetos de la MIB, y mostrar el contenido y estructura de la MIB (**snmptranslate**).
- Un navegador gráfico de la MIB (**tkmib**), usando Tk/perl.
- Un demonio para recibir notificaciones SNMP (**snmptrapd**). Las notificaciones seleccionadas pueden guardarse en un log (como **syslog** o un archivo de texto plano), ser reenviadas a otro sistema de gestión de SNMP, o ser pasadas a una aplicación externa.
- Un agente configurable para responder a peticiones SNMP para información de gestión (**snmpd**). Incluye soporte para un amplio rango de módulos de información de la MIB, y puede ser extendido usando módulos cargados dinámicamente, scripts externos y comandos.
- Una biblioteca para el desarrollo de nuevas aplicaciones SNMP, con APIs para C y Perl.

3.1.4.1 SNMPGET

snmpget es una aplicación SNMP que usa la petición SNMP GET para obtener información de una entidad de red. Uno o más identificadores de objeto (OIDs) pueden ser proporcionados como argumentos en la línea de comandos.

Para realizar la prueba, ejecutamos el siguiente comando desde el manejador:

```
# snmpget -v 2c -c sta um_pc1 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: UM_PC1
```

A lo que obtenemos la siguiente captura de Wireshark:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
12	92.1083374...	192.168.251.100	192.168.251.20	SNMP	82	get-request 1.3.6.1.2.1.1.6.0
13	92.1178720...	192.168.251.20	192.168.251.100	SNMP	90	get-response 1.3.6.1.2.1.1.6.0
<pre>+ Frame 12: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0 + Ethernet II, Src: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64), Dst: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14) + Internet Protocol Version 4, Src: 192.168.251.100, Dst: 192.168.251.20 + User Datagram Protocol, Src Port: 48606, Dst Port: 161 - Simple Network Management Protocol version: v2c (1) community: sta data: get-request (0) get-request request-id: 953579668 error-status: noError (0) error-index: 0 variable-bindings: 1 item 1.3.6.1.2.1.1.6.0: Value (Null) Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0) Value (Null)</pre>						

Como podemos observar, **fium_server** realiza una petición a **fium_pc1** en relación al objeto 1.3.6.1.2.1.1.6.0, que corresponde con el objeto:

OID value: 1.3.6.1.2.1.1.6

OID description:

sysLocation OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..255))
ACCESS read-write
STATUS mandatory
DESCRIPTION
 "The physical location of this node (e.g.,
 'telephone closet, 3rd floor')."
 ::= { system 6 }

Por la definición de nuestras políticas de acceso, el agente responde correctamente a la solicitud:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
12	92.1083374...	192.168.251.100	192.168.251.20	SNMP	82	get-request 1.3.6.1.2.1.1.6.0
13	92.1178720...	192.168.251.20	192.168.251.100	SNMP	90	get-response 1.3.6.1.2.1.1.6.0
<pre>+ Frame 13: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0 + Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64) + Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100 + User Datagram Protocol, Src Port: 161, Dst Port: 48606 Simple Network Management Protocol version: v2c (1) community: sta data: get-response (2) get-response request-id: 953579668 error-status: noError (0) error-index: 0 variable-bindings: 1 item 1.3.6.1.2.1.1.6.0: 4649554d5f504331 Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0) Value (OctetString): 4649554d5f504331</pre>						

El campo **Value** corresponde a la respuesta del agente, que es codificada en Octal. Por tanto, **554d5f504331** en hexadecimal equivale a **UM_PC1**.

3.1.4.2 SNMPWALK

snmpwalk es una aplicación SNMP que utiliza solicitudes SNMP GETNEXT para consultar un árbol de información a una entidad de red.

Un identificador de objeto (OID) se puede proporcionar en la línea de comandos. Este OID especifica qué parte del espacio del identificador de objeto se buscará utilizando las peticiones GETNEXT. Todas las variables en el subárbol del OID se consultan y sus valores se presentan al usuario.

Si no hay ningún argumento OID, snmpwalk buscará el subárbol SNMPv2-SMI::mib-2. Si la entidad de red tiene un error al procesar el paquete de solicitud, se devolverá un paquete de

error y se mostrará un mensaje, lo que ayudará a determinar por qué la solicitud se ha malformado.

Si la búsqueda de árbol provoca intentos de búsqueda más allá del final de la MIB, se mostrará el mensaje "End of MIB".

Realizamos la operación con:

```
# snmpwalk -v 2c -c sta um_pc1 system
SNMPv2-MIB::sysDescr.0 = STRING: Linux fium_pc1 4.9.51-1-MANJARO #1 SMP
PREEMPT Wed Sep 20 10:37:40 UTC 2017 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (196645) 0:32:46.45
SNMPv2-MIB::sysContact.0 = STRING: Valentin <admin@n1.net>
SNMPv2-MIB::sysName.0 = STRING: um_pc1
SNMPv2-MIB::sysLocation.0 = STRING: UM_PC1
SNMPv2-MIB::sysServices.0 = INTEGER: 72
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (5) 0:00:00.05
SNMPv2-MIB::sysORID.1 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.2 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORID.3 = OID:
SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.4 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.5 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
SNMPv2-MIB::sysORID.6 = OID: TCP-MIB::tcpMIB
...
...
```

Y observamos la siguiente traza:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	80	get-next-request 1.3.6.1.2.1.1.1
2	0.017338203	192.168.251.20	192.168.251.100	SNMP	164	get-response 1.3.6.1.2.1.1.1.0
3	0.017460452	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.1.0
4	0.017570661	192.168.251.20	192.168.251.100	SNMP	92	get-response 1.3.6.1.2.1.1.2.0
5	0.017642515	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.2.0
6	0.018421644	192.168.251.20	192.168.251.100	SNMP	85	get-response 1.3.6.1.2.1.1.3.0
7	0.018473395	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.3.0

+ Frame 1: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64), Dst: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14)
+ Internet Protocol Version 4, Src: 192.168.251.100, Dst: 192.168.251.20
+ User Datagram Protocol, Src Port: 39519, Dst Port: 161
Simple Network Management Protocol
version: v2c (1)
community: sta
data: get-next-request (1)
get-next-request
request-id: 1353012650
error-status: noError (0)
error-index: 0
variable-bindings: 1 item
1.3.6.1.2.1.1: Value (Null)
Object Name: 1.3.6.1.2.1.1 (iso.3.6.1.2.1.1)
Value (Null)

En esta captura, podemos ver el procedimiento de snmpwalk. El procedimiento empieza solicitando un primer objeto **1.3.6.1.2.1.1** (system), a lo que el cliente responde con el objeto inmediatamente posterior:

snmp							
No.	Time	Source	Destination	Protocol	Length	Info	
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	80	get-next-request 1.3.6.1.2.1.1	
2	0.017338203	192.168.251.20	192.168.251.100	SNMP	164	get-response 1.3.6.1.2.1.1.1.0	
3	0.017460452	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.1.0	
4	0.017570661	192.168.251.20	192.168.251.100	SNMP	92	get-response 1.3.6.1.2.1.1.2.0	
5	0.017642515	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.2.0	
6	0.018421644	192.168.251.20	192.168.251.100	SNMP	85	get-response 1.3.6.1.2.1.1.3.0	
7	0.018473395	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.3.0	

+ Frame 2: 164 bytes on wire (1312 bits), 164 bytes captured (1312 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64)
+ Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100
+ User Datagram Protocol, Src Port: 161, Dst Port: 39519
Simple Network Management Protocol
version: v2c (1)
community: sta
data: get-response (2)
get-response
request-id: 1353012650
error-status: noError (0)
error-index: 0
variable-bindings: 1 item
1.3.6.1.2.1.1.0: 4c696e7578206669756d5f70633120342e392e35312d312d...
Object Name: 1.3.6.1.2.1.1.0 (iso.3.6.1.2.1.1.0)
Value (OctetString): 4c696e7578206669756d5f70633120342e392e35312d312d...
Variable-binding-string: Linux fium_pc1 4.9.51-1-MANJARO #1 SMP PREEMPT Wed Sep 20 10:37:40 UTC 2017 x86_64

Es decir, con **1.3.6.1.2.1.1.1.0** (system.sysDescr). Esto se realiza para todo el árbol de system hasta recibir un mensaje de finalización.

3.1.4.3 SNMPBULKWALK

El funcionamiento de **snmpbulkwalk** es exactamente igual a [snmpwalk](#) con una crucial diferencia: Optimiza las llamadas realizadas al agente para no saturar la red. Esto se consigue haciendo uso de los mensajes SNMP GETBULK.

Para mostrarlo, utilizaremos la misma orden que en *snmpwalk*:

```
# snmpbulkwalk -v 2c -c sta um_pc1 system
SNMPv2-MIB::sysDescr.0 = STRING: Linux fium_pc1 4.9.51-1-MANJARO #1 SMP
PREEMPT Wed Sep 20 10:37:40 UTC 2017 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (196645) 0:32:46.45
SNMPv2-MIB::sysContact.0 = STRING: Valentin <admin@n1.net>
SNMPv2-MIB::sysName.0 = STRING: um_pc1
SNMPv2-MIB::sysLocation.0 = STRING: UM_PC1
SNMPv2-MIB::sysServices.0 = INTEGER: 72
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (5) 0:00:00.05
SNMPv2-MIB::sysORID.1 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.2 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORID.3 = OID:
SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.4 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.5 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
SNMPv2-MIB::sysORID.6 = OID: TCP-MIB::tcpMIB
```

...

Y la traza del comando:

No.	Time	Source	Destination	Protocol	Length	Info
3	12.1406145...	192.168.251.100	192.168.251.20	SNMP	80	getBulkRequest 1.3.6.1.2.1.1
4	12.1407928...	192.168.251.20	192.168.251.100	SNMP	372	get-response 1.3.6.1.2.1.1.0 1.3.6.1.2.1.2.0 1.
5	12.1409376...	192.168.251.100	192.168.251.20	SNMP	84	getBulkRequest 1.3.6.1.2.1.1.9.1.2.2
6	12.1410330...	192.168.251.20	192.168.251.100	SNMP	416	get-response 1.3.6.1.2.1.1.9.1.2.3 1.3.6.1.2.1.1.9.
7	12.1411339...	192.168.251.100	192.168.251.20	SNMP	84	getBulkRequest 1.3.6.1.2.1.1.9.1.3.2
8	12.1412255...	192.168.251.20	192.168.251.100	SNMP	606	get-response 1.3.6.1.2.1.1.9.1.3.3 1.3.6.1.2.1.1.9.
9	12.1413117...	192.168.251.100	192.168.251.20	SNMP	84	getBulkRequest 1.3.6.1.2.1.1.9.1.4.2
10	12.1414529...	192.168.251.20	192.168.251.100	SNMP	239	get-response 1.3.6.1.2.1.1.9.1.4.3 1.3.6.1.2.1.1.9.

+ Frame 4: 372 bytes on wire (2976 bits), 372 bytes captured (2976 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64)
+ Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100
+ User Datagram Protocol, Src Port: 161, Dst Port: 41098
Simple Network Management Protocol
version: v2c (1)
community: sta
data: get-response (2)
get-response
request-id: 136528340
error-status: noError (0)
error-index: 0
variable-bindings: 10 items
1.3.6.1.2.1.1.1.0: 4c696e7578206669756d5f70633120342e392e35312d312d...
Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)
Value (OctetString): 4c696e7578206669756d5f70633120342e392e35312d312d...
Variable-binding-string: Linux fium_pc1 4.9.51-1-MANJARO #1 SMP PREEMPT Wed Sep 20 10:37:40 UTC 2017 x86_64
1.3.6.1.2.1.1.2.0: 1.3.6.1.4.1.8072.3.2.10 (iso.3.6.1.4.1.8072.3.2.10)
Object Name: 1.3.6.1.2.1.1.2.0 (iso.3.6.1.2.1.1.2.0)
Value (OID): 1.3.6.1.4.1.8072.3.2.10 (iso.3.6.1.4.1.8072.3.2.10)
1.3.6.1.2.1.1.3.0: 453309
Object Name: 1.3.6.1.2.1.1.3.0 (iso.3.6.1.2.1.1.3.0)
Value (Timeticks): 453309
1.3.6.1.2.1.1.4.0: 56616c656e74696e203c61646d696e406e312e6e65743e
Object Name: 1.3.6.1.2.1.1.4.0 (iso.3.6.1.2.1.1.4.0)
Value (OctetString): 56616c656e74696e203c61646d696e406e312e6e65743e
1.3.6.1.2.1.1.5.0: 6669756d5f706331
Object Name: 1.3.6.1.2.1.1.5.0 (iso.3.6.1.2.1.1.5.0)
Value (OctetString): 6669756d5f706331

Al igual que en snmpwalk, el procedimiento empieza solicitando el objeto 1.3.6.1.2.1.1 (system). Sin embargo, recibimos como respuesta un conjunto de 10 objetos referentes a la consulta.

De esta forma, evitamos saturar la red con peticiones SNMP.

3.1.4.4 SNMPTRANSLATE

snmptranslate es una utilidad que traduce uno o más valores de identificador de objeto SNMP de sus formas textuales a sus formas numéricas (o viceversa). OID es un identificador de objeto numérico o textual.

Por su naturaleza, **snmptranslate** no hace uso de la red porque utiliza los objetos MIBS del propio sistema.

Entre todas las posibilidades que ofrece, destaca la traducción a la forma textual:

```
# snmptranslate 1.3.6.1.2.1.1
```

SNMPv2-MIB::system

y viceversa:

```
# snmptranslate -On SNMPv2-MIB::system  
.1.3.6.1.2.1.1
```

Además, es de gran utilidad ver una representación estilo árbol de un objeto. Lo podemos realizar con la siguiente orden:

```
# snmptranslate -Tp SNMPv2-MIB::system  
---system(1)  
|  
+-- -R-- String      sysDescr(1)  
|      Textual Convention: DisplayString  
|      Size: 0..255  
+-- -R-- ObjID       sysObjectID(2)  
+-- -R-- TimeTicks   sysUpTime(3)  
|  
|  
|  +--sysUpTimeInstance(0)  
  
+-- -RW- String      sysContact(4)  
|      Textual Convention: DisplayString  
|      Size: 0..255  
+-- -RW- String      sysName(5)  
|      Textual Convention: DisplayString  
|      Size: 0..255  
+-- -RW- String      sysLocation(6)  
|      Textual Convention: DisplayString  
|      Size: 0..255  
+-- -R-- INTEGER    sysServices(7)  
|      Range: 0..127  
+-- -R-- TimeTicks  sysORLastChange(8)  
|      Textual Convention:TimeStamp  
  
---sysORTable(9)  
|  
+--sysOREntry(1)  
|  Index: sysORIndex  
  
|  --- - INTEGER  sysORIndex(1)  
|  |  Range: 1..2147483647  
+-- -R-- ObjID     sysORID(2)  
+-- -R-- String    sysORDescr(3)  
|  Textual Convention: DisplayString  
|  Size: 0..255  
+-- -R-- TimeTicks sysORUpTime(4)  
|  Textual Convention:TimeStamp
```

Nota: Para no describir la ruta completa al objeto, podemos hacer uso del parámetro **-IR**, tal que así:

```
# snmptranslate -Tp -IR system
---system(1)
|
+-- -R-- String      sysDescr(1)
|      Textual Convention: DisplayString
|      Size: 0..255
+-- -R-- ObjID       sysObjectID(2)
...
...
```

3.1.4.5 SNMPGETNEXT

Este método es similar a **snmpget**, con la optimización de red descrita en **snmpwalk**. Cuando se consulta un objeto, el resultado que devuelve el agente es el inmediatamente posterior.

Así pues, si consultamos el objeto **system.sysLocation.0**, nos devolverá **system.sysServices.0**, pues este se halla inmediatamente después en el árbol SNMPv2-MIB.

```
# snmpgetnext -v 2c -c sta um_pc1 system.sysLocation.0
SNMPv2-MIB::sysServices.0 = INTEGER: 72
```

Aquí podemos ver la traza de dicho comando:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.6.0
2	0.000145094	192.168.251.20	192.168.251.100	SNMP	83	get-response 1.3.6.1.2.1.1.7.0

- ⊕ Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
- ⊕ Ethernet II, Src: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64), Dst: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14)
- ⊕ Internet Protocol Version 4, Src: 192.168.251.100, Dst: 192.168.251.20
- ⊕ User Datagram Protocol, Src Port: 42262, Dst Port: 161
- ⊖ Simple Network Management Protocol
 - version: v2c (1)
 - community: sta
 - ⊖ data: get-next-request (1)
 - ⊖ get-next-request
 - request-id: 692821185
 - error-status: noError (0)
 - error-index: 0
 - ⊖ variable-bindings: 1 item
 - ⊖ 1.3.6.1.2.1.1.6.0: Value (Null)
 - Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
 - Value (Null)

Solicitud GETNEXT

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	82	get-next-request 1.3.6.1.2.1.1.6.0
2	0.000145094	192.168.251.20	192.168.251.100	SNMP	83	get-response 1.3.6.1.2.1.1.7.0

+ Frame 2: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64)
+ Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100
+ User Datagram Protocol, Src Port: 161, Dst Port: 42262
Simple Network Management Protocol
version: v2c (1)
community: sta
data: get-response (2)
get-response
request-id: 692821185
error-status: noError (0)
error-index: 0
variable-bindings: 1 item
1.3.6.1.2.1.1.7.0: 72
Object Name: 1.3.6.1.2.1.1.7.0 (iso.3.6.1.2.1.1.7.0)
Value (Integer32): 72

Respuesta GETNEXT

Podemos ver claramente que una solicitud a **1.3.6.1.2.1.1.6.0** devuelve su objeto posterior, **1.3.6.1.2.1.1.7.0**

3.1.4.6 SNMPTABLE

Aunque se puede usar [snmpwalk](#) para recuperar el contenido de una tabla, se listarán los resultados de cada columna uno a uno. Esto no es lo que la mayoría de la gente esperaría ver una tabla. Aquí es donde entra snmptable.

```
# snmptable -v 2c -c sta -Os um_pc1 sysORTable
SNMP table: sysORTable
sysORID                      sysORDescr          sysORUpTime
snmpMIB           he Mib module for SNMPv2 entities.
0:0:00:00.82
    ifMIB      generic objects for network interface sub-layers
0:0:00:00.81
    ip         The MIB module for managing IP and ICMP
0:0:00:00.83
    udpMIB    The MIB module for managing UDP implementations
0:0:00:00.82
```

Para conseguirlo, hace uso de las peticiones SNMP GETNEXT o SNMP GETBULK dependiendo de la necesidad:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	192.168.251.100	192.168.251.20	SNMP	83	getBulkRequest 1.3.6.1.2.1.1.9.1.0
2	0.000156217	192.168.251.20	192.168.251.100	SNMP	307	get-response 1.3.6.1.2.1.1.9.1.2.1 1.3.6.1.2.1.1.9.1.2.10
3	0.000242802	192.168.251.100	192.168.251.20	SNMP	84	getBulkRequest 1.3.6.1.2.1.1.9.1.2.10
4	0.000326947	192.168.251.20	192.168.251.100	SNMP	729	get-response 1.3.6.1.2.1.1.9.1.3.1 1.3.6.1.2.1.1.9.1.3.10
5	0.000379751	192.168.251.100	192.168.251.20	SNMP	84	getBulkRequest 1.3.6.1.2.1.1.9.1.3.10
6	0.000455068	192.168.251.20	192.168.251.100	SNMP	241	get-response 1.3.6.1.2.1.1.9.1.4.1 1.3.6.1.2.1.1.9.1.4.10
7	0.000505371	192.168.251.100	192.168.251.20	SNMP	84	getBulkRequest 1.3.6.1.2.1.1.9.1.4.10
8	0.000598555	192.168.251.20	192.168.251.100	SNMP	249	get-response 1.3.6.1.2.1.2.1.0 1.3.6.1.2.1.2.1.1.9.1.4.10

+ Frame 4: 729 bytes on wire (5832 bits), 729 bytes captured (5832 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64)
+ Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100
+ User Datagram Protocol, Src Port: 161, Dst Port: 54663
Simple Network Management Protocol
version: v2c (1)
community: sta
data: get-response (2)
get-response
request-id: 878517625
error-status: noError (0)
error-index: 0
variable-bindings: 10 items
1.3.6.1.2.1.1.9.1.3.1: 546865204d494220666f72204d6573736167652050726f63...
Object Name: 1.3.6.1.2.1.1.9.1.3.1 (iso.3.6.1.2.1.1.9.1.3.1)
Value (OctetString): 546865204d494220666f72204d6573736167652050726f63...
1.3.6.1.2.1.1.9.1.3.2: 546865206d616e6167656d656e7420696e666f726d617469...
Object Name: 1.3.6.1.2.1.1.9.1.3.2 (iso.3.6.1.2.1.1.9.1.3.2)
Value (OctetString): 546865206d616e6167656d656e7420696e666f726d617469...
1.3.6.1.2.1.1.9.1.3.3: 54686520534e4d50204d616e6167656d656e742041726368...
Object Name: 1.3.6.1.2.1.1.9.1.3.3 (iso.3.6.1.2.1.1.9.1.3.3)
Value (OctetString): 54686520534e4d50204d616e6167656d656e742041726368...

Para darle formato a la salida, podemos utilizar algunos parámetros, tales como:

- **-Os** para no describir la ruta completa del objeto
- **-Ci** para agregar un índice a las filas
- **-Cw** para establecer un límite de anchura

Combinando todos los anteriores, podemos conseguir una salida como esta:

```
# snmputable -v 2c -c sta -Os -Ci -Cw 70 um_pc1 sysORTable
SNMP table: sysORTable

index          sysORID
1      snmpMPDCompliance
2      usmMIBCompliance
3 snmpFrameworkMIBCompliance
4          snmpMIB
5      vacmBasicGroup
6          tcpMIB
7          ip
8          udpMIB
9  snmpNotifyFullCompliance
10     notificationLogMIB

SNMP table sysORTable, part 2

index          sysORDescr
1      The MIB for Message Processing and Dispatching.
2 The management information definitions for the SNMP User-based Security Model.
3          The SNMP Management Architecture MIB.
4          The MIB module for SNMPv2 entities
5          View-based Access Control Model for SNMP.
6          The MIB module for managing TCP implementations
7          The MIB module for managing IP and ICMP implementations
8          The MIB module for managing UDP implementations
9          The MIB modules for managing SNMP Notification, plus filtering.
```

```
10
```

The MIB module for logging SNMP Notifications.

```
SNMP table sysORTable, part 3
```

```
index  sysORUpTime
1 0:0:00:00.00
2 0:0:00:00.00
3 0:0:00:00.00
4 0:0:00:00.00
5 0:0:00:00.00
6 0:0:00:00.00
7 0:0:00:00.00
8 0:0:00:00.00
9 0:0:00:00.00
10 0:0:00:00.00
```

A diferencia de otros comandos, `snmptable` **únicamente** puede trabajar con objetos de tablas:

```
# snmptable -v 2c -c sta um_pc1 system.sysLocation
Was that a table? SNMPv2-MIB::sysLocation
# snmptable -v 2c -c sta um_pc1 system.sysLocation.0
Was that a table? SNMPv2-MIB::sysLocation.0
# snmptable -v 2c -c sta um_pc1 1.3.6.1.2.1.1.6.0
Was that a table? SNMPv2-MIB::sysLocation.0
# snmptable -v 2c -c sta um_pc1 system
Was that a table? SNMPv2-MIB::system
```

3.1.4.7 SNMPBULKGET

`snmpbulkget`, al igual que [snmpbulkwalk](#), se apoya en mensajes SNMP GETBULK para optimizar los recursos y no inundar la red con mensajes. Por defecto, cuando pedimos un objeto, recibimos 10 objetos siguiendo la misma mecánica que con SNMP GETNEXT.

Así pues, solicitar el objeto `system` nos devuelve los 10 objetos inmediatamente después en el árbol:

```
# snmpbulkget -v 2c -c sta um_pc1 system
SNMPv2-MIB::sysDescr.0 = STRING: Linux um_pc1 4.9.51-1-MANJARO #1 SMP
PREEMPT Wed Sep 20 10:37:40 UTC 2017 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (765793) 2:07:37.93
SNMPv2-MIB::sysContact.0 = STRING: Valentin <admin@n1.net>
SNMPv2-MIB::sysName.0 = STRING: um_pc1
SNMPv2-MIB::sysLocation.0 = STRING: UM_PC1
SNMPv2-MIB::sysServices.0 = INTEGER: 72
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORID.1 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.2 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
```

Y su respectiva traza:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	80	getBulkRequest 1.3.6.1.2.1.1
2	0.000179891	192.168.251.20	192.168.251.100	SNMP	366	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0 1.

+ Frame 2: 366 bytes on wire (2928 bits), 366 bytes captured (2928 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64)
+ Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100
+ User Datagram Protocol, Src Port: 161, Dst Port: 57084
Simple Network Management Protocol
version: v2c (1)
community: sta
data: get-response (2)
get-response
request-id: 1722188629
error-status: noError (0)
error-index: 0
variable-bindings: 10 items
1.3.6.1.2.1.1.1.0: 4c696e757820756d5f70633120342e392e35312d312d4d41...
Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)
Value (OctetString): 4c696e757820756d5f70633120342e392e35312d312d4d41...
1.3.6.1.2.1.1.2.0: 1.3.6.1.4.1.8072.3.2.10 (iso.3.6.1.4.1.8072.3.2.10)
Object Name: 1.3.6.1.2.1.1.2.0 (iso.3.6.1.2.1.1.2.0)
Value (OID): 1.3.6.1.4.1.8072.3.2.10 (iso.3.6.1.4.1.8072.3.2.10)
1.3.6.1.2.1.1.3.0: 785401
Object Name: 1.3.6.1.2.1.1.3.0 (iso.3.6.1.2.1.1.3.0)
Value (Timeticks): 785401
1.3.6.1.2.1.1.4.0: 56616c656e74696e203c61646d696e406e312e6e65743e
Object Name: 1.3.6.1.2.1.1.4.0 (iso.3.6.1.2.1.1.4.0)
Value (OctetString): 56616c656e74696e203c61646d696e406e312e6e65743e
1.3.6.1.2.1.1.5.0: 756d5f706331
Object Name: 1.3.6.1.2.1.1.5.0 (iso.3.6.1.2.1.1.5.0)
Value (OctetString): 756d5f706331
1.3.6.1.2.1.1.6.0: 554d5f504331
Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
Value (OctetString): 554d5f504331

Este comportamiento se puede modificar con el parámetro **-Cr<N>** donde **<N>** es el número de objetos inmediatamente posteriores.

```
# snmpbulkget -v 2c -c sta -Cr3 um_pc1 system
SNMPv2-MIB::sysDescr.0 = STRING: Linux um_pc1 4.9.51-1-MANJARO #1 SMP
PREEMPT Wed Sep 20 10:37:40 UTC 2017 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1277856) 3:32:58.56
```

3.1.4.8 SNMPTRAP

snmptrap es una herramienta que nos permite generar mensajes trap desde un agente a un manejador. Para generar un mensaje trap, ejecutamos el siguiente comando:

```
# snmptrap -v 2c -c sta um_server ""
NET-SNMP-EXAMPLES-MIB::netSnmpExampleHeartbeatNotification
```

En este caso, enviamos un trap de prueba al servidor desde cualquier agente. Podemos comprobar que fue satisfactorio en la siguiente traza:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.20	192.168.251.100	SNMP	112	snmpV2-trap 1.3.6.1.2.1.1.3.0 1.3.6.1.6.3.1
<pre>+ Frame 1: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface 0 + Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64) + Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100 + User Datagram Protocol, Src Port: 39073, Dst Port: 162 Simple Network Management Protocol version: v2c (1) community: sta data: snmpV2-trap (7) snmpV2-trap request-id: 981536283 error-status: noError (0) error-index: 0 variable-bindings: 2 items 1.3.6.1.2.1.1.3.0: 1773404 Object Name: 1.3.6.1.2.1.1.3.0 (iso.3.6.1.2.1.1.3.0) Value (Timeticks): 1773404 1.3.6.1.6.3.1.1.4.1.0: 1.3.6.1.4.1.8072.2.3.0.1 (iso.3.6.1.4.1.8072.2.3.0.1) Object Name: 1.3.6.1.6.3.1.1.4.1.0 (iso.3.6.1.6.3.1.1.4.1.0) Value (OID): 1.3.6.1.4.1.8072.2.3.0.1 (iso.3.6.1.4.1.8072.2.3.0.1)</pre>						

Como observamos, el manejador no responde con ningún mensaje (ni hay ICMP de vuelta), por lo que el mensaje se ha enviado correctamente.

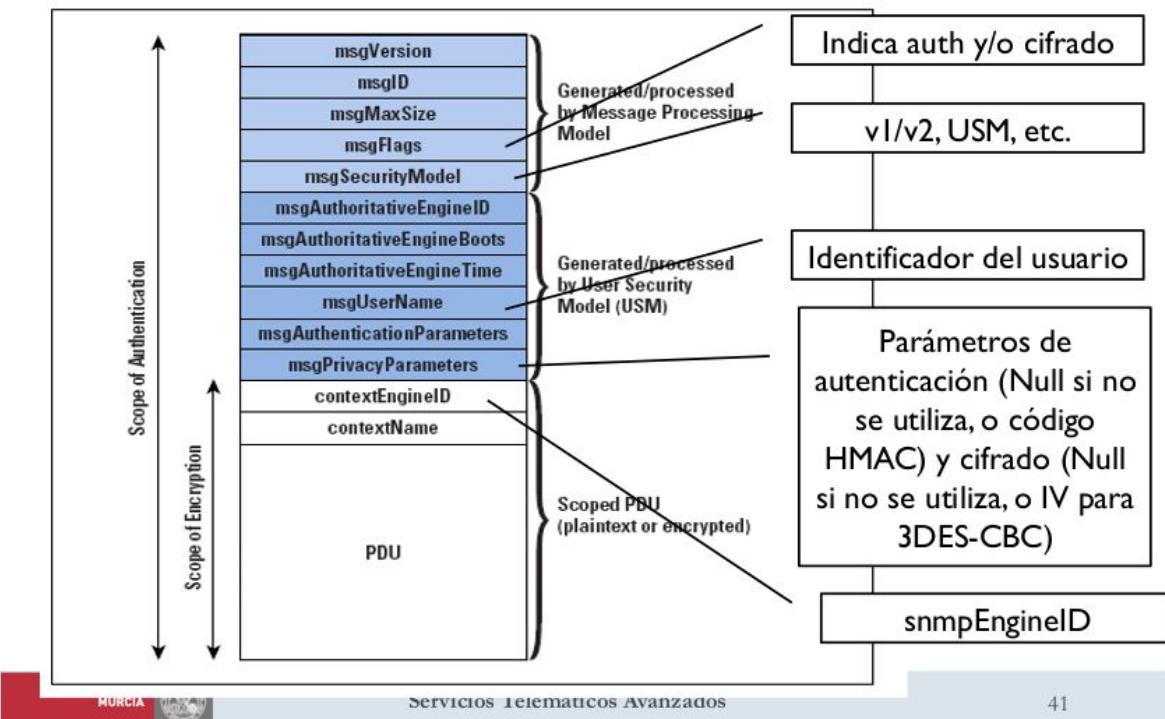
Además, podemos comprobar que hemos recibido adecuadamente el mensaje en los logs de snmptrapd:

```
...
2017-10-03 13:04:10 lego_um_pc1_1.lego_red1 [UDP: [192.168.251.20]:41554->[192.168.251.100]:162]:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1812359) 5:02:03.59      SNMPv2-MIB::snmpTrapOID.0 =
OID: NET-SNMP-EXAMPLES-MIB::netSnmpExampleHeartbeatNotification
```

3.1.4.9 SNMPv3

Una de las principales características de SNMPv3 es la introducción de sistema de seguridad.

SNMPv3: User Security Model



Así, podemos utilizar todos los comandos anteriores con una capa de autentificación.

3.1.4.9.1 SNMPv3 Cleartext

La primera manera de utilizar el protocolo 3 es sin autenticación ni cifrado:

```
root@um_server:/# snmpget -v 3 -c "sta" -u NoAuthUser um_pc1
system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: UM_PC1
```

Y su traza representativa:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	106	get-request
2	0.000156566	192.168.251.20	192.168.251.100	SNMP	158	report 1.3.6.1.6.3.15.1.1.4.0
3	0.000209098	192.168.251.100	192.168.251.20	SNMP	165	get-request 1.3.6.1.2.1.1.6.0
4	0.000302686	192.168.251.20	192.168.251.100	SNMP	174	get-response 1.3.6.1.2.1.1.6.0

+ Frame 3: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64), Dst: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14)
+ Internet Protocol Version 4, Src: 192.168.251.100, Dst: 192.168.251.20
+ User Datagram Protocol, Src Port: 50810, Dst Port: 161
Simple Network Management Protocol
msgVersion: snmpv3 (3)
+ msgGlobalData
+ msgAuthoritativeEngineID: 80001f88803d8c115bdf7ae45900000000
msgAuthoritativeEngineBoots: 1
msgAuthoritativeEngineTime: 2377
msgUserName: NoAuthUser
msgAuthenticationParameters: <MISSING>
msgPrivacyParameters: <MISSING>
- msgData: plaintext (0)
- plaintext
+ contextEngineID: 80001f88803d8c115bdf7ae45900000000
contextName:
- data: get-request (0)
- get-request
request-id: 1113384490
error-status: noError (0)
error-index: 0
- variable-bindings: 1 item
- 1.3.6.1.2.1.1.6.0: Value (Null)
Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
Value (Null)

Las primeras dos trazas son XXXXXXXXXXXXXXXXXXXXXXXX. A continuación, tenemos la petición que hemos realizado al objeto concreto.

3.1.4.9.2 SNMPv3 Auth

Otra opción es utilizar un mecanismo de autentificación para realizar las consultas:

```
root@um_server:/# snmpget -v 3 -c "sta" -u MD5User -l authNoPriv -a MD5
-A pass-md5 um_pc1 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: UM_PC1
```

En este caso, indicamos que utilizamos el usuario **MD5User** con contraseña **pass-md5**:

snmp						
No.	Time	Source	Destination	Protocol	Length	Info
3	5.323310195	192.168.251.100	192.168.251.20	SNMP	106	get-request
4	5.323427103	192.168.251.20	192.168.251.100	SNMP	158	report 1.3.6.1.6.3.15.1.1.4.0
5	5.323480639	192.168.251.100	192.168.251.20	SNMP	175	get-request 1.3.6.1.2.1.1.6.0
6	5.323557023	192.168.251.20	192.168.251.100	SNMP	183	get-response 1.3.6.1.2.1.1.6.0


```
+ Frame 5: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64), Dst: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14)
+ Internet Protocol Version 4, Src: 192.168.251.100, Dst: 192.168.251.20
+ User Datagram Protocol, Src Port: 33616, Dst Port: 161
- Simple Network Management Protocol
  msgVersion: snmpv3 (3)
  + msgGlobalData
  + msgAuthoritativeEngineID: 80001f88803d8c115bdf7ae45900000000
    msgAuthoritativeEngineBoots: 1
    msgAuthoritativeEngineTime: 3444
    msgUserName: MD5User
    msgAuthenticationParameters: 125fc7564788c779be9feeee
    msgPrivacyParameters: <MISSING>
  - msgData: plaintext (0)
    - plaintext
      + contextEngineID: 80001f88803d8c115bdf7ae45900000000
        contextName:
      - data: get-request (0)
        - get-request
          request-id: 720309779
          error-status: noError (0)
          error-index: 0
        - variable-bindings: 1 item
          - 1.3.6.1.2.1.1.6.0: Value (Null)
            Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
            Value (Null)
```

Podemos ver que las cabeceras han cambiado respecto a la traza anterior de SNMPv3, incluyendo los parámetros utilizados.

3.1.4.9.3 SNMPv3 Auth + Crypt

A pesar que el anterior modo evita uso no autorizado, filtra la información que se intercambia a través de la red. Esto se solventa con autentificación + cifrado:

```
root@um_server:/# snmpget -v 3 -c "sta" -u MD5DESUser -l authPriv -a MD5
-A pass-md5 -x DES -X pass-des um_pc1 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: FIUM_PC1
```

A lo que obtenemos:

snmp							
No.	Time	Source	Destination	Protocol	Length	Info	
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	106	get-request	
2	0.000139236	192.168.251.20	192.168.251.100	SNMP	158	report 1.3.6.1.6.3.15.1.1.4.0	
3	0.000209813	192.168.251.100	192.168.251.20	SNMP	191	encryptedPDU: privKey Unknown	
4	0.000303720	192.168.251.20	192.168.251.100	SNMP	199	encryptedPDU: privKey Unknown	

```
+ Frame 3: 191 bytes on wire (1528 bits), 191 bytes captured (1528 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64), Dst: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14)
+ Internet Protocol Version 4, Src: 192.168.251.100, Dst: 192.168.251.20
+ User Datagram Protocol, Src Port: 56362, Dst Port: 161
Simple Network Management Protocol
  msgVersion: snmpv3 (3)
  + msgGlobalData
    + msgAuthoritativeEngineID: 80001f88803d8c115bdf7ae459000000000
      msgAuthoritativeEngineBoots: 1
      msgAuthoritativeEngineTime: 3902
      msgUserName: MD5DESUser
      msgAuthenticationParameters: 1796c8dfcadbbd369acef296
      msgPrivacyParameters: 000000013cba3e4b
    - msgData: encryptedPDU (1)
      encryptedPDU: d8fc02082e8ca593c6231197d037c2fd75ccb6c5b1c3b9dd...
```

Como podemos observar, tenemos visibles las cabeceras del protocolo siendo oculta toda la información que circula por la red.

Si introducimos los datos correspondientes en Wireshark, podemos desencriptar los paquetes y ver su contenido:

snmp							
No.	Time	Source	Destination	Protocol	Length	Info	
1	0.000000000	192.168.251.100	192.168.251.20	SNMP	106	get-request	
2	0.000139236	192.168.251.20	192.168.251.100	SNMP	158	report 1.3.6.1.6.3.15.1.1.4.0	
3	0.000209813	192.168.251.100	192.168.251.20	SNMP	191	get-request 1.3.6.1.2.1.1.6.0	
4	0.000303720	192.168.251.20	192.168.251.100	SNMP	199	get-response 1.3.6.1.2.1.1.6.0	

```
+ Frame 4: 199 bytes on wire (1592 bits), 199 bytes captured (1592 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:14 (02:42:c0:a8:fb:14), Dst: 02:42:c0:a8:fb:64 (02:42:c0:a8:fb:64)
+ Internet Protocol Version 4, Src: 192.168.251.20, Dst: 192.168.251.100
+ User Datagram Protocol, Src Port: 161, Dst Port: 56362
Simple Network Management Protocol
  msgVersion: snmpv3 (3)
  + msgGlobalData
    + msgAuthoritativeEngineID: 80001f88803d8c115bdf7ae459000000000
      msgAuthoritativeEngineBoots: 1
      msgAuthoritativeEngineTime: 3902
      msgUserName: MD5DESUser
    + msgAuthenticationParameters: 53971306cebd217b5ce93a3
      msgPrivacyParameters: 000000016da8937c
    - msgData: encryptedPDU (1)
      encryptedPDU: a54410d08cd5dc69d90fb073e1f5c61f8b592b38079d5ed9...
        + Decrypted ScopedPDU: 303b641180001f88803d8c115bdf7ae459000000000400a2...
          + contextEngineID: 80001f88803d8c115bdf7ae45900000000
            contextName:
          - data: get-response (2)
            + get-response
              request-id: 539714370
              error-status: noError (0)
              error-index: 0
            - variable-bindings: 1 item
              + 1.3.6.1.2.1.1.6.0: 4649554d5f504331
                Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
                Value (OctetString): 4649554d5f504331
```

Tal como se ha indicado en la línea de comandos, se ha solicitado el objeto system.sysLocation.0

3.2 Asterisk

Asterisk es un programa de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (**PBX**). Como cualquier **PBX**, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí dentro de una misma organización e incluso acceder a comunicaciones fuera de la misma a la **PSTN** o conectando a un proveedor de VoIP.

Asterisk

www.asterisk.org



Asterisk™

Información general

Desarrolladores	Digium y comunidad
Última versión estable	13.9.1 13 de mayo de 016
Género	PBX
Licencia	GPL
Sistema Operativo	Multiplataforma

3.2.2 Archivos de configuración

Para la configuración del servicio VoIP en cada una de las organizaciones, tenemos 3 archivos principales que tenemos que modificar para adaptar el servicio a nuestras necesidades:

1. **sip.conf**. Núcleo de Asterisk. Se define las partes más críticas relacionadas con el servicio y cómo se comporta
2. **users.conf**. Base de usuarios. Se establecen qué usuarios hay en el sistema y sus propiedades.

-
- 3. extensions.conf.** Configuración del dialpan. Se establecen las relaciones de llamadas entre los usuarios y cómo debe actuar el sistema ante peticiones.

Así pues, nos encontramos con:

sip.conf:

```
[general]
context=um ; Default context for incoming calls.

udpbindaddr=0.0.0.0 ; IP address to bind UDP listen socket

tcpenable=yes ; Enable server for incoming TCP
connections (default is no)
tcpbindaddr=0.0.0.0 ; IP address for TCP server to bind to
(0.0.0.0 binds to all interfaces)
; Optionally add a port number,
192.168.1.1:5062 (default is port 5060)

tlsenable=yes ; Enable server for incoming TLS (secure)
tlsbindaddr=0.0.0.0 ; IP address for TLS server to bind to
(0.0.0.0)

transport=tls,udp,tcp

disallow=all
allow=g722
allow=alaw

tlscertfile=/certs/voip-cert.pem
tlsprivatekey=/certs/voip-key.pem
tlscapath=/ca_ssl/

directmedia=yes
```

Destacamos que escuchamos en 3 protocolos (TCP, UDP y TLS), siendo TLS el que tiene preferencia. Por otro lado, permitimos únicamente 2 códecs (g722 y alaw), y definimos que la comunicación de voz entre los clientes debe ser directa, sin pasar por la centralita

users.conf:

```
[Asterisk-UPM]
type=friend
host=192.168.252.2
context=um
nat=no
insecure=invite
hassip = yes
```

```
[plantilla-um](!)
type=friend
host=dynamic
context=um
nat=no
hassip = yes
```

Definimos el peer de la otra centralita, y una plantilla para nuestros usuarios.

extensions.conf

```
[um]
exten = 251000,1,Answer()
    same = n,Wait(1)
    same = n,Playback(hello-world)
    same = n,Hangup()
exten => _251XXX,1,Dial(SIP/${EXTEN}) ;Llamadas internas

exten => _252XXX,1,Dial(SIP/Asterisk-UPM/${EXTEN}) ;Llamadas a otra organización
```

Establecemos que únicamente pueden haber llamadas a prefijos **251XXX** y **252XXX**, siendo estos últimos trasladados al Asterisk de la otra organización.

La descripción anterior es para el servicio VoIP de la *UM*. En el caso de la *UPM*, es homólogo a esta.

3.2.X Despliegue

El despliegue de este servicio, se define así en el docker-compose de forma homóloga para cada universidad:

```
um_voip: #VoIP
  build: ./um/voip/
  hostname: voip
  domainname: um.es
  depends_on:
    - "um_snmp"
  volumes:
    - um-certs-voip:/certs
    - lego_ca:/ca_ssl
  networks:
    red1:
      aliases:
        - voip.um.es
  ipv4_address: 192.168.251.2
```

Definimos el directorio donde se encuentra el servicio y la dirección IP, entre otros detalles como el dominio y el hostname.

Además, este servicio se ejecuta después del servidor SNMP y tiene dos volúmenes: El certificado con la clave privada del servicio (TLS) y los CAs públicos de ambas organizaciones.

Por otra parte, el Dockerfile es el siguiente:

```
FROM ubuntu:16.04

ENV DEBIAN_FRONTEND noninteractive

RUN echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
        apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
            apt-get update && apt-get --no-install-recommends -y install apt-fast && \
                apt-fast install -y build-essential git-core pkg-config subversion
libjansson-dev sqlite autoconf \
    automake libtool libxml2-dev libncurses5-dev unixodbc unixodbc-dev
libasound2-dev libogg-dev \
    libvorbis-dev libneon27-dev libspandsp-dev uuid uuid-dev sqlite3
libsdl2-dev aria2 \
    libgnutls-dev ca-certificates wget \
    net-tools \
    supervisor \
    snmpd snmp snmp-mibs-downloader \
    nano \
&& rm -rf /var/lib/apt/lists/

ENV ASTERISK_VERSION 14.6.2

RUN aria2c -x5 -s5 -d /usr/src/
https://github.com/asterisk/asterisk/archive/$ASTERISK_VERSION.tar.gz && tar --directory /usr/src/ -xvf /usr/src/asterisk-$ASTERISK_VERSION.tar.gz

# Configure
RUN cd /usr/src/asterisk-$ASTERISK_VERSION && \
    ./configure && \
    make -j $((($getconf _NPROCESSORS_ONLN)+1)) menuselect.makeopts && \
    menuselect \
    --disable BUILD_NATIVE \
    --enable cdr_csv \
    --enable res_snmp \
    --enable res_http_websocket \
    --enable res_hep_pjsip \
    --enable res_hep_rtcp \
    --enable res_sorcery_astdb \
    --enable res_sorcery_config \
    --enable res_sorcery_memory \
    --enable res_sorcery_memory_cache \
    --enable res_pjproject \
    --enable res_rtp_asterisk \
    --enable res_ari \
    --enable res_ari_applications \
    --enable res_ari_asterisk \
    --enable res_ari_bridges \
    --enable res_ari_channels \
    --enable res_ari_device_states \
    --enable res_ari_endpoints \
    --enable res_ari_events \
    --enable res_ari_mailboxes \
```

```

--enable res_ari_model \
--enable res_ari_playbacks \
--enable res_ari_recordings \
--enable res_ari_sounds \
--enable res_pjsip \
--enable res_pjsip_acl \
--enable res_pjsip_authenticator_digest \
--enable res_pjsip_caller_id \
--enable res_pjsip_config_wizard \
--enable res_pjsip_dialog_info_body_generator \
--enable res_pjsip_diversion \
--enable res_pjsip_dlg_options \
--enable res_pjsip_dtmf_info \
--enable res_pjsip_empty_info \
--enable res_pjsip_endpoint_identifier_anonymous \
--enable res_pjsip_endpoint_identifier_ip \
--enable res_pjsip_endpoint_identifier_user \
--enable res_pjsip_exten_state \
--enable res_pjsip_header_funcs \
--enable res_pjsip_logger \
--enable res_pjsip.messaging \
--enable res_pjsip_mwi \
--enable res_pjsip_mwi_body_generator \
--enable res_pjsip_nat \
--enable res_pjsip_notify \
--enable res_pjsip_one_touch_record_info \
--enable res_pjsip_outbound_authenticator_digest \
--enable res_pjsip_outbound_publish \
--enable res_pjsip_outbound_registration \
--enable res_pjsip_path \
--enable res_pjsip_pidf_body_generator \
--enable res_pjsip_publish_asterisk \
--enable res_pjsip_pubsub \
--enable res_pjsip_refer \
--enable res_pjsip_registrar \
--enable res_pjsip_registrar_expire \
--enable res_pjsip_rfc3326 \
--enable res_pjsip_sdp_rtp \
--enable res_pjsip_send_to_voicemail \
--enable res_pjsip_session \
--enable res_pjsip_sips_contact \
--enable res_pjsip_t38 \
--enable res_pjsip_transport_management \
--enable res_pjsip_transport_websocket \
--enable res_pjsip_xpidf_body_generator \
--enable res_stasis \
--enable res_stasis_answer \
--enable res_stasis_device_state \
--enable res_stasis_mailbox \
--enable res_stasis_playback \
--enable res_stasis_recording \
--enable res_stasis_snoop \
--enable res_stasis_test \
--enable res_statsd \
--enable res_timing_timerfd \
--enable res_config_ldap \
menuselect.makeopts && \
\
make -j $(( ${getconf _NPROCESSORS_ONLN} + 1 )) && \
make -j $(( ${getconf _NPROCESSORS_ONLN} + 1 )) install && \
make -j $(( ${getconf _NPROCESSORS_ONLN} + 1 )) samples && \
make -j $(( ${getconf _NPROCESSORS_ONLN} + 1 )) config && \
\
sed -i -e 's/# MAXFILES=/MAXFILES=/' /usr/sbin/safe_asterisk && \
rm -Rf /usr/src/asterisk*

```

```

# Copy in default configs
COPY conf/sip.conf /etc/asterisk/sip.conf
COPY conf/users.conf /etc/asterisk/users.conf
COPY conf/extensions.conf /etc/asterisk/extensions.conf

#Wrapper
COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf
#SNMP
COPY conf/snmpd.conf /etc/snmp/snmpd.conf

RUN sed -i 's/.*mibs :/#&/' /etc/snmp/snmp.conf #Enable all MIBS
CMD ["/usr/bin/supervisord"]

```

Al igual que en el apartado anterior, hay 3 secciones principales: Descargar las dependencias, Compilar Asterisk y trasladar todos los archivos de configuración al contenedor.

Por último, la configuración de Supervisord:

```

[supervisord]
nodaemon=true

[program:snmpd]
command=/etc/init.d/snmpd start
priority=100
startretries=0

[program:asterisk]
command=asterisk -f -vvvvvvv
priority=100

```

3.2.3 Trazas Asterisk

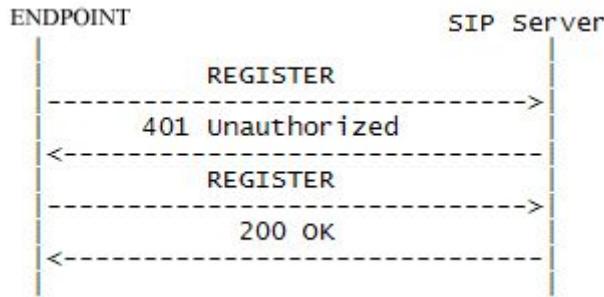
Las trazas relacionados con Asterisk se centran especialmente en el protocolo SIP. El protocolo de inicio de sesión (en inglés: **Session Initiation Protocol** o **SIP**) es un protocolo desarrollado por el grupo de trabajo MMUSIC del IETF con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el *video*, *voz*, *mensajería instantánea*, *juegos en línea* y *realidad virtual*.

Para realizar las trazas, utilizaremos como medio de transporte TCP.

3.2.3.1 SIP REGISTER

Cuando un usuario inicializa su terminal, el agente de usuario SIP que reside en dicho terminal envía una petición con el método REGISTER a un Servidor de Registro, informando a qué dirección física debe asociarse la dirección lógica del usuario.

Podemos ver este proceso cuando conectamos nuestro cliente VoIP a la centralita Asterisk. El intercambio de mensajes SIP que se produce es el siguiente:



Tras poner Wireshark en marcha, podemos ver claramente las cuatro tramas:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.020178061	192.168.252.1	192.168.252.2	SIP	478	Request: REGISTER sip:192.168.252.2 (1 binding)
6	0.029414722	192.168.252.2	192.168.252.1	SIP	590	Status: 401 Unauthorized
8	0.060733732	192.168.252.1	192.168.252.2	SIP	640	Request: REGISTER sip:192.168.252.2 (1 binding)
9	0.063130973	192.168.252.2	192.168.252.1	SIP	610	Status: 200 OK (1 binding)

Frame 4: 478 bytes on wire (3824 bits), 478 bytes captured (3824 bits) on interface 0
 Ethernet II, Src: 02:42:f6:f6:8a:d0 (02:42:f6:f6:8a:d0), Dst: 02:42:c0:a8:fc:02 (02:42:c0:a8:fc:02)
 Internet Protocol Version 4, Src: 192.168.252.1, Dst: 192.168.252.2
 Transmission Control Protocol, Src Port: 50024, Dst Port: 5060, Seq: 1, Ack: 1, Len: 412
 Session Initiation Protocol (REGISTER)
 Request-Line: REGISTER sip:192.168.252.2 SIP/2.0
 Method: REGISTER
 Request-URI: sip:192.168.252.2
 [Resent Packet: False]
 Message Header
 Contact: <sip:252001@192.168.252.1:50024>
 Expires: 600
 To: <sip:252001@192.168.252.2>
 Via: SIP/2.0/TCP 192.168.252.1:50024;alias;rport;branch=z9hG4bK1387801638
 From: <sip:252001@192.168.252.2>;tag=629961919
 Call-ID: 1172882363@192.168.252.2
 CSeq: 5 REGISTER
 User-Agent: YATE/6.0.0
 Max-Forwards: 70
 Allow: ACK, INVITE, BYE, CANCEL, OPTIONS, INFO
 Content-Length: 0

En el primer mensaje, el cliente (192.168.252.1) realiza una petición de REGISTER a la centralita (192.168.252.2). Además, podemos ver la siguiente información:

- El usuario de la centralita es el 252001
- La señalización es TCP (SIP/2.0/TCP)
- El cliente VoIP es YATE
- Se pueden usar los siguientes mensajes: ACK, INVITE, BYE, CANCEL, OPTIONS, INFO

No.	Time	Source	Destination	Protocol	Length	Info
4	0.020178061	192.168.252.1	192.168.252.2	SIP	478	Request: REGISTER sip:192.168.252.2 (1 binding)
6	0.020414722	192.168.252.2	192.168.252.1	SIP	590	Status: 401 Unauthorized
8	0.060733732	192.168.252.1	192.168.252.2	SIP	640	Request: REGISTER sip:192.168.252.2 (1 binding)
9	0.063130973	192.168.252.2	192.168.252.1	SIP	610	Status: 200 OK (1 binding)

+ Frame 6: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fc:02 (02:42:c0:a8:fc:02), Dst: 02:42:f6:f6:8a:d0 (02:42:f6:f6:8a:d0)
+ Internet Protocol Version 4, Src: 192.168.252.2, Dst: 192.168.252.1
+ Transmission Control Protocol, Src Port: 50024, Dst Port: 50024, Seq: 1, Ack: 413, Len: 524

Session Initiation Protocol (401)

- Status-Line: SIP/2.0 401 Unauthorized
 - [Status-Code: 401]
 - [Resent Packet: False]
- Message Header
 - + Via: SIP/2.0/TCP 192.168.252.1:50024;alias;branch=z9hG4bK1387801638;received=192.168.252.1;rport=50024
 - + From: <sip:252001@192.168.252.2>;tag=629961919
 - + To: <sip:252001@192.168.252.2>;tag=as6f2adbeb
 - + Call-ID: 1172882363@192.168.252.2
 - + CSeq: 5 REGISTER
 - Server: Asterisk PBX 14.6.2
 - Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE
 - Supported: replaces, timer
 - + WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="434d802b"
 - Authentication Scheme: Digest
 - Algorithm: MD5
 - Realm: "asterisk"
 - Nonce Value: "434d802b"
 - Content-Length: 0

La centralita responde con un 401 (Unauthorized). Asterisk solicita que el usuario se autentique antes de registrarlo. Entre sus opciones, indica que se debe hacer mediante MD5 con una semilla concreta (434d802b). Además, podemos obtener metainformación como el cliente de la centralita o los métodos SIP que podemos realizar.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.020178061	192.168.252.1	192.168.252.2	SIP	478	Request: REGISTER sip:192.168.252.2 (1 binding)
6	0.020414722	192.168.252.2	192.168.252.1	SIP	590	Status: 401 Unauthorized
8	0.060733732	192.168.252.1	192.168.252.2	SIP	640	Request: REGISTER sip:192.168.252.2 (1 binding)
9	0.063130973	192.168.252.2	192.168.252.1	SIP	610	Status: 200 OK (1 binding)

+ Frame 8: 640 bytes on wire (5120 bits), 640 bytes captured (5120 bits) on interface 0
+ Ethernet II, Src: 02:42:f6:f6:8a:d0 (02:42:f6:f6:8a:d0), Dst: 02:42:c0:a8:fc:02 (02:42:c0:a8:fc:02)
+ Internet Protocol Version 4, Src: 192.168.252.1, Dst: 192.168.252.2
+ Transmission Control Protocol, Src Port: 50024, Dst Port: 5060, Seq: 413, Ack: 525, Len: 574

Session Initiation Protocol (REGISTER)

- Request-Line: REGISTER sip:192.168.252.2 SIP/2.0
 - [Method: REGISTER]
 - [Request-URI: sip:192.168.252.2]
 - [Resent Packet: False]
- Message Header
 - Contact: <sip:252001@192.168.252.1:50024>
 - + Contact URI: sip:252001@192.168.252.1:50024
 - Expires: 600
 - To: <sip:252001@192.168.252.2>
 - + SIP to address: sip:252001@192.168.252.2
 - + Via: SIP/2.0/TCP 192.168.252.1:50024;alias;rport;branch=z9hG4bK1616429740
 - + From: <sip:252001@192.168.252.2>;tag=629961919
 - + Call-ID: 1172882363@192.168.252.2
 - User-Agent: YATE/6.0.0
 - Max-Forwards: 70
 - Allow: ACK, INVITE, BYE, CANCEL, OPTIONS, INFO
 - CSeq: 6 REGISTER
 - + Authorization: Digest username="252001", realm="asterisk", nonce="434d802b", uri="sip:192.168.252.2", response="b8edb01906ba5d4d3d68ea1
 - Authentication Scheme: Digest
 - Username: "252001"
 - Realm: "asterisk"
 - Nonce Value: "434d802b"
 - Authentication URI: "sip:192.168.252.2"
 - Digest Authentication Response: "b8edb01906ba5d4d3d68ea1dc771520d"
 - Algorithm: MD5
 - Content-Length: 0

El cliente vuelve a realizar REGISTER, esta vez con la información solicitada por la centralita en el anterior mensaje. Podemos ver que se incluye un MD5 de la contraseña + nonce.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.020178061	192.168.252.1	192.168.252.2	SIP	478	Request: REGISTER sip:192.168.252.2 (1 binding)
6	0.020414722	192.168.252.2	192.168.252.1	SIP	590	Status: 401 Unauthorized
8	0.060733732	192.168.252.1	192.168.252.2	SIP	640	Request: REGISTER sip:192.168.252.2 (1 binding)
9	0.063130973	192.168.252.2	192.168.252.1	SIP	610	Status: 200 OK (1 binding)

Frame 9: 610 bytes on wire (4880 bits), 610 bytes captured (4880 bits) on interface 0

- + Ethernet II, Src: 02:42:c0:a8:fc:02 (02:42:c0:a8:fc:02), Dst: 02:42:f6:f6:8a:d0 (02:42:f6:f6:8a:d0)
- + Internet Protocol Version 4, Src: 192.168.252.2, Dst: 192.168.252.1
- + Transmission Control Protocol, Src Port: 5060, Dst Port: 50024, Seq: 525, Ack: 987, Len: 544
- Session Initiation Protocol (200)
 - Status-Line: SIP/2.0 200 OK
 - Status-Code: 200
 - [Resent Packet: False]
 - Message Header
 - + Via: SIP/2.0/TCP 192.168.252.1:50024;alias;branch=z9hG4bK1616429740;received=192.168.252.1;rport=50024
 - + From: <sip:252001@192.168.252.2>;tag=629961919
 - + SIP from address: sip:252001@192.168.252.2
 - + SIP from tag: 629961919
 - + To: <sip:252001@192.168.252.2>;tag=as6f2adbeb
 - + SIP to address: sip:252001@192.168.252.2
 - + SIP to tag: as6f2adbeb
 - + Call-ID: 1172882363@192.168.252.2
 - + CSeq: 6 REGISTER
 - Server: Asterisk PBX 14.6.2
 - Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE
 - Supported: replaces, timer
 - Expires: 600
 - + Contact: <sip:252001@192.168.252.1:50024>;expires=600
 - + Contact URI: sip:252001@192.168.252.1:50024
 - Contact parameter: expires=600
 - Date: Wed, 06 Dec 2017 09:13:32 GMT
 - Content-Length: 0

Tras verificar que la autenticación, el servidor asocia al cliente con su dirección física y responde con un SIP 200 (OK)

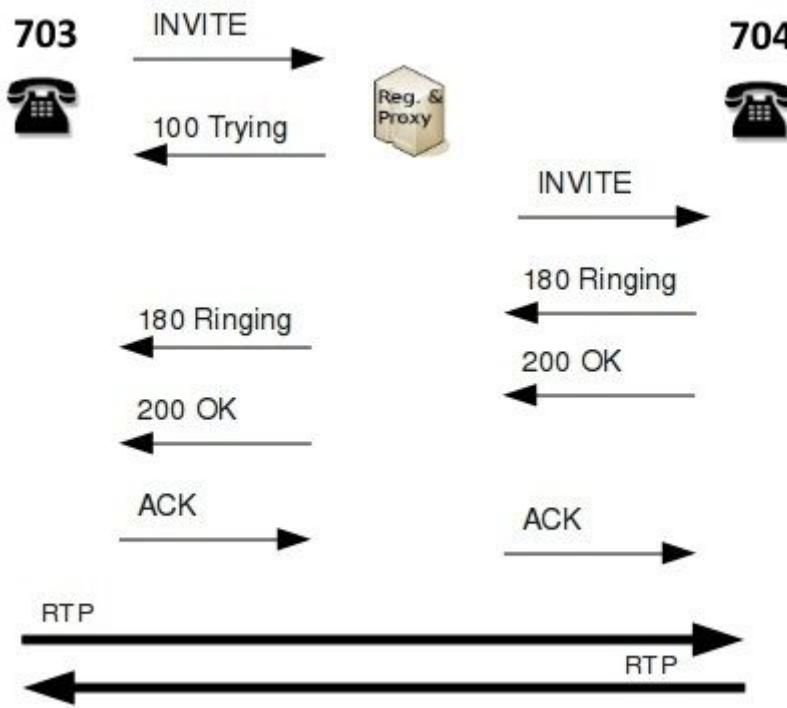
3.2.3.2 SIP INVITE

Para el establecimiento de una llamada, interviene esencialmente las operaciones INVITE:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.251.1	192.168.251.2	SIP/SDP	1007	Request: INVITE sip:251002@192.168.251.2
2	0.000022434	192.168.251.2	192.168.251.1	TCP	66	5060 -> 58230 [ACK] Seq=1 Ack=942 Win=326 Len=0 TSval=772684
3	0.000248697	192.168.251.2	192.168.251.1	SIP	589	Status: 401 Unauthorized
4	0.040300220	192.168.251.1	192.168.251.2	SIP	429	Request: ACK sip:251002@192.168.251.2
5	0.08059785	192.168.251.2	192.168.251.1	TCP	66	5060 -> 58230 [ACK] Seq=524 Ack=1305 Win=343 Len=0 TSval=772
6	0.080641299	192.168.251.1	192.168.251.2	SIP/SDP	1176	Request: INVITE sip:251002@192.168.251.2
7	0.080649811	192.168.251.2	192.168.251.1	TCP	66	5060 -> 58230 [ACK] Seq=524 Ack=2415 Win=360 Len=0 TSval=772
8	0.081290793	192.168.251.2	192.168.251.1	SIP	548	Status: 100 Trying
9	0.082130665	192.168.251.2	192.168.251.1	SIP/SDP	943	Request: INVITE sip:251002@192.168.251.1:58234
10	0.082145231	192.168.251.1	192.168.251.2	TCP	66	58234 -> 5060 [ACK] Seq=1 Ack=878 Win=287 Len=0 TSval=772709
11	0.123965604	192.168.251.1	192.168.251.2	TCP	66	58230 -> 5060 [ACK] Seq=2415 Ack=1006 Win=296 Len=0 TSval=77
12	0.134613543	192.168.251.1	192.168.251.2	SIP	385	Status: 100 Trying
13	0.177286391	192.168.251.2	192.168.251.1	TCP	66	5060 -> 58234 [ACK] Seq=878 Ack=320 Win=280 Len=0 TSval=7727
14	0.177298702	192.168.251.1	192.168.251.2	SIP	492	Status: 180 Ringing
15	0.177306926	192.168.251.2	192.168.251.1	TCP	66	5060 -> 58234 [ACK] Seq=878 Ack=746 Win=289 Len=0 TSval=7727
16	0.177619589	192.168.251.2	192.168.251.1	SIP	564	Status: 180 Ringing
17	0.177636515	192.168.251.1	192.168.251.2	TCP	66	58230 -> 5060 [ACK] Seq=2415 Ack=1504 Win=305 Len=0 TSval=77
18	2.305117268	192.168.251.1	192.168.251.2	SIP/SDP	705	Status: 200 OK
19	2.305137814	192.168.251.2	192.168.251.1	TCP	66	5060 -> 58234 [ACK] Seq=878 Ack=1385 Win=299 Len=0 TSval=773

Hay que tener en cuenta que, según nuestra configuración en la centralita, el establecimiento de llamada se realizará a través de Asterisk o directamente entre los terminales.

Podemos ver claramente el proceso en esta representación más gráfica:



Como podemos observar, tras realizar el cliente un primer INVITE, la centralita realiza el INVITE al cliente que se quiere llamar y envía mensajes RINGING hasta que el otro cliente contesta.

En ese caso, Asterisk manda al solicitante un OK. Por último, la centralita manda un ACK al solicitado y empieza la comunicación RTP.

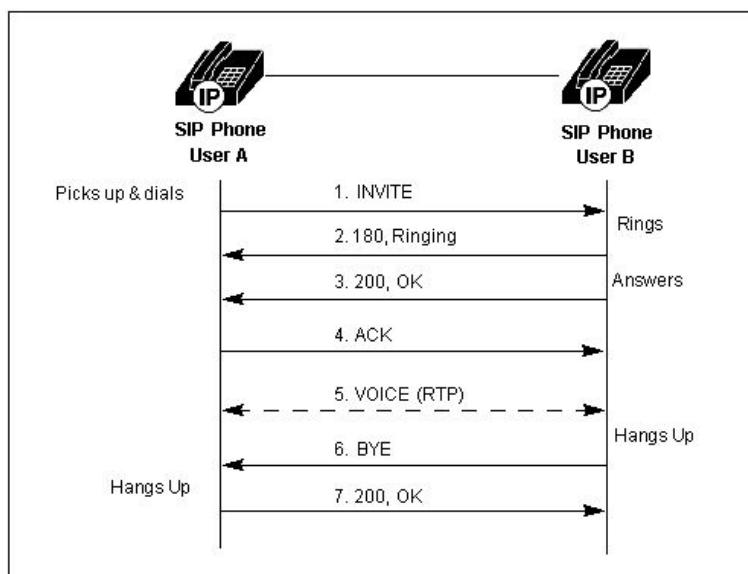
En nuestro caso, el OK se recibe en la trama 18:

No.	Time	Source	Destination	Protocol	Length	Info
13	0.177266391	192.168.251.2	192.168.251.1	TCP	66	5060 → 58234 [ACK] Seq=878 Ack=320 Win=280 Len=0 TS
14	0.177298702	192.168.251.1	192.168.251.2	SIP	492	Status: 180 Ringing
15	0.177306926	192.168.251.2	192.168.251.1	TCP	66	5060 → 58234 [ACK] Seq=878 Ack=746 Win=289 Len=0 TS
16	0.177619589	192.168.251.2	192.168.251.1	SIP	564	Status: 180 Ringing
17	0.177636515	192.168.251.1	192.168.251.2	TCP	66	58230 → 5060 [ACK] Seq=2415 Ack=1504 Win=305 Len=0 TS
18	2.305117268	192.168.251.1	192.168.251.2	SIP/SDP	705	Status: 200 OK
19	2.305137814	192.168.251.2	192.168.251.1	TCP	66	5060 → 58234 [ACK] Seq=878 Ack=1385 Win=299 Len=0 TS
20	2.305340472	192.168.251.2	192.168.251.1	SIP	485	Request: ACK sip:251002@192.168.251.1:58234
21	2.305348672	192.168.251.1	192.168.251.2	TCP	66	58234 → 5060 [ACK] Seq=1385 Ack=1297 Win=300 Len=0 TS
22	2.305857508	192.168.251.2	192.168.251.1	SIP/SDP	845	Status: 200 OK
23	2.305871700	192.168.251.1	192.168.251.2	TCP	66	58230 → 5060 [ACK] Seq=2415 Ack=2283 Win=317 Len=0 TS
24	2.306327604	192.168.251.2	192.168.251.1	SIP/SDP	921	Request: INVITE sip:251002@192.168.251.1:58234, in-addr-a

+ Frame 18: 705 bytes on wire (5640 bits), 705 bytes captured (5640 bits) on interface 0
+ Ethernet II, Src: 02:42:f7:bb:69:14 (02:42:f7:bb:69:14), Dst: 02:42:c0:a8:fb:02 (02:42:c0:a8:fb:02)
+ Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.2
+ Transmission Control Protocol, Src Port: 58234, Dst Port: 5060, Seq: 746, Ack: 878, Len: 639
+ Session Initiation Protocol (200)
- Status-Line: SIP/2.0 200 OK
 Status-Code: 200
 [Resent Packet: False]
- Message Header
 + Via: SIP/2.0/TCP 192.168.251.2:5060;branch=z9hG4bK63ef0915;received=192.168.251.2
 - From: "Alice" <sip:251001@192.168.251.2>;tag=as3ac222db
 SIP Display info: "Alice"
 + SIP from address: sip:251001@192.168.251.2
 + SIP from tag: as3ac222db
 - To: <sip:251002@192.168.251.1:58234>;tag=1988302493
 + SIP to address: sip:251002@192.168.251.1:58234
 + SIP to tag: 1988302493
 Call-ID: 16c868fc68503a70284d83382111a911@192.168.251.2:5060
 - CSeq: 102 INVITE
 Server: YATE/6.0.0
 - Contact: <sip:251002@192.168.251.1:58234>
 + Contact URI: sip:251002@192.168.251.1:58234
 Allow: ACK, INVITE, BYE, CANCEL, OPTIONS, INFO
 Content-Type: application/sdp
 Content-Length: 185
- Message Body
- Session Description Protocol
 Session Description Protocol Version (v): 0
 + Owner/Creator, Session Id (o): yate 1512553046 1512553046 IN IP4 192.168.251.1
 Session Name (s): SIP Call
 + Connection Information (c): IN IP4 192.168.251.1
 + Time Description, active time (t): 0 0
 + Media Description, name and address (m): audio 26996 RTP/AVP 8 101
 + Media Attribute (a): rtpmap:8 PCMA/8000
 + Media Attribute (a): rtpmap:101 telephone-event/8000

Podemos observar que en el mensaje SIP hay un nuevo campo, SDP. En dicho campo se describe toda la información relativa a la sesión (llamada): ID, codecs, protocolos, direcciones, etc.

Cuando la llamada finaliza, el cliente envía una señal BYE:



A lo que el otro cliente responde con un OK. Esto podemos observarlo en las siguientes tramas:

No.	Time	Source	Destination	Protocol	Length	Info
201	3.954568827	192.168.251.1	192.168.251.2	SIP	529	Request: BYE sip:251001@192.168.251.2:5060;transport=tcp
202	3.955004528	192.168.251.2	192.168.251.1	SIP	534	Status: 200 OK

Frame 201: 529 bytes on wire (4232 bits), 529 bytes captured (4232 bits) on interface 0
Ethernet II, Src: 02:42:f7:bb:69:14 (02:42:f7:bb:69:14), Dst: 02:42:c0:a8:fb:02 (02:42:c0:a8:fb:02)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.2
Transmission Control Protocol, Src Port: 58234, Dst Port: 5060, Seq: 9169, Ack: 11489, Len: 463
Session Initiation Protocol (BYE)
Request-Line: BYE sip:251001@192.168.251.2:5060;transport=tcp SIP/2.0
Method: BYE
Request-URI: sip:251001@192.168.251.2:5060;transport=tcp
[Resent Packet: False]
Message Header
Call-ID: 16c868fc68503a70284d83382111a911@192.168.251.2:5060
From: <sip:251002@192.168.251.1:58234>;tag=1988302493
To: <sip:251001@192.168.251.2>;tag=as3ac222db
P-RTT-Stat: PS=3, OS=480, PR=1, OR=160, PL=0
Via: SIP/2.0/TCP 192.168.251.1:58234;alias;rport;branch=z9hG4bK1470429295
CSeq: 31 BYE
User-Agent: YATE/6.0.0
Max-Forwards: 70
Allow: ACK, INVITE, BYE, CANCEL, OPTIONS, INFO
Content-Length: 0

3.3 Owncloud

ownCloud es una aplicación de software libre del tipo Servicio de alojamiento de archivos, que permite el almacenamiento en línea y aplicaciones en línea (cloud computing).

ownCloud puede ser instalado dentro de un servidor que disponga de una versión reciente de PHP (mayor o igual a 5.6) y soporte de SQLite (base de datos por defecto), MySQL o PostgreSQL.

Owncloud	
owncloud.org	
	
Información general	
Desarrolladores	ownCloud GmbH. Comunidad
Última versión estable	10.0.3 15 de septiembre de 2017
Género	Cloud computing
Programado en	PHP y JavaScript
Licencia	AGPLv3
Sistema Operativo	Multiplataforma

3.3.1 Archivos de configuración

Owncloud es un software pensado para instalar-ejecutar: viene empaquetado con todo lo necesario para ponerlo en marcha sin realizar apenas esfuerzo.

Sin embargo, es interesante comentar dos aspectos de este servicio: La puesta en marcha inicial (contraseña maestra, etc.) y el soporte para TLS, junto a la configuración del OpenLDAP.

Por un lado, la configuración TLS es relativa al servidor web Apache. Para desplegar el servicio Owncloud con nuestros propios certificados y CRLs, debemos modificar los archivos `topath.conf` y `subpath.conf`, quienes se usan con los inicializadores de `/etc/owncloud.d/`.

En ambos, modificaremos la parte relacionada con el SSL:

```
SSLEngine on
SSLCertificateFile ${OWNCLOUD_VOLUME_CERTS}/owncloud-cert.pem
SSLCertificateKeyFile ${OWNCLOUD_VOLUME_CERTS}/owncloud-key.pem
SSLCertificateChainFile ${OWNCLOUD_VOLUME_CERTS}/um.pem
SSLCARevocationFile /etc/apache2/crl/um.crl
```

Por otro lado, cuando el servicio se ha iniciado, no está configurado con nuestra configuración. Usaremos un script bash que haga el trabajo:

```
#!/bin/bash
set -e

#CRLs
mkdir /etc/apache2/crl
wget http://distribution.um.es/um.crl -P /etc/apache2/crl

supervisorctl start owncloud

echo "Waiting owncloud to launch on 80..."

while ! nc -z localhost 80; do
    sleep 0.1 # wait for 1/10 of the second before check again
done

echo "Owncloud launched"

su - www-data -s /bin/bash -c "php /var/www/owncloud/occ app:enable user_ldap"

#Delete previous configs
while read -r conf; do
    su - www-data -s /bin/bash -c "php /var/www/owncloud/occ ldap:delete-config '$conf'"
done <<< "$(
    su - www-data -s /bin/bash -c "php /var/www/owncloud/occ ldap:show-config | grep 'Configuration' | tr -d ' ' | cut -d'|' -f3"
)"

su - www-data -s /bin/bash -c "cd /var/www/owncloud/ && \
    php occ app:enable user_ldap && \
    php occ ldap:create-empty-config && \
    php occ ldap:set-config '' ldapHost ldap.um.es && \
    php occ ldap:set-config '' ldapPort 389 && \
    php occ ldap:set-config '' ldapAgentName cn=admin,dc=um,dc=es && \
    php occ ldap:set-config '' ldapAgentPassword um_password && \
    php occ ldap:set-config '' ldapBase dc=um,dc=es && \
    php occ ldap:set-config '' ldapUserFilter \
'(&(|(objectclass/inetOrgPerson))(|(memberof=cn=owncloud,ou=Servicios,dc=um,dc=es)))' && \
    php occ ldap:set-config '' ldapLoginFilter \
'(&(&(|(objectclass/inetOrgPerson))(|(memberof=cn=owncloud,ou=Servicios,dc=um,dc=es))))(|(cn=%uid)(|('
```

```

mailPrimaryAddress=%uid)(mail=%uid))))' &\ 
    php occ ldap:set-config '' ldapLoginFilterEmail 1 && \
    php occ ldap:set-config '' ldapUserDisplayName cn && \
    php occ ldap:set-config '' ldapUserFilterObjectclass inetOrgPerson && \
    php occ ldap:set-config '' ldapConfigurationActive 1 && \
    php occ ldap:set-config '' hasMemberOfFilterSupport 1 && \
    php occ ldap:set-config '' ldapUserFilterGroups owncloud && \
    php occ ldap:set-config '' ldapBaseGroups dc=um,dc=es && \
    php occ ldap:set-config '' ldapBaseUsers dc=um,dc=es && \
    php occ ldap:set-config '' ldapQuotaAttribute postOfficeBox && \
    php occ ldap:set-config '' ldapQuotaDefault 1073741824"

```

El procedimiento se divide en 4 partes:

1. Descargamos la lista de revocación en el directorio que hemos usado previamente con la puesta en marcha de TLS
2. Esperamos a que el servicio se ponga en marcha
3. Limpiamos posible configuración previa. El motivo de hacerlo es que si levantamos el proyecto de docker compose dos veces, este script se ejecuta dos veces, creando configuración residual y rompiendo la funcional
4. Inicializamos LDAP con todos los parámetros relativos a nuestra organización

3.3.2 Despliegue

El despliegue de este servicio, se define así en el docker-compose de forma homóloga para cada universidad:

```

um_owncloud:
  build: ./um/owncloud
  hostname: owncloud
  domainname: um.es
  volumes:
    - um-certs-owncloud:/etc/apache2/keys/
  depends_on:
    - "um_db"
    - "um_opendap"
    - "um_crl"
    - "um_snmp"
  links:
    - um_db:owncloud-db
  ports:
    - "80:80"
    - "443:443"
  environment:
    - OWNCLLOUD_DB_HOST=db.um.es
    - OWNCLLOUD_DB_TYPE=pgsql
    - OWNCLLOUD_DB_NAME=owncloud
    - OWNCLLOUD_DB_USERNAME=postgres
    - OWNCLLOUD_DB_PASSWORD=owncloud
    - OWNCLLOUD_ADMIN_USERNAME=owncloud

```

```

- OWNCLLOUD_ADMIN_PASSWORD=owncloud
- OWNCLLOUD_VOLUME_CERTS=/etc/apache2/keys/
networks:
red1:
    aliases:
        - owncloud.um.es
    ipv4_address: 192.168.251.5

```

Definimos el directorio donde se encuentra el servicio y la dirección IP, entre otros detalles como el dominio y el hostname.

Además, este servicio se ejecuta después de otros servicios (SNMP, LDAP, etc.) y tiene un volumen: El certificado con la clave privada del servicio (TLS).

Por otra parte, el Dockerfile es el siguiente:

```

FROM owncloud/server:9.1.6
LABEL maintainer="valiantsin.kivachuk@um.es"

ENV DEBIAN_FRONTEND noninteractive

#TODO ffix duplicated sourcelist
RUN echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main " >>
/etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main " >>
/etc/apt/sources.list.d/apt-fast.list && \
    apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
    apt-get update && apt-get --no-install-recommends -y install apt-fast && apt-fast
install -y \
    net-tools \
    traceroute \
    netcat \
    dnsutils \
    iputils-ping \
    curl \
    nano \
    supervisor \
    snmpd snmp snmp-mibs-downloader \
    libldap2-dev \
&& rm -rf /var/lib/apt/lists/

COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf
COPY conf/snmpd.conf /etc/snmp/snmpd.conf

#Do magic for LDAP
COPY misc/owncloud_ldap.sh /usr/bin

#SSL
RUN mkdir /etc/apache2/keys
RUN ln -s /certs/owncloud-cert.pem /etc/apache2/keys/ssl-cert.crt
RUN ln -s /certs/owncloud-key.pem /etc/apache2/keys/ssl-cert.key

COPY conf/apache_sites-available/toppath.conf /root/owncloud/toppath.conf
COPY conf/apache_sites-available/subpath.conf /root/owncloud/subpath.conf

CMD ["/usr/bin/supervisord"]

```

A diferencia de los demás servicios, este tiene 2 secciones principales: Descargar las dependencias, y trasladar todos los archivos de configuración al contenedor, pues trabajamos con la imagen oficial de Owncloud.

Por último, la configuración de Supervisord:

```
[supervisord]
nodaemon=true

[program:snmpd]
command=/etc/init.d/snmpd start
priority=100
startretries=0

[program:owncloud]
command=owncloud
priority=20
startretries=0
autostart=false

[program:owncloud_ldap]
command=/usr/bin/owncloud_ldap.sh
priority=100
```

3.4 FreeRADIUS

FreeRADIUS es una suite RADIUS modular de alto rendimiento, desarrollada y distribuida bajo la Licencia Pública General de GNU, versión 2, y es gratuita para descargar y usar. FreeRADIUS Suite incluye un servidor RADIUS, una biblioteca cliente RADIUS con licencia BSD, una biblioteca PAM, un módulo Apache y numerosas utilidades adicionales relacionadas con RADIUS y bibliotecas de desarrollo.

En la mayoría de los casos, la palabra "FreeRADIUS" hace referencia al servidor RADIUS de código abierto gratuito de este paquete. Es el servidor RADIUS de código abierto más popular y ampliamente implementado en el mundo, compatible con todos los protocolos de autenticación comunes.

FreeRADIUS	
freeradius.org	
freeRADIUS	
Información general	
Desarrolladores	FreeRADIUS Development Team
Última versión estable	3.0.15 17 de Julio de 2017
Género	Servidor Radius
Programado en	C y scripts Perl
Licencia	GPLv2
Sistema Operativo	Tipo Unix

3.4.1 Archivos de configuración

Para la configuración del servicio RADIUS en cada una de las organizaciones, tenemos 4 archivos principales que tenemos que modificar para adaptar el servicio a nuestras necesidades:

1. **eap.conf**. Núcleo de autenticación del servidor RADIUS. Se definen los distintos métodos de autenticación como los mecanismos relacionados (TLS, mschapv2, etc.)

-
- a. Es preciso desmarcar los módulos *eap* del archivo default
 - 2. **proxy.conf**. En este apartado definimos los realms: dominio que define una entidad. Aquí se establece dónde se debe autenticar cada tipo de usuario.
 - 3. **clients.conf**. Establecemos los servidores RADIUS externos que puedan llegar a usarse en algún momento en nuestro servidor.
 - 4. **ldap**. Archivo que provee la configuración para delegar la autenticación en un servidor LDAP externo. Implica descomentar los módulos *ldap* del siguientes archivo:
 - a. inner-tunnel
 - b. default

Por tanto, tenemos los siguientes archivos :

eap.conf

```

eap {
    default_eap_type = peap
    timer_expire    = 60
    ignore_unknown_eap_types = no
    cisco_accounting_username_bug = no
    max_sessions = ${max_requests}
    md5 {
    }
    leap {
    }
    gtc {
        auth_type = PAP
    }
    tls {
        certdir = ${confdir}/keys
        cadir = ${confdir}/keys
        private_key_file = ${certdir}/radius-key.pem
        certificate_file = ${certdir}/radius-cert.pem
        CA_file = ${cadir}/um.pem
        dh_file = ${certdir}/dh
        random_file = /dev/urandom
        CA_path = ${cadir}
        check_cert_cn = %{User-Name}
        cipher_list = "DEFAULT"
        make_cert_command = "${certdir}/bootstrap"
        ecdh_curve = "prime256v1"
        cache {
            enable = no
            lifetime = 24
            max_entries = 255
        }
        verify {
        }
        ocsp {
            enable = yes
        }
    }
}
```

```

        override_cert_url = no
    }
}
ttls {
    default_eap_type = md5
    copy_request_to_tunnel = no
    use_tunneled_reply = no
    virtual_server = "inner-tunnel"
}
peap {
    default_eap_type = mschapv2
    copy_request_to_tunnel = no
    use_tunneled_reply = no
    virtual_server = "inner-tunnel"
}
mschapv2 {
}
}

```

Definimos que usaremos EAP junto a nuestros propios certificados, ubicados **/etc/freeradius/keys**. Destacar el campo *check_cert_cn*, quien se encarga de denegar el acceso a los certificados que no tengan el identificador en el CN del certificado.

proxy.conf:

```

realm upm.es {
    #Autenticacion y autorizacion.
    authhost = 192.168.252.3:1812
    accthost = 192.168.252.3:1813
    secret = upm_router_password
    nostrip
}

realm um.es {
    authhost = LOCAL
    accthost = LOCAL
}

```

Establecemos que nuestra organización es *um.es* y la organización vecina, con su correspondiente servidor radius y secreto

clients.conf

```

client 192.168.252.3 {
    secret      = um_router_password
    shortname   = upm_router
}

client 192.168.251.0/24 {

```

```

secret      = um_router_password
shortname   = um_router
}

```

En este apartado establecemos el secreto compartido que tendrán los diversos clientes con nuestro servidor RADIUS. En el caso de la *UM*, será **um_router_password**.

ldap:

```

ldap {
    server = "ldap.um.es"
    identity = "cn=admin,dc=um,dc=es"
    password = um_password
    basedn = "dc=um,dc=es"
    filter =
"(&(uid=%{${Stripped-User-Name}}:-%{User-Name} })(memberof=cn=radius,ou=Se
rvicios,dc=um,dc=es))"

...

```

Por último, la configuración relativa a nuestro servidor LDAP, junto con el filtro del grupo. Remarcar que el servidor RADIUS obtiene todos los atributos que tenga el servidor LDAP en el objeto *radiusprofile*, por lo que podemos establecer parámetros especiales de conexión para ciertos usuarios.

3.4.2 Despliegue

El despliegue de este servicio, se define así en el docker-compose de forma homóloga para cada universidad:

```

um_radius:
    build: ./um/radius/
    hostname: radius
    domainname: um.es
    volumes:
        - um-certs-radius:/etc/freeradius/keys/
    depends_on:
        - "um_snmp"
        - "um_opendap"
    networks:
        red1:
            aliases:
                - radius.um.es
    ipv4_address: 192.168.251.3

```

Definimos el directorio donde se encuentra el servicio y la dirección IP, entre otros detalles como el dominio y el hostname.

Además, este servicio se ejecuta después de otros servicios (SNMP y LDAP) y tiene un volumen: El certificado con la clave privada del servicio (TLS).

Por otra parte, el Dockerfile es el siguiente:

```
FROM ubuntu:16.04
LABEL maintainer="valiantsin.kivachuk@um.es"

#ORG1-N1

RUN echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >> /etc/apt/sources.list.d/apt-fast.list && \
    apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
    apt-get update && apt-get --no-install-recommends -y install apt-fast && apt-fast
install -y \
net-tools \
traceroute \
netcat \
dnsutils \
iputils-ping \
curl \
nano \
ifupdown2 \
supervisor \
snmpd snmp snmp-mibs-downloader \
freeradius \
&& rm -rf /var/lib/apt/lists/

RUN apt-fast update && apt-fast install -y \
    freeradius-ldap \
&& rm -rf /var/lib/apt/lists/

COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf

RUN sed -i 's/.*mibs :/#/' /etc/snmp/snmp.conf #Insert comment

COPY conf/snmpd.conf /etc/snmp/snmpd.conf

#Freeradius
COPY conf/clients.conf /etc/freeradius/clients.conf
COPY conf/eap.conf /etc/freeradius/eap.conf
COPY conf/sites-available/default /etc/freeradius/sites-available/default
COPY conf/sites-available/inner-tunnel /etc/freeradius/sites-available/inner-tunnel
COPY conf/proxy.conf /etc/freeradius/proxy.conf
COPY conf/modules/ldap /etc/freeradius/modules/ldap

RUN sed -i 's/.*mibs :/#/' /etc/snmp/snmp.conf #Enable all MIBS

CMD ["/usr/bin/supervisord"]
```

Al igual que el anterior servicio, este tiene 2 secciones principales: Descargar las dependencias, y trasladar todos los archivos de configuración al contenedor.

Por último, la configuración de Supervisord:

```
[supervisord]
nodaemon=true

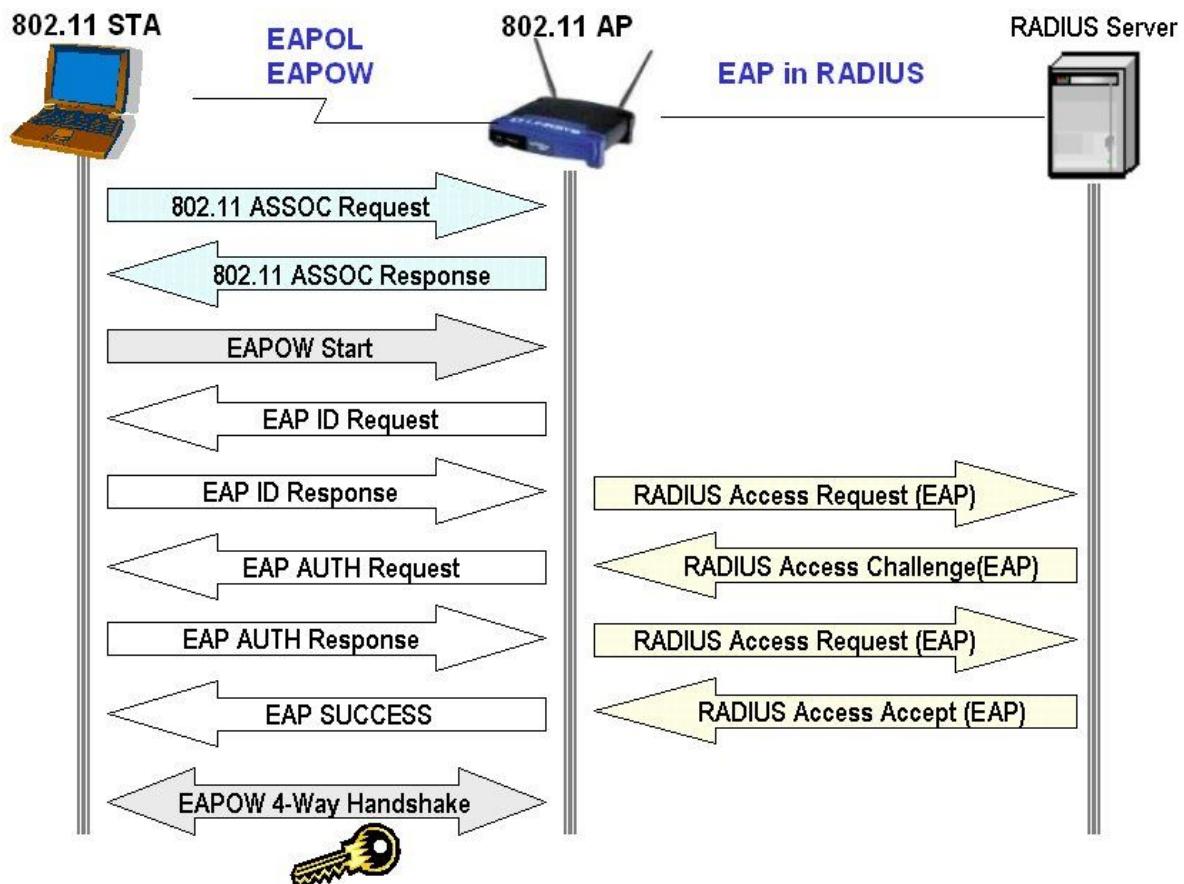
[program:snmpd]
command=/etc/init.d/snmpd start
priority=100
startretries=0

[program:freeradius]
command=freeradius -X -f -l /dev/stdout
priority=100
startretries=0
```

3.4.3 Trazas RADIUS

Para poder mostrar el completo funcionamiento de RADIUS, debemos comprender el contexto en el que ocurre el proceso de autenticación.

Cuando un usuario se asocia a un AP, entra en juego EAP, quien se dedica a coordinar los mecanismos de autenticación. Este proceso lo podemos observar en el siguiente diagrama:



http://tldp.org/HOWTO/html_single/8021X-HOWTO/

Es importante destacar que, por un lado, ocurre un proceso de autenticación **PEAP** o **EAP-TLS** entre el usuario y el cliente RADIUS, y por otro, la comunicación del servidor RADIUS con otras entidades

Para exemplificar este proceso, se captura el tráfico RADIUS durante una autenticación multidominio en la red *UM*.

3.4.3.1 Identity Request

Es el primer mensaje, enviado por el servidor al cliente. Tiene un campo *Type*, que es a lo que tiene que responder el cliente. Algunos valores:

- (1) Identity
- (2) Notification
- (3) Nak (Response only)
- (4) MD5-Challenge
- (5) One Time Password (OTP)
- (6) Generic Token Card (GTC)
- (254) Expanded Types
- (255) Experimental use

No.	Time	Source	Destination	Protocol	Length	Info
	73 3.039579597	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	83	Request, Identity
	79 3.124994054	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	EAP	95	Response, Identity
	81 3.132670217	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	84	Request, Protected EAP (EAP-PEAP)

+ Frame 73: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
+ Radiotap Header v0, Length 36
+ 802.11 radio information
+ IEEE 802.11 QoS Data, Flags:F.C
+ Logical-Link Control
+ 802.1X Authentication
■ Extensible Authentication Protocol
 Code: Request (1)
 Id: 0
 Length: 5
 Type: Identity (1)

El primer mensaje, es la petición de la identidad, *Identity Request*.

3.4.3.2 Identity Response

Este mensaje es la respuesta al *Request*, descrito anteriormente.

No.	Time	Source	Destination	Protocol	Length	Info
73	3.039579597	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	83	Request, Identity
79	3.124994054	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	EAP	95	Response, Identity
81	3.132670217	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	84	Request, Protected EAP (EAP-PEAP)

+ Frame 79: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0
+ Radiotap Header v0, Length 36
+ 802.11 radio information
+ IEEE 802.11 QoS Data, Flags:TC
+ Logical-Link Control
+ 802.1X Authentication
Extensible Authentication Protocol
Code: Response (2)
Id: 0
Length: 17
Type: Identity (1)
Identity: user1@upm.es

Como vemos, el cliente ha respondido a la solicitud del servidor del campo Identity, siendo **user1@upm.es** debido a que no hay una identidad anónima establecida.

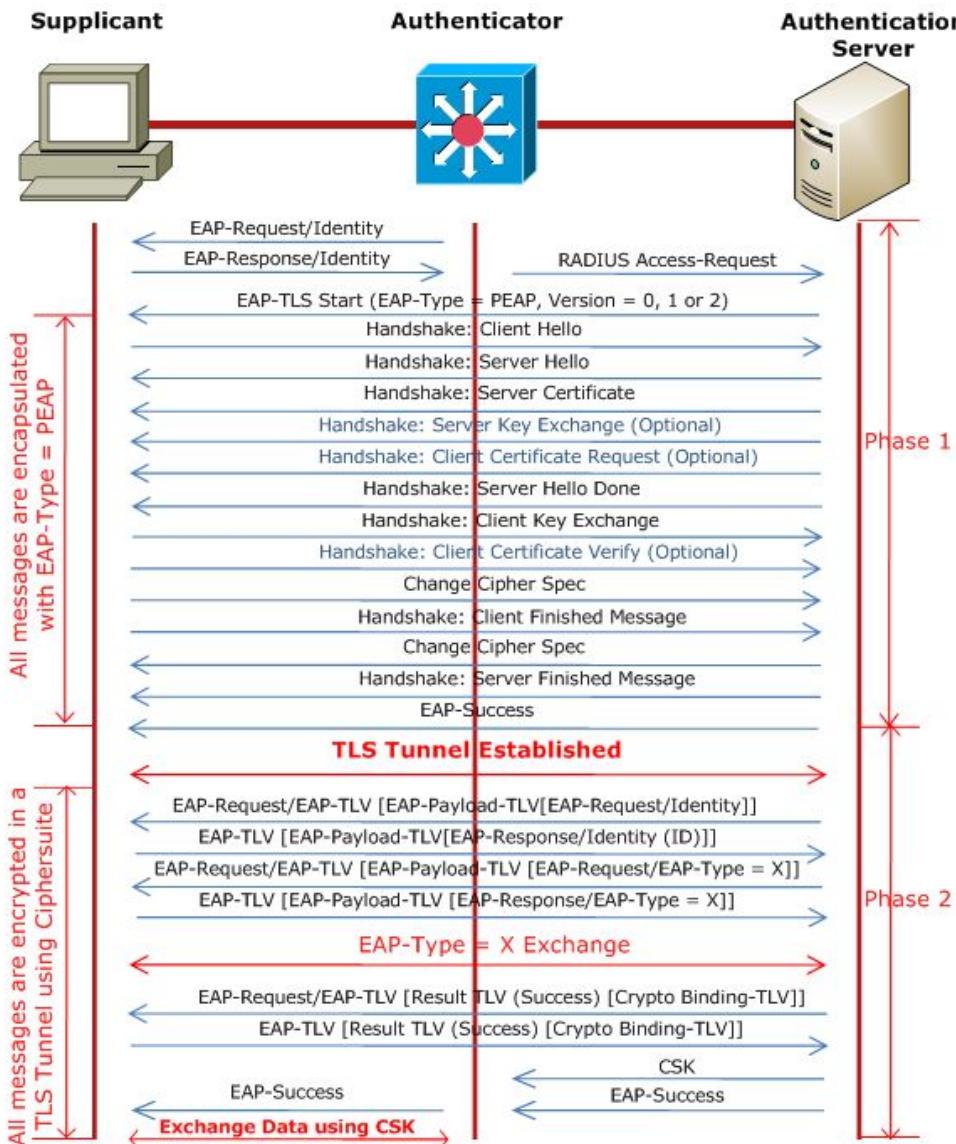
3.4.3.3 EAP Request

El servidor solicita al STA un método EAP de alto nivel (EAP-PEAP).

No.	Time	Source	Destination	Protocol	Length	Info
73	3.039579597	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	83	Request, Identity
79	3.124994054	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	EAP	95	Response, Identity
81	3.132670217	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	84	Request, Protected EAP (EAP-PEAP)

+ Frame 81: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
+ Radiotap Header v0, Length 36
+ 802.11 radio information
+ IEEE 802.11 QoS Data, Flags:F.C
+ Logical-Link Control
+ 802.1X Authentication
Extensible Authentication Protocol
Code: Request (1)
Id: 1
Length: 6
Type: Protected EAP (EAP-PEAP) (25)
EAP-TLS Flags: 0x20
0... = Length Included: False
.0... = More Fragments: False
..1. = Start: True
.... .000 = Version: 0

En este momento, empieza el procedimiento para establecer un túnel TLS para utilizar las credenciales del usuario:



<https://sites.google.com/site/amitsciscozone/home/switching/peap---protected-eap-protocol>

3.4.3.4 EAP Response

En esta trama, el cliente responde a la solicitud del servidor, tras la primera fase de establecimiento del canal seguro TLS:

No.	Time	Source	Destination	Protocol	Length	Info
73	3.039579597	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	83	Request, Identity
79	3.124994054	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	EAP	95	Response, Identity
81	3.132670217	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	EAP	84	Request, Protected EAP (EAP-PEAP)
83	3.137466897	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	TLSv1.2	253	Client Hello
85	3.152101445	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	TLSv1.2	1102	Server Hello, Certificate, Server Key Exchange, Server Name Indication
87	3.154328507	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	EAP	84	Response, Protected EAP (EAP-PEAP)

+ Frame 87: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
+ Radiotap Header v0, Length 36
+ 802.11 radio information
+ IEEE 802.11 QoS Data, Flags:TC
+ Logical-Link Control
+ 802.1X Authentication
- Extensible Authentication Protocol
 Code: Response (2)
 Id: 2
 Length: 6
 Type: Protected EAP (EAP-PEAP) (25)
 EAP-TLS Flags: 0x00
 0... = Length Included: False
 .0... = More Fragments: False
 ..0.... = Start: False
 000 = Version: 0

A continuación, se termina de establecer el canal seguro para el intercambio de credenciales con el servidor RADIUS:

102	3.213043632	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	TLSv1.2	118	Application Data
104	3.216954099	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	TLSv1.2	126	Application Data
106	3.222692847	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	TLSv1.2	147	Application Data
108	3.228651627	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	TLSv1.2	180	Application Data
110	3.242797955	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	TLSv1.2	160	Application Data
112	3.246813538	AirgoNet_09:e2:2c	BelkinIn_44:73:2b	TLSv1.2	115	Application Data
114	3.252378127	BelkinIn_44:73:2b	AirgoNet_09:e2:2c	TLSv1.2	124	Application Data

3.4.3.4 Radius

Durante el proceso de autorización, el primer mensaje es desde el router (cliente RADIUS) a nuestro servidor. Entre los parámetros, destacar el username (en el que también aparece el dominio al que pertenece, **@upm.es**)

No.	Time	Source	Destination	Protocol	Length	Info
57	3.364760994	192.168.251.1	192.168.251.3	RADIUS	226	Access-Request(1) (id=0, l=184)
60	3.365229434	192.168.251.3	192.168.252.3	RADIUS	229	Access-Request(1) (id=11, l=187)
61	3.366221838	192.168.252.3	192.168.251.3	RADIUS	212	Access-Accept(2) (id=11, l=170)
62	3.366300563	192.168.251.3	192.168.251.1	RADIUS	209	Access-Accept(2) (id=0, l=167)

+ Frame 57: 226 bytes on wire (1808 bits), 226 bytes captured (1808 bits) on interface 0
+ Ethernet II, Src: BelkinIn_44:73:29 (58:ef:68:44:73:29), Dst: 02:42:c0:a8:fb:03 (02:42:c0:a8:fb:03)
+ Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.3
+ User Datagram Protocol, Src Port: 2048, Dst Port: 1812
- RADIUS Protocol
 Code: Access-Request (1)
 Packet identifier: 0x0 (0)
 Length: 184
 Authenticator: dbcb1f2729e272d2aa23b56dd17318ed
 [The response to this request is in frame 62]
- Attribute Value Pairs
 + AVP: 1=14 t=User-Name(1): user1@upm.es
 + AVP: 1=6 t=NAS-IP-Address(4): 192.168.251.1
 + AVP: 1=14 t=Called-Station-Id(30): 58ef6844732b
 + AVP: 1=14 t=Calling-Station-Id(31): 000af509e22c
 + AVP: 1=14 t=NAS-Identifier(32): 58ef6844732b
 + AVP: 1=6 t=NAS-Port(5): 7
 + AVP: 1=6 t=Framed-MTU(12): 1400
 + AVP: 1=18 t=State(24): 6fd08d7868d894dcceeeec97b5a71a0
 + AVP: 1=6 t=NAS-Port-Type(61): Wireless-802.11(19)
 + AVP: 1=48 t=EAP-Message(79) Last Segment[1]
 + AVP: 1=18 t=Message-Authenticator(80): ff7a9496b3cf12c07569c985aae67ab1

Al no tratarse de su dominio, el servidor RADIUS de **UM** redirige la petición hacia el RADIUS de la organización **UPM**.

No.	Time	Source	Destination	Protocol	Length	Info
57	3.364760994	192.168.251.1	192.168.251.3	RADIUS	226	Access-Request(1) (id=0, l=184)
60	3.365229434	192.168.251.3	192.168.252.3	RADIUS	229	Access-Request(1) (id=11, l=187)
61	3.366221838	192.168.252.3	192.168.251.3	RADIUS	212	Access-Accept(2) (id=11, l=170)
62	3.366300563	192.168.251.3	192.168.251.1	RADIUS	209	Access-Accept(2) (id=0, l=167)

Frame 60: 229 bytes on wire (1832 bits), 229 bytes captured (1832 bits) on interface 0

Ethernet II, Src: 02:42:c0:a8:fb:03 (02:42:c0:a8:fb:03), Dst: BelkinIn_44:73:29 (58:ef:68:44:73:29)

Internet Protocol Version 4, Src: 192.168.251.3, Dst: 192.168.252.3

User Datagram Protocol, Src Port: 1814, Dst Port: 1812

RADIUS Protocol

- Code: Access-Request (1)
- Packet identifier: 0xb (11)
- Length: 187
- Authenticator: 9fef87cf754c39e29c6ad710dfc06723
- [The response to this request is in frame 61]

Attribute Value Pairs

- + AVP: 1=14 t=User-Name(1): user1@upm.es
- + AVP: 1=6 t=NAS-IP-Address(4): 192.168.251.1
- + AVP: 1=14 t=Called-Station-Id(30): 58ef6844732b
- + AVP: 1=14 t=Calling-Station-Id(31): 000af509e22b
- + AVP: 1=14 t=NAS-Identifier(32): 58ef6844732b
- + AVP: 1=6 t=NAS-Port(5): 7
- + AVP: 1=6 t=Framed-MTU(12): 1400
- + AVP: 1=18 t=State(24): 6fd08d7868d894dccceeeec97b5a71a0
- + AVP: 1=6 t=NAS-Port-Type(61): Wireless-802.11(19)
- + AVP: 1=48 t=EAP-Message(79) Last Segment[1]
- + AVP: 1=18 t=Message-Authenticator(80): 79d56a8f2842996b76a8f001a024320c
- + AVP: 1=3 t=Proxy-State(33): 30

Podemos verificar que se trata de la redirección por el campo *Proxy-State*.

Después, recibimos un mensaje del RADIUS de **UPM**:

No.	Time	Source	Destination	Protocol	Length	Info
57	3.364760994	192.168.251.1	192.168.251.3	RADIUS	226	Access-Request(1) (id=0, l=184)
60	3.365229434	192.168.251.3	192.168.252.3	RADIUS	229	Access-Request(1) (id=11, l=187)
61	3.366221838	192.168.252.3	192.168.251.3	RADIUS	212	Access-Accept(2) (id=11, l=170)
62	3.366300563	192.168.251.3	192.168.251.1	RADIUS	209	Access-Accept(2) (id=0, l=167)

Frame 61: 212 bytes on wire (1696 bits), 212 bytes captured (1696 bits) on interface 0

Ethernet II, Src: BelkinIn_44:73:29 (58:ef:68:44:73:29), Dst: 02:42:c0:a8:fb:03 (02:42:c0:a8:fb:03)

Internet Protocol Version 4, Src: 192.168.252.3, Dst: 192.168.251.3

User Datagram Protocol, Src Port: 1812, Dst Port: 1814

RADIUS Protocol

- Code: Access-Accept (2)
- Packet identifier: 0xb (11)
- Length: 170
- Authenticator: 852283ef0bcc27faf9a2538ffab33892
- [This is a response to a request in frame 60]
- [Time from request: 0.000992404 seconds]

Attribute Value Pairs

- + AVP: 1=58 t=Vendor-Specific(26) v=Microsoft(311)
- + AVP: 1=58 t=Vendor-Specific(26) v=Microsoft(311)
- + AVP: 1=6 t=EAP-Message(79) Last Segment[1]
- + AVP: 1=18 t=Message-Authenticator(80): 06c6f3e044e77077011c74c6ec1ab965
- + AVP: 1=7 t=User-Name(1): user1
- + AVP: 1=3 t=Proxy-State(33): 30

Vemos que el servidor de la otra organización autoriza al usuario **user1@upm.es** por el campo *Code (Access-Accept)*.

Por último, la respuesta de la organización **UPM** se redirige al router:

No.	Time	Source	Destination	Protocol	Length	Info
57	3.364760994	192.168.251.1	192.168.251.3	RADIUS	226	Access-Request(1) (id=0, l=184)
60	3.365229434	192.168.251.3	192.168.252.3	RADIUS	229	Access-Request(1) (id=11, l=187)
61	3.366221838	192.168.252.3	192.168.251.3	RADIUS	212	Access-Accept(2) (id=11, l=170)
62	3.366300563	192.168.251.3	192.168.251.1	RADIUS	209	Access-Accept(2) (id=0, l=167)

+ Frame 62: 209 bytes on wire (1672 bits), 209 bytes captured (1672 bits) on interface 0

+ Ethernet II, Src: 02:42:c0:a8:fb:03 (02:42:c0:a8:fb:03), Dst: BelkinIn_44:73:29 (58:ef:68:44:73:29)

+ Internet Protocol Version 4, Src: 192.168.251.3, Dst: 192.168.251.1

+ User Datagram Protocol, Src Port: 1812, Dst Port: 2048

□ RADIUS Protocol

- Code: Access-Accept (2)
- Packet identifier: 0x0 (0)
- Length: 167
- Authenticator: fe7d517dd3c3e337fa9f5fbf1f5d2a25
- [This is a response to a request in frame 57]
- [Time from request: 0.001539569 seconds]

□ Attribute Value Pairs

- + AVP: 1=58 t=Vendor-Specific(26) v=Microsoft(311)
- + AVP: 1=58 t=Vendor-Specific(26) v=Microsoft(311)
- + AVP: 1=6 t=EAP-Message(79) Last Segment[1]
- + AVP: 1=18 t=Message-Authenticator(80): c9a50d816a4d51522bdd6c1ea2ecd726
- + AVP: 1=7 t=User-Name(1): user1

3.5 OpenLDAP

OpenLDAP es una implementación libre y de código abierto del protocolo Lightweight Directory Access Protocol (LDAP) desarrollada por el proyecto OpenLDAP.

Está liberada bajo su propia licencia OpenLDAP Public License. LDAP es un protocolo de comunicación independiente de la plataforma.

OpenLDAP	
www.openldap.org	
	OpenLDAP™ http://www.OpenLDAP.org
Información general	
Desarrolladores	Proyecto OpenLDAP
Última versión estable	2.4.45 1 de junio de 2017
Género	LDAP
Programado en	C
Licencia	OpenLDAP Public License
Sistema Operativo	Multiplataforma

3.5.1 Archivos de configuración

El servicio de OpenLDAP funciona sobre una imagen de Docker que trae la configuración básica inicial. Los parámetros más importantes se pasan como argumentos en la definición de docker-compose, con los que se inicializa **slapd**.

Sin embargo, es importante destacar que para nuestro caso concreto, hay que realizar algunas modificaciones en la propia imagen:

- Sobrecarga del método **memberOf**. Para poder utilizar la característica de filtros en base a grupos, es imprescindible que el método *memberOf* sea sobrecargado para

usar identificadores únicos. La imagen utiliza por defecto *memberUniqueOf*, método que el servicio **Owncloud** no contempla.

- Para utilizar Asterisk con OpenLDAP, es necesario cargar el **schema** de Asterisk en la imagen, pues no lo trae por defecto. Otros esquemas como el de *RADIUS* vienen de serie.

Por tanto, únicamente necesitamos declarar nuestro esquema para que el servicio sea operativo.

De forma homóloga para cada universidad, el esquema es cómo el siguiente:

```
# Define dónde van a estar los usuarios
dn: ou=Personas,dc=um,dc=es
ou: Personas
objectclass: organizationalUnit

#Grupo de servicios
dn: ou=Servicios,dc=um,dc=es
ou: Servicios
objectclass: organizationalunit

#####
##### USUARIOS #####
#####
dn: uid=user1,ou=Personas,dc=um,dc=es
cn: Juan
sn: Perez
mail: user1@um.es
postOfficeBox: 2147483648
AstAccountContext: um
AstAccountNAT: no
AstAccountType: friend
AstAccountHost: dynamic
AstAccountCallerID: 251001
radiusFramedMTU: 999
objectClass: inetOrgPerson
objectClass: AsteriskSIPUser
objectClass: radiusprofile
objectclass: top
userPassword: password1

dn: uid=user2,ou=Personas,dc=um,dc=es
cn: Sofia
sn: Martinez
mail: user2@um.es
AstAccountContext: um
AstAccountNAT: no
AstAccountType: friend
AstAccountHost: dynamic
AstAccountCallerID: 251002
objectClass: inetOrgPerson
objectClass: AsteriskSIPUser
objectclass: top
userPassword: password2

dn: uid=user3,ou=Personas,dc=um,dc=es
cn: Leticia
sn: Gutierrez
```

```

mail: user3@um.es
AstAccountContext: um
AstAccountNAT: no
AstAccountType: friend
AstAccountHost: dynamic
AstAccountCallerID: 251003
objectClass: inetOrgPerson
objectClass: AsteriskSIPUser
objectclass: top
userPassword: password3

#####
##### SERVICIOS #####
#####

#Owncloud
# postOfficeBox = Disk quota size in bytes
dn: cn=owncloud,ou=Servicios,dc=um,dc=es
cn: owncloud
objectclass: groupOfNames
objectclass: top
member: uid=user1,ou=Personas,dc=um,dc=es
member: uid=user2,ou=Personas,dc=um,dc=es

# Radius
#### radiusprofile
# radiusFramedMTU
dn: cn=radius,ou=Servicios,dc=um,dc=es
cn: radius
objectclass: groupOfNames
objectclass: top
member: uid=user1,ou=Personas,dc=um,dc=es
member: uid=user2,ou=Personas,dc=um,dc=es

# Asterisk
#### AsteriskSIPUser
# AstAccountContext: CONTEXTO
# AstAccountNAT: NAT
# AstAccountType: friend
# AstAccountHost: dynamic
# AstAccountCallerID: NÚMERO
dn: cn=asterisk,ou=Servicios,dc=um,dc=es
cn: asterisk
objectclass: groupOfNames
objectclass: top
member: uid=user1,ou=Personas,dc=um,dc=es
member: uid=user2,ou=Personas,dc=um,dc=es

```

Este archivo de configuración tiene 3 partes esenciales:

1. La primera es la definición de unidades organizativas. Es decir, la base de nuestro **Direction Information Tree (DIT)**. Contemplamos dos entidades:
 - a. Personas
 - b. Servicios
2. La segunda parte se centra en crear los diversos usuarios que existen en la base de datos junto a sus atributos. Es importante remarcar que los diversos atributos

pertenecen a los objetos que declaramos. Estos atributos pueden ser obligatorios u opcionales

3. La última parte relaciona los servicios con los usuarios. Concretamente, define la pertenencia de un usuario (definido previamente) al servicio (entidad de la unidad organizativa *Servicios*).

3.5.2 Despliegue

El despliegue de este servicio, se define así en el docker-compose de forma homóloga para cada universidad:

```
um_opendap:  
    build: ./um/opendap/  
    command: --loglevel info  
    hostname: ldap  
    domainname: um.es  
    depends_on:  
        - "um_snmp"  
    environment:  
        - LDAP_ORGANISATION=Universidad de Murcia  
        - LDAP_DOMAIN=um.es  
        - LDAP_ADMIN_PASSWORD=um_password  
        - HOSTNAME=ldap.um.es #Bug  
        - LDAP_TLS=true  
        - LDAP_TLS_CRT_FILENAME=ldap-cert.pem  
        - LDAP_TLS_KEY_FILENAME=ldap-key.pem  
        - LDAP_TLS_CA_CRT_FILENAME=um.pem  
        - LDAP_TLS_ENFORCE=false  
    volumes:  
        - um-certs-ldap:/container/service/slapd/assets/certs  
    networks:  
        red1:  
            aliases:  
                - ldap.um.es  
    ipv4_address: 192.168.251.6
```

A diferencia de todos los demás servicios, esta imagen tiene la particularidad de lanzar su propio bootstrap. Además de las variables de entorno habituales para definir nuestra organización, hacemos uso de *command* para pasarle argumentos al bootstrap mencionado.

Además, definimos el uso del *LDAP* sobre *TLS*, aunque no lo forzamos. Esto es posible a través del volumen de Docker junto a las variables de entorno.

Por otra parte, el Dockerfile es el siguiente:

```
FROM vk496/opendap
```

```

LABEL maintainer="valiantsin.kivachuk@um.es"

ENV DEBIAN_FRONTEND noninteractive

#TODO ffix duplicated sourcelist
RUN echo "deb http://deb.debian.org/debian stretch main non-free contrib" >>
/etc/apt/sources.list && \
echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main " >>
/etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main " >>
/etc/apt/sources.list.d/apt-fast.list && \
    apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
    apt-get update && apt-get --no-install-recommends -y install apt-fast && apt-fast
install -y \
    net-tools \
    traceroute \
    netcat \
    dnsutils \
    iutils-ping \
    curl \
    nano \
    supervisor \
    snmpd snmp snmp-mibs-downloader \
&& rm -rf /var/lib/apt/lists/

COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf
COPY conf/snmpd.conf /etc/snmp/snmpd.conf

#Extend LDAP Schema
COPY conf/um.ldif /container/service/slapd/assets/config/bootstrap/ldif/custom/
RUN sed -i 's/.*mibs :/#/' /etc/snmp/snmp.conf #Enable all MIBS

CMD ["/usr/bin/supervisord"]

```

Al igual que el anterior servicio, este tiene 2 secciones principales: Descargar las dependencias, y trasladar todos los archivos de configuración al contenedor.

Por último, la configuración de Supervisord:

```

[supervisord]
nodaemon=true

[program:snmpd]
command=/etc/init.d/snmpd start
priority=100
startretries=0

```

Como podemos observar, el servicio no se lanza desde supervisor porque ya lo hace la propia imagen.

3.5.3 Trazas OpenLDAP

Para realizar diversas operaciones con el directorio activo, usaremos las herramientas que vienen con el paquete más comunes.

Para poder realizar consultas, es requisito autenticarse en las consultas

3.5.3.1 LDAPSEARCH

ldapsearch abre una conexión a un servidor LDAP, enlaza y realiza una búsqueda usando los parámetros especificados. El filtro debe ser conforme a la representación de cadena de filtros de búsqueda como se define en el RFC 4515. Si no se proporciona, el filtro predeterminado, (**objectClass = ***) , se utiliza.

Si *ldapsearch* encuentra una o más entradas, los atributos especificados por *attrs* se devuelven. Si * está en la lista, se devuelven todos los atributos de usuario. Si + aparece, se devuelven todos los atributos operativos. Si no hay *attrs*, se devuelven todos los atributos de usuario. Si está 1.1 está en la lista, no se devolverán los atributos.

Podemos efectuar la orden con el siguiente comando:

```
# ldapsearch -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password -b ou=Personas,dc=um,dc=es dn
# extended LDIF
#
# LDAPv3
# base <ou=Personas,dc=um,dc=es> with scope subtree
# filter: (objectclass=*)
# requesting: dn
#
# Personas, um.es
dn: ou=Personas,dc=um,dc=es

# user1, Personas, um.es
dn: uid=user1,ou=Personas,dc=um,dc=es

# user2, Personas, um.es
dn: uid=user2,ou=Personas,dc=um,dc=es

# user3, Personas, um.es
dn: uid=user3,ou=Personas,dc=um,dc=es

# search result
search: 2
result: 0 Success
```

```
# numResponses: 5
# numEntries: 4
```

En donde los distintos argumentos son:

- **-x:** Usar autenticación simple
- **-H:** LDAP URI
- **-D:** Base dn con la que se hará la consulta. Debe tener permisos suficientes para efectuar la operación
- **-w:** Contraseña para la autenticación simple
- **-b:** Base dn en la que se realizará la búsqueda
- **dn:** Filtro de la búsqueda

La traza relativa a este comando se puede ver a continuación gracias a Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000100763	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.017725236	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.017964195	192.168.251.1	192.168.251.6	LDAP	132	searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
9	0.018118817	192.168.251.6	192.168.252.6	LDAP	100	searchResEntry(2) "ou=Personas,dc=um,dc=es"
10	0.018196674	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es"
12	0.018238239	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user2,ou=Personas,dc=um,dc=es"
13	0.018292032	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user3,ou=Personas,dc=um,dc=es"
14	0.018312092	192.168.251.6	192.168.252.6	LDAP	80	searchResDone(2) success [4 results]
16	0.018362340	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 4: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 40720, Dst Port: 389, Seq: 1, Ack: 1, Len: 45
Lightweight Directory Access Protocol
LDAPMessage bindRequest(1) "cn=admin,dc=um,dc=es" simple
messageID: 1
protocolOp: bindRequest (0)
bindRequest
version: 3
name: cn=admin,dc=um,dc=es
authentication: simple (0)
simple: um_password

El proceso comienza con una petición bind para cambiar el estado de la autorización a la de “**cn=admin,dc=um,dc=es**”. Esto se realiza a través de la petición *bindRequest*. Cabe remarcar que la contraseña se transmite en texto plano.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000100763	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.017725236	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.017964195	192.168.251.1	192.168.251.6	LDAP	132	searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
9	0.018118817	192.168.251.6	192.168.252.6	LDAP	100	searchResEntry(2) "ou=Personas,dc=um,dc=es"
10	0.018196674	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es"
12	0.018238239	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user2,ou=Personas,dc=um,dc=es"
13	0.018292032	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user3,ou=Personas,dc=um,dc=es"
14	0.018312092	192.168.251.6	192.168.252.6	LDAP	80	searchResDone(2) success [4 results]
16	0.018362340	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 6: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 40720, Dst Port: 389, Seq: 1, Ack: 46, Len: 14
Lightweight Directory Access Protocol
LDAPMessage bindResponse(1) success
messageID: 1
protocolOp: bindResponse (1)
bindResponse
resultCode: success (0)
matchedDN:
errorMessage:

Recibimos la confirmación del servidor, quien nos indica que el cambio se ha realizado correctamente. A partir de este momento, las operaciones se realizan vía el *dn* solicitado.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000100763	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.017725236	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.017964195	192.168.251.1	192.168.251.6	LDAP	132	searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
9	0.018118817	192.168.251.6	192.168.252.6	LDAP	100	searchResEntry(2) "ou=Personas,dc=um,dc=es"
10	0.018196674	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es"
12	0.018238239	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user2,ou=Personas,dc=um,dc=es"
13	0.018292032	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user3,ou=Personas,dc=um,dc=es"
14	0.018312092	192.168.251.6	192.168.252.6	LDAP	80	searchResDone(2) success [4 results]
16	0.018362340	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 8: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 40720, Dst Port: 389, Seq: 46, Ack: 15, Len: 66
Lightweight Directory Access Protocol
    LDAPMessage searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
        messageID: 2
        protocolOp: searchRequest (3)
            searchRequest
                baseObject: ou=Personas,dc=um,dc=es
                scope: wholeSubtree (2)
                derefAliases: neverDerefAliases (0)
                sizeLimit: 0
                timeLimit: 0
                typesOnly: False
                filter: (objectclass=*)
                    filter: present (7)
                        present: objectclass
                attributes: 1 item
                    AttributeDescription: dn

```

A continuación, se realiza la operación real que hemos ejecutado a través de la línea de comandos. Como vemos, se trata de un **searchRequest** en **ou=Personas,dc=um,dc=es** donde se usará el filtro por defecto (**objectclass=***) y los atributos **dn**. Además, se establecen algunos parámetros por defecto, tales como:

- scope: Indica en qué parte del árbol se realizarán las búsquedas:
 - base. Únicamente en el nivel superior
 - one. Solo en un nivel
 - sub. Todos los niveles partiendo de la base. *Opción por defecto*
 - children. Todos los niveles excepto el nivel superior
- derefAliases: Indica si debemos desenlazar el alias de su entrada:
 - never. Nunca desenlazar. *Opción por defecto*.
 - always. Siempre desenlazar.
 - search. Desenlazar únicamente durante la resolución de nombre.
 - find. Desenlazar únicamente después de la resolución de nombre
- sizeLimit: Indica número máximo de entradas a devolver. Sin límite por defecto
- timeLimit: Indica segundos máximo de espera. Sin límite por defecto
- typesOnly: Devolver los atributos sin su valor. *Falso por defecto*

A continuación, el servidor nos envía una respuesta por cada entrada, como se puede ver a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000100763	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.017725236	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.017964195	192.168.251.1	192.168.251.6	LDAP	132	searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
9	0.018118817	192.168.251.6	192.168.252.6	LDAP	100	searchResEntry(2) "ou=Personas,dc=um,dc=es"
10	0.018196674	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es"
12	0.018238239	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user2,ou=Personas,dc=um,dc=es"
13	0.018292032	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user3,ou=Personas,dc=um,dc=es"
14	0.018312092	192.168.251.6	192.168.252.6	LDAP	80	searchResDone(2) success [4 results]
16	0.018362340	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 10: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 40720, Seq: 49, Ack: 112, Len: 44
Lightweight Directory Access Protocol
 LDAPMessage searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es" [4 results]
 messageID: 2
 protocolOp: searchResEntry (4)
 searchResEntry
 objectName: uid=user1,ou=Personas,dc=um,dc=es
 attributes: 0 items

Cuando se han devuelto todos los resultados, el servidor termina con un mensaje de *searchResDone*:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000100763	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.017725236	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.017964195	192.168.251.1	192.168.251.6	LDAP	132	searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
9	0.018118817	192.168.251.6	192.168.252.6	LDAP	100	searchResEntry(2) "ou=Personas,dc=um,dc=es"
10	0.018196674	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es"
12	0.018238239	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user2,ou=Personas,dc=um,dc=es"
13	0.018292032	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user3,ou=Personas,dc=um,dc=es"
14	0.018312092	192.168.251.6	192.168.252.6	LDAP	80	searchResDone(2) success [4 results]
16	0.018362340	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 14: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 40720, Seq: 181, Ack: 112, Len: 14
Lightweight Directory Access Protocol
 LDAPMessage searchResDone(2) success [4 results]
 messageID: 2
 protocolOp: searchResDone (5)
 searchResDone
 resultCode: success (0)
 matchedDN:
 errorMessage:

Por último, el cliente termina la conexión con un mensaje *unbindRequest*, que no recibe confirmación del servidor:

No.	Time	Source	Destination	Protocol	Length	Info
5	0.837444675	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
7	0.837601152	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
9	0.837693493	192.168.251.1	192.168.251.6	LDAP	132	searchRequest(2) "ou=Personas,dc=um,dc=es" wholeSubtree
10	0.837820464	192.168.251.6	192.168.252.6	LDAP	100	searchResEntry(2) "ou=Personas,dc=um,dc=es"
11	0.837858475	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user1,ou=Personas,dc=um,dc=es"
13	0.837888393	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user2,ou=Personas,dc=um,dc=es"
14	0.837914810	192.168.251.6	192.168.252.6	LDAP	110	searchResEntry(2) "uid=user3,ou=Personas,dc=um,dc=es"
15	0.837936578	192.168.251.6	192.168.252.6	LDAP	80	searchResDone(2) success [4 results]
17	0.837978310	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 17: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 40918, Dst Port: 389, Seq: 112, Ack: 195, Len: 7
Lightweight Directory Access Protocol
 LDAPMessage unbindRequest(3)
 messageID: 3
 protocolOp: unbindRequest (2)
 unbindRequest

3.5.3.2 LDAPADD

Idapmodify es una interfaz de shell accesible a llamadas de las bibliotecas. **Idapadd** se implementa como un enlace duro a la herramienta Idapmodify. Cuando se invoca como Idapadd, el argumento -a (añadir nueva entrada) se usa automáticamente.

ldapmodify abre una conexión a un servidor LDAP, enlaza y modifica o añade entradas. La información de entrada se lee de la entrada estándar o desde el archivo a través del uso de la -f opción.

Para añadir una entrada, usaremos el siguiente contenido en el archivo **my.ldif**:

```
dn: uid=user4,ou=Personas,dc=um,dc=es
cn: Juan
sn: Palomo
mail: user4@um.es
AstAccountContext: um
AstAccountNAT: no
AstAccountType: friend
AstAccountHost: dynamic
AstAccountCallerID: 251004
objectClass: inetOrgPerson
objectClass: AsteriskSIPUser
objectclass: top
userPassword: password4
```

Podemos efectuar la orden con el siguiente comando:

```
# ldapadd -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password -f my.ldif
adding new entry "uid=user4,ou=Personas,dc=um,dc=es"
```

En donde los distintos argumentos son:

- **-x**: Usar autenticación simple
- **-H**: LDAP URI
- **-D**: Base dn con la que se hará la consulta. Debe tener permisos suficientes para efectuar la operación
- **-w**: Contraseña para la autenticación simple
- **-f**: Archivo de entrada

La traza relativa a este comando se puede ver a continuación gracias a Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
172	51.997360969	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
174	51.997584046	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
176	51.997681426	192.168.251.1	192.168.251.6	LDAP	391	addRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
177	52.029173560	192.168.251.6	192.168.252.6	LDAP	80	addResponse(2) success
178	52.029481929	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 172: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface 0
 Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
 Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
 Transmission Control Protocol, Src Port: 42196, Dst Port: 389, Seq: 1, Ack: 1, Len: 45
 Lightweight Directory Access Protocol
 LDAPMessage bindRequest(1) "cn=admin,dc=um,dc=es" simple
 messageID: 1
 protocolOp: bindRequest (0)
 bindRequest
 version: 3
 name: cn=admin,dc=um,dc=es
 authentication: simple (0)
 simple: um_password

El proceso comienza con una petición bind para cambiar el estado de la autorización a la de “**cn=admin,dc=um,dc=es**”. Esto se realiza a través de la petición *bindRequest*. Cabe remarcar que la contraseña se transmite en texto plano.

No.	Time	Source	Destination	Protocol	Length	Info
172	51.997360969	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
174	51.997584046	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
176	51.997681426	192.168.251.1	192.168.251.6	LDAP	391	addRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
177	52.029173560	192.168.251.6	192.168.252.6	LDAP	80	addResponse(2) success
178	52.029481929	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 174: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
 Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
 Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
 Transmission Control Protocol, Src Port: 389, Dst Port: 42196, Seq: 1, Ack: 46, Len: 14
 Lightweight Directory Access Protocol
 LDAPMessage bindResponse(1) success
 messageID: 1
 protocolOp: bindResponse (1)
 bindResponse
 resultCode: success (0)
 matchedDN:
 errorMessage:

Recibimos la confirmación del servidor, quien nos indica que el cambio se ha realizado correctamente. A partir de este momento, las operaciones se realizan vía el *dn* solicitado.

No.	Time	Source	Destination	Protocol	Length	Info
172	51.997360969	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
174	51.997584046	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
176	51.997681426	192.168.251.1	192.168.251.6	LDAP	391	addRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
177	52.029173560	192.168.251.6	192.168.252.6	LDAP	80	addResponse(2) success
178	52.029481929	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

Frame 176: 391 bytes on wire (3128 bits), 391 bytes captured (3128 bits) on interface 0
 Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
 Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
 Transmission Control Protocol, Src Port: 42196, Dst Port: 389, Seq: 46, Ack: 15, Len: 325
 Lightweight Directory Access Protocol
 LDAPMessage addRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
 messageID: 2
 protocolOp: addRequest (8)
 addRequest
 entry: uid=user4,ou=Personas,dc=um,dc=es
 attributes: 10 items
 + AttributeList item cn
 + AttributeList item sn
 + AttributeList item mail
 + AttributeList item AstAccountContext
 + AttributeList item AstAccountNAT
 + AttributeList item AstAccountType
 + AttributeList item AstAccountHost
 + AttributeList item AstAccountCallerID
 + AttributeList item objectClass
 + AttributeList item userPassword

A continuación, se realiza la operación real que hemos ejecutado a través de la línea de comandos. Como vemos, se trata de un *addRequest* en **uid=4,ou=Personas,dc=um,dc=es**. En él se incluyen todos los elementos que hemos indicado en el archivo *my.ldif*

A continuación, el servidor nos envía la respuesta a nuestra solicitud:

No.	Time	Source	Destination	Protocol	Length	Info
172	51.997360969	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
174	51.997584046	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
176	51.997681426	192.168.251.1	192.168.251.6	LDAP	391	addRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
177	52.029173560	192.168.251.6	192.168.252.6	LDAP	80	addResponse(2) success
178	52.029481929	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 177: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 42196, Seq: 15, Ack: 371, Len: 14
Lightweight Directory Access Protocol
 LDAPMessage addResponse(2) success
    messageID: 2
    protocolOp: addResponse (9)
        addResponse
            resultCode: success (0)
            matchedDN:
            errorMessage:

```

En nuestro caso, la operación se ha realizado con éxito y recibimos la respuesta *success*.

Por último, el cliente termina la conexión con un mensaje *unbindRequest*, que no recibe confirmación del servidor:

No.	Time	Source	Destination	Protocol	Length	Info
172	51.997360969	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
174	51.997584046	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
176	51.997681426	192.168.251.1	192.168.251.6	LDAP	391	addRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
177	52.029173560	192.168.251.6	192.168.252.6	LDAP	80	addResponse(2) success
178	52.029481929	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 178: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 42196, Dst Port: 389, Seq: 371, Ack: 29, Len: 7
Lightweight Directory Access Protocol
 LDAPMessage unbindRequest(3)
    messageID: 3
    protocolOp: unbindRequest (2)
        unbindRequest

```

Podemos verificar que todos los campos se han agregado correctamente con el siguiente comando:

```

# ldapsearch -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password -b "ou=Personas,dc=um,dc=es" "uid=user4"
# extended LDIF
#
# LDAPv3
# base <ou=Personas,dc=um,dc=es> with scope subtree
# filter: uid=user4
# requesting: ALL
#
# user4, Personas, um.es

```

```

dn: uid=user4,ou=Personas,dc=um,dc=es
cn: Juan
sn: Palomo
mail: user4@um.es
AstAccountContext: um
AstAccountNAT: no
AstAccountType: friend
AstAccountHost: dynamic
AstAccountCallerID: 251004
objectClass: inetOrgPerson
objectClass: AsteriskSIPUser
objectClass: top
userPassword:: cGFzc3dvcmQ0
uid: user4

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1

```

3.5.3.3 LDAPMODIFY

Como ya hemos comentado, `ldapadd` no es más que `ldapmodify` con un parámetro subyacente (`-a`).

Para realizar esta prueba, vamos a modificar la entrada que hemos agregado previamente. Para ello, partimos del siguiente archivo `ldiff my.ldiff`:

```

dn: uid=user4,ou=Personas,dc=um,dc=es
changetype: modify
replace: cn
cn: Alfonso

```

Ejecutamos la siguiente orden:

```

# ldapmodify -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password -f my.ldiff
modifying entry "uid=user4,ou=Personas,dc=um,dc=es"

```

En donde los distintos argumentos son:

- `-x`: Usar autenticación simple
- `-H`: LDAP URI

-
- **-D:** Base dn con la que se hará la consulta. Debe tener permisos suficientes para efectuar la operación
 - **-w:** Contraseña para la autenticación simple
 - **-f:** Archivo de entrada

La traza relativa a este comando se puede ver a continuación gracias a Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000072121	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.000259086	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.000339341	192.168.251.1	192.168.251.6	LDAP	132	modifyRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
9	0.018999250	192.168.251.6	192.168.252.6	LDAP	80	modifyResponse(2) success
10	0.019107816	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 4: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 45164, Dst Port: 389, Seq: 1, Ack: 1, Len: 45
Lightweight Directory Access Protocol
  LDAPMessage bindRequest(1) "cn=admin,dc=um,dc=es" simple
    messageID: 1
  protocolOp: bindRequest (0)
    bindRequest
      version: 3
      name: cn=admin,dc=um,dc=es
    authentication: simple (0)
      simple: um_password

```

El proceso comienza con una petición bind para cambiar el estado de la autorización a la de “**cn=admin,dc=um,dc=es**”. Esto se realiza a través de la petición *bindRequest*. Cabe remarcar que la contraseña se transmite en texto plano.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000072121	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.000259086	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.000339341	192.168.251.1	192.168.251.6	LDAP	132	modifyRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
9	0.018999250	192.168.251.6	192.168.252.6	LDAP	80	modifyResponse(2) success
10	0.019107816	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 6: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 45164, Seq: 1, Ack: 46, Len: 14
Lightweight Directory Access Protocol
  LDAPMessage bindResponse(1) success
    messageID: 1
  protocolOp: bindResponse (1)
    bindResponse
      resultCode: success (0)
      matchedDN:
      errorMessage:

```

Recibimos la confirmación del servidor, quien nos indica que el cambio se ha realizado correctamente. A partir de este momento, las operaciones se realizan vía el *dn* solicitado.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000072121	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.000259086	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.000339341	192.168.251.1	192.168.251.6	LDAP	132	modifyRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
9	0.018999250	192.168.251.6	192.168.252.6	LDAP	80	modifyResponse(2) success
10	0.019107816	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 8: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 45164, Dst Port: 389, Seq: 46, Ack: 15, Len: 66
Lightweight Directory Access Protocol
  LDAPMessage modifyRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
    messageID: 2
    protocolOp: modifyRequest (6)
      modifyRequest
        object: uid=user4,ou=Personas,dc=um,dc=es
        modification: 1 item
          modification item
            operation: replace (2)
            modification cn
              type: cn
              vals: 1 item
                AttributeValue: Alfonso

```

Después, se realiza la operación real que hemos ejecutado a través de la línea de comandos. Como vemos, se trata de un *modifyRequest* en **uid=4,ou=Personas,dc=um,dc=es**. De forma jerárquica, se indica que se va a realizar una sustitución en el atributo *cn*, siendo su nuevo valor *Alfonso*

A continuación, el servidor nos envía la respuesta a nuestra solicitud:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000072121	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.000259086	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.000339341	192.168.251.1	192.168.251.6	LDAP	132	modifyRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
9	0.018999250	192.168.251.6	192.168.252.6	LDAP	80	modifyResponse(2) success
10	0.019107816	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 9: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 45164, Seq: 15, Ack: 112, Len: 14
Lightweight Directory Access Protocol
  LDAPMessage modifyResponse(2) success
    messageID: 2
    protocolOp: modifyResponse (7)
      modifyResponse
        resultCode: success (0)
        matchedDN:
        errorMessage:

```

En nuestro caso, la operación se ha realizado con éxito y recibimos la respuesta *success*.

Por último, el cliente termina la conexión con un mensaje *unbindRequest*, que no recibe confirmación del servidor:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000072121	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
6	0.000259086	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
8	0.000339341	192.168.251.1	192.168.251.6	LDAP	132	modifyRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
9	0.018999250	192.168.251.6	192.168.252.6	LDAP	80	modifyResponse(2) success
10	0.019107816	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 10: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 45164, Dst Port: 389, Seq: 112, Ack: 29, Len: 7
Lightweight Directory Access Protocol
  LDAPMessage unbindRequest(3)
    messageID: 3
    protocolOp: unbindRequest (2)
      unbindRequest

```

Podemos verificar que el campo se ha modificado de forma satisfactoria con el siguiente comando:

```
# ldapsearch -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password -b "ou=Personas,dc=um,dc=es" "uid=user4" "cn"
# extended LDIF
#
# LDAPv3
# base <ou=Personas,dc=um,dc=es> with scope subtree
# filter: uid=user4
# requesting: cn
#
# user4, Personas, um.es
dn: uid=user4,ou=Personas,dc=um,dc=es
cn: Alfonso

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

3.5.3.4 LDAPDELETE

ldapdelete abre una conexión a un servidor LDAP, enlaza y borra una o más entradas. Si uno o más DN se proporcionan como argumentos, dichas entradas se eliminan.

Cada DN deberá proporcionarse utilizando la representación de cadena LDAPv3 como se define en el RFC 4514. Si no se proporcionan DNs como argumentos, una lista de DN se lee de la entrada estándar (o del archivo si el -f se utilizara).

Podemos efectuar la orden con el siguiente comando:

```
# ldapdelete -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password uid=user4,ou=Personas,dc=um,dc=es
```

En donde los distintos argumentos son:

- **-x:** Usar autenticación simple
- **-H:** LDAP URI
- **-D:** Base dn con la que se hará la consulta. Debe tener permisos suficientes para efectuar la operación
- **-w:** Contraseña para la autenticación simple

- **uid=user4,ou=Personas,dc=um,dc=es**: Entrada a eliminar

La traza relativa a este comando se puede ver a continuación gracias a Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
6	22.470131493	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
8	22.470336739	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
10	22.470404031	192.168.251.1	192.168.251.6	LDAP	106	delRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
11	22.489949418	192.168.251.6	192.168.252.6	LDAP	80	delResponse(2) success
12	22.490034758	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 6: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 45754, Dst Port: 389, Seq: 1, Ack: 1, Len: 45
Lightweight Directory Access Protocol
  LDAPMessage bindRequest(1) "cn=admin,dc=um,dc=es" simple
    messageID: 1
    protocolOp: bindRequest (0)
      bindRequest
        version: 3
        name: cn=admin,dc=um,dc=es
      authentication: simple (0)
        simple: um_password

```

El proceso comienza con una petición bind para cambiar el estado de la autorización a la de “**cn=admin,dc=um,dc=es**”. Esto se realiza a través de la petición *bindRequest*. Cabe remarcar que la contraseña se transmite en texto plano.

No.	Time	Source	Destination	Protocol	Length	Info
6	22.470131493	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
8	22.470336739	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
10	22.470404031	192.168.251.1	192.168.251.6	LDAP	106	delRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
11	22.489949418	192.168.251.6	192.168.252.6	LDAP	80	delResponse(2) success
12	22.490034758	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 8: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 45754, Seq: 1, Ack: 46, Len: 14
Lightweight Directory Access Protocol
  LDAPMessage bindResponse(1) success
    messageID: 1
    protocolOp: bindResponse (1)
      bindResponse
        resultCode: success (0)
        matchedDN:
        errorMessage:

```

Recibimos la confirmación del servidor, quien nos indica que el cambio se ha realizado correctamente. A partir de este momento, las operaciones se realizan vía el *dn* solicitado.

No.	Time	Source	Destination	Protocol	Length	Info
6	22.470131493	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
8	22.470336739	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
10	22.470404031	192.168.251.1	192.168.251.6	LDAP	106	delRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
11	22.489949418	192.168.251.6	192.168.252.6	LDAP	80	delResponse(2) success
12	22.490034758	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 10: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 45754, Dst Port: 389, Seq: 46, Ack: 15, Len: 40
Lightweight Directory Access Protocol
  LDAPMessage delRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
    messageID: 2
    protocolOp: delRequest (10)
      delRequest: uid=user4,ou=Personas,dc=um,dc=es

```

A continuación, se realiza la operación real que hemos ejecutado a través de la línea de comandos. Como vemos, se trata de un `delRequest` en `ou=Personas,dc=um,dc=es` de la entrada `uid=user4`.

A continuación, el servidor nos envía la respuesta a nuestra solicitud:

No.	Time	Source	Destination	Protocol	Length	Info
6	22.470131493	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
8	22.470336739	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
10	22.470404031	192.168.251.1	192.168.251.6	LDAP	106	delRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
11	22.489949418	192.168.251.6	192.168.252.6	LDAP	80	delResponse(2) success
12	22.490034758	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 11: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06), Dst: 02:42:90:3a:60:70 (02:42:90:3a:60:70)
Internet Protocol Version 4, Src: 192.168.251.6, Dst: 192.168.252.6
Transmission Control Protocol, Src Port: 389, Dst Port: 45754, Seq: 15, Ack: 86, Len: 14
Lightweight Directory Access Protocol
    LDAPMessage delResponse(2) success
        messageID: 2
        protocolOp: delResponse (11)
            delResponse
                resultCode: success (0)
                matchedDN:
                errorMessage:

```

En nuestra caso, la operación se ha realizado con éxito y recibimos la respuesta `success`.

Por último, el cliente termina la conexión con un mensaje `unbindRequest`, que no recibe confirmación del servidor:

No.	Time	Source	Destination	Protocol	Length	Info
6	22.470131493	192.168.251.1	192.168.251.6	LDAP	111	bindRequest(1) "cn=admin,dc=um,dc=es" simple
8	22.470336739	192.168.251.6	192.168.252.6	LDAP	80	bindResponse(1) success
10	22.470404031	192.168.251.1	192.168.251.6	LDAP	106	delRequest(2) "uid=user4,ou=Personas,dc=um,dc=es"
11	22.489949418	192.168.251.6	192.168.252.6	LDAP	80	delResponse(2) success
12	22.490034758	192.168.251.1	192.168.251.6	LDAP	73	unbindRequest(3)

```

Frame 12: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: 02:42:90:3a:60:70 (02:42:90:3a:60:70), Dst: 02:42:c0:a8:fb:06 (02:42:c0:a8:fb:06)
Internet Protocol Version 4, Src: 192.168.251.1, Dst: 192.168.251.6
Transmission Control Protocol, Src Port: 45754, Dst Port: 389, Seq: 86, Ack: 29, Len: 7
Lightweight Directory Access Protocol
    LDAPMessage unbindRequest(3)
        messageID: 3
        protocolOp: unbindRequest (2)
            unbindRequest

```

Podemos verificar que la entrada se ha eliminado con el siguiente comando:

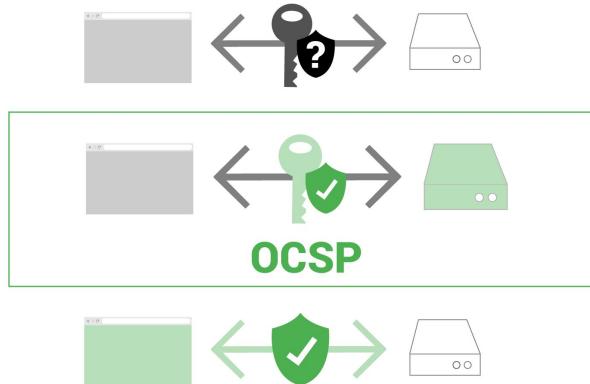
```

# ldapsearch -x -H ldap://192.168.251.6 -D "cn=admin,dc=um,dc=es" -w
um_password -b "ou=Personas,dc=um,dc=es" "uid=user4"
# extended LDIF
#
# LDAPv3
# base <ou=Personas,dc=um,dc=es> with scope subtree
# filter: uid=user4
# requesting: ALL
#
# search result
search: 2
result: 0 Success

```

```
# numResponses: 1
```

3.6 OCSP



Protocolo de comprobación del Estado de un Certificado En línea u Online Certificate Status Protocol (**OCSP**) es un método para determinar el estado de vigencia de un certificado digital X.509 usando otros medios que no sean el uso de *CRL* (Listas de Revocación de Certificados). Este protocolo se describe en el [RFC 6960](#) y está en el registro de estándares de Internet.

Los mensajes OCSP se codifican en *ASN.1* y habitualmente se transmiten sobre el protocolo HTTP. La naturaleza de las peticiones y respuestas de OCSP hace que a los servidores OCSP se les conozca como "*OCSP responders*".

3.6.1 Archivos de configuración

Para desplegar el servicio OCSP, necesitamos 3 elementos:

- Un servidor que responda a las solicitudes OCSP. Existen diversos proyectos para llevar esto a cabo (*openCA*, *cfssl*, etc.). En nuestro caso, delegaremos directamente en la herramienta **openssl**.
- Un certificado emitido por el **CA** que tenga la extensión OCSP habilitada. Concretamente, se trata de **X509v3 Extended Key Usage: OCSP Signing**.
- La base de datos de todos los certificados expedidos por el CA. De lo contrario, OCSP devolverá el estado **unknown** (Otras herramientas permiten emitir la respuesta **good** en caso de que el certificado no se encuentre en la base de datos).

Para expedir un certificado capaz de emitir respuestas OCSP, debemos emitirlo con la siguiente extensión:

```
[ v3_OCSP ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = OCSPSigning
```

Además, publicaremos en el puerto 80 la CRL a través de un servidor web.

3.6.2 Despliegue

El despliegue de este servicio, se define así en el docker-compose de forma homóloga para cada universidad:

```
um_distribution:
  build: ./um/distribution/
  hostname: distribution
  domainname: um.es
  volumes:
    - lego_ca:/web
    - um-certs-distribution:/certs
  networks:
    red1:
      aliases:
        - distribution.um.es
  ipv4_address: 192.168.251.99
```

La definición es similar a los demás servicios. Destacar que en este caso tenemos dos volúmenes: uno con la CRL de la organización y otro con los certificados del OCSP Responder.

Por otra parte, el Dockerfile es el siguiente:

```
FROM ubuntu:16.04
LABEL maintainer="valiantsin.kivachuk@um.es"

#ORG1-N1

RUN echo "deb http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >>
/etc/apt/sources.list.d/apt-fast.list && \
    echo "deb-src http://ppa.launchpad.net/saiarcot895/myppa/ubuntu xenial main" >>
/etc/apt/sources.list.d/apt-fast.list && \
    apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys DC058F40 && \
    apt-get update && apt-get --no-install-recommends -y install apt-fast && apt-fast
install -y \
    net-tools \
    traceroute \
    netcat \
    dnsutils \
    iputils-ping \
    curl \
    nano \
    ifupdown2 \
    supervisor \
```

```

    snmpd snmp snmp-mibs-downloader \
    openssl \
    python \
&& rm -rf /var/lib/apt/lists/*
COPY misc/supervisord.conf /etc/supervisor/conf.d/supervisord.conf
RUN sed -i 's/.*mibs :/#&/' /etc/snmp/snmp.conf #Enable all MIBS
COPY conf/snmpd.conf /etc/snmp/snmpd.conf
CMD ["/usr/bin/supervisord"]

```

Este tiene 2 secciones principales: Descargar las dependencias, y trasladar todos los archivos de configuración al contenedor.

Por último, la configuración de Supervisord:

```

[supervisord]
nodaemon=true

[program:snmpd]
command=/etc/init.d/snmpd start
priority=100
startretries=0

[program:crl]
directory=/web
command=python -m SimpleHTTPServer 80
priority=100
startretries=0

[program:ocsp]
directory=/certs
command=openssl ocsp -port 0.0.0.0:8080 -text -sha256 -CA upm.pem -rkey
distribution-key.pem -rsigner distribution-cert.pem -index .db.pem
-nrequest 30
priority=100
startretries=0

```

En este caso, desplegamos dos procesos principales: Un servidor web para la CRL y el OCSP Responder.

3.6.3 Trazas OCSP

Para poner en prueba la verificación de un certificado con el OCSP Responder, usaremos el host de VoIP. Para ello, entramos en el host y ejecutamos la siguiente orden:

```
$ openssl ocsp -CAfile /certs/um.pem -url http://distribution.um.es:8080 -resp_text  
-issuer /certs/um.pem -cert /certs/voip-cert.pem
```

Donde los parámetros son:

- **-CAfile.** Certificado del CA debido a que no lo tenemos instalado en el sistema
- **-url.** Url del OCSP Responder
- **-resp_text.** Mostrar información legible acerca del proceso
- **-issuer.** Certificado del CA que ha emitido el certificado que vamos a comprobar
- **-cert.** Certificado que vamos a comprobar

A lo que obtenemos la siguiente salida en el terminal:

```
OCSP Response Data:  
    OCSP Response Status: successful (0x0)  
    Response Type: Basic OCSP Response  
    Version: 1 (0x0)  
    Responder Id:  
        Produced At: Dec 17 19:01:33 2017 GMT  
    Responses:  
        Certificate ID:  
            Hash Algorithm: sha1  
            Issuer Name Hash: DC8838BD5E0A7C34DBE1EE691012B6C3FB741FA6  
            Issuer Key Hash: 890F04586ED954BA13FE0ED2503C281F9B07CC5C  
            Serial Number: E3FBA1F30FF800EA  
            Cert Status: good  
            This Update: Dec 17 19:01:33 2017 GMT  
  
    Response Extensions:  
    OCSP Nonce:  
        04109D8E0779F7DDEC2130FF847D82CD3E9B  
    Signature Algorithm: sha256WithRSAEncryption  
    6a:72:c2:d8:3b:90:d8:c5:f6:30:8a:3f:37:cc:16:bd:54:40:  
    b9:2a:2a:a3:c0:13:1c:b7:93:c4:80:79:3c:fc:d2:ab:f9:65:  
    61:5b:ae:84:52:f2:7e:2b:11:79:71:4c:d7:de:81:62:1a:0d:  
    54:79:ff:87:f1:dc:0f:f1:52:f5:20:c1:dd:7c:51:60:1a:a4:  
    52:9c:87:0e:99:12:87:5c:ec:19:b4:45:76:f3:89:c7:e2:02:  
    5a:4a:e9:41:cb:f0:eb:03:fe:41:72:1a:c3:71:7d:f5:6b:36:  
    0e:d4:e8:b1:37:9d:9c:8b:bb:b7:34:81:88:1c:91:9a:dc:d6:  
    ee:40:93:13:30:73:af:11:a6:81:ee:de:58:68:27:21:70:f4:  
    a0:91:23:e4:59:28:b1:2e:cc:54:f0:e2:62:fb:32:a7:a2:74:  
    c2:90:7d:7c:da:e4:ac:d3:ac:dd:4a:26:36:57:2d:46:da:9d:  
    8e:e1:ea:0a:b4:76:31:7b:09:19:1f:2a:65:e0:d8:76:f5:99:  
    56:4f:e4:5e:d3:26:53:56:52:c9:62:3d:41:b2:c1:31:ab:4b:  
    a8:06:30:91:1a:f9:ef:c8:4d:ea:89:bc:1c:73:a0:dc:7f:24:  
    b7:13:45:17:94:c8:82:1b:f5:96:e6:7b:72:dd:bf:a6:c4:57:  
    d8:4b:2f:d3
```

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 16427902131081052392 (0xe3fba1f30ff800e8)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=um
  Validity
    Not Before: Dec 17 16:38:47 2017 GMT
    Not After : Dec 12 16:38:47 2018 GMT
  Subject:
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
        Modulus:
          00:c9:a2:7c:c3:fb:72:59:0a:8b:fa:45:b1:48:2d:
          e8:e4:38:65:39:ed:96:3f:78:8a:40:c0:22:88:a9:
          59:79:b2:4f:67:95:a5:22:26:fb:eb:dd:f5:22:e2:
          75:a6:ab:8e:a7:11:81:ac:45:da:b4:b7:97:4d:44:
          df:f2:49:60:f2:16:e1:82:53:d9:ad:4f:98:f3:4b:
          f9:df:a8:44:00:99:af:12:e8:32:e1:5f:9f:e8:6c:
          cb:bd:a6:03:90:ac:ac:39:dd:f7:9f:8a:6b:e3:4a:
          5c:f8:70:f2:ef:1c:f0:61:ea:c1:ec:04:2e:ee:35:
          84:97:75:48:ed:e9:88:76:bf:f9:65:8a:8e:fb:b7:
          85:41:e8:b5:88:b7:e0:42:68:c1:cd:31:30:22:41:
          41:12:14:b4:de:53:ff:23:7a:eb:85:22:07:44:40:
          1d:25:c3:a8:47:f9:d8:f5:80:76:9e:15:9f:a2:0b:
          c3:bb:07:e0:20:f4:fe:8e:e9:6b:53:56:78:c3:55:
          09:31:ae:f0:b5:c4:64:07:58:a6:c7:36:4a:5f:ea:
          83:08:f7:4a:f6:ea:d4:ec:d5:54:33:a4:ac:c8:eb:
          bd:5e:52:ba:fd:57:c1:f5:01:95:8f:a7:bd:bc:68:
          65:16:9e:99:4f:1a:aa:d0:ac:e5:67:82:44:2f:34:
          eb:01
        Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    X509v3 Key Usage:
      Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
      OCSP Signing
  Signature Algorithm: sha256WithRSAEncryption
  4a:98:67:71:69:0e:4d:4f:d6:65:b0:b8:7e:4d:d0:be:37:b9:
  90:cb:1d:4b:23:29:bc:15:ed:bf:72:63:af:2a:fc:28:e5:a7:
  69:70:f0:5d:85:ad:cf:a0:95:19:fc:f9:f1:d7:9e:6e:09:7f:
  07:8f:d2:54:5e:55:3d:20:0b:9c:ff:7e:15:2a:48:d7:4c:d4:
  33:c7:43:1f:2e:91:c6:89:22:e9:e2:8a:70:1b:be:3b:b1:5d:
  4b:3c:7d:55:c3:0e:61:0d:d8:ef:2f:40:f2:d7:18:a3:bb:de:
  5a:20:a6:54:a4:97:c9:1f:ec:21:c5:86:81:aa:0f:f5:0f:40:
  45:64:a1:dd:dc:6e:af:32:3c:47:f1:c6:e5:76:47:83:ed:8c:
  95:a0:f4:3b:6e:3c:d6:71:6b:25:b6:59:ce:e7:8c:1a:7b:6f:

```

```

d4:65:be:38:74:db:88:db:74:06:93:f1:41:4c:eb:9f:7d:5f:
c5:25:62:b1:6c:99:58:0b:7f:21:dc:85:8a:9b:68:5b:75:65:
d2:75:ef:5d:75:f7:12:ed:0e:75:1b:a1:3b:1d:c5:41:05:18:
45:d5:50:8f:a3:ff:bf:93:51:6e:62:7f:1c:e5:7b:8b:96:10:
be:77:f1:9e:22:97:24:cf:a1:9b:6d:3c:d8:ed:7b:d3:1a:f6:
19:30:e0:7d:08:74:f2:1d:40:5d:e1:e4:d6:8a:87:aa:b5:e5:
72:7f:e1:9c:55:09:45:2a:49:0e:28:3e:4e:51:b5:9c:75:12:
fd:9c:a0:6c:33:ce:36:94:9b:c9:a7:bc:39:10:89:63:7d:29:
d2:67:03:8c:d1:be:85:b5:70:3c:1b:5e:9b:a0:7c:b6:e1:66:
1e:6a:26:83:15:c7:64:cb:2b:97:e1:4c:63:17:9c:96:12:40:
5e:2d:86:a4:c2:01:44:76:59:ff:c2:83:31:12:72:a6:d7:aa:
19:b6:28:67:88:02:4d:d5:80:04:f2:ef:28:f1:df:42:82:70:
2f:04:3e:21:7a:8c:8f:f6:91:49:f4:eb:2e:f8:9b:1b:b8:e0:
95:ba:9d:12:4d:b4:4e:0f:d6:6e:f2:03:8e:9f:58:6e:45:e7:
78:85:96:46:31:9d:93:7b:b7:ec:ca:63:5a:2e:85:4f:49:d5:
41:12:e0:4e:75:f5:16:4d:b8:15:1b:33:f8:78:2e:ca:46:3c:
0d:1a:72:8c:df:68:e9:bb:b4:53:80:fb:47:6d:e1:14:09:43:
ae:20:47:46:7b:1c:2c:ee:a0:0a:66:63:4a:20:9e:a8:2b:22:
ed:72:ea:2f:e9:f7:58:48:b4:97:7f:90:ec:d0:31:39:a0:a2:
22:59:76:29:f2:cc:f7:e5
-----BEGIN CERTIFICATE-----
MIIDVzCCAaegAwIBAgIJAOP7ofMP+ADoMA0GCSqGSIB3DQEBCwUAMA0xCzAJBgNV
BAMMAAnVtMB4XDTE3MTIxNzE2Mzg0N1oXDTE4MTIxMjE2Mzg0N1owADCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAMMifMP7c1kKi/pFsUgt60Q4ZTnt1j94
ikDAIoipWxmyT2eVpSI++vd9SLidaarjqcRgaxF2rS3101E3/JJYPIW4YJT2a1P
mPNL+d+oRACZrxLoMuFFn+hsy72mA5CsrDnd95+Ka+NKXPhw8u8c8GHqwewELu41
hJd1SO3piHa/+WWKjvu3hUHotYi34EJowc0xMCJBQRIUtN5T/yN664UiB0RAHSXD
qEf52PWAdp4Vn6ILw7sH4CD0/o7pa1NWeMNVCTGu8LXEZAdYpsc2S1/qgwj3Svbq
10zVVVD0krMjrvV5Suv1XwfUB1Y+nvbxoZRaemU8aqtCs5WeCRC806wECAwEAAaMv
MC0wCQYDVR0TBAlwADALBgNVHQ8EBAMCBeAwEwYDVR01BAwwCgYIKwYBBQUHAwkW
DQYJKoZIhvcNAQELBQADggIBAEqYZ3FpDk1P1mlwuH5N0L43uZDLHUsjKbwV7b9y
Y68q/Cjlp2lW8F2Frc+glRn8+fHXnm4JfweP01ReVT0gC5z/fhUqSNdM1DPHQx8u
kcaJIuniinAbvjuxXUs8fVXDDmEN208vQPLXGK073logp1Sk18kf7CHFhoGqD/UP
QEvkod3cbq8yPEfxvuV2R4PtjJWg9DtUPNzayW2Wc7njBp7b9R1vjh024jbdAaT
8UFM6599X8UlYrFsmVgLfYHchYqbaFt1ZdJ171119xLtDnUboTsdxUEFGEXVUI+j
/7+TUW5ifxzle4uWEI538Z4ilyTPoZttPNjte9Ma9hkw4H0IdPIdQF3h5NaKh6q1
5XJ/4ZxVCUUqSQ4oPk5Rtzx1Ev2coGwzzjaUm8mnvDkQiWN9KdJnA4zRvoW1cDwb
XpugfLbhZh5qJoMvx2TLK5fhTGMXnJYSQF4thqTCAUR2Wf/CgzEScqbxqhm2KGel
Ak3VgATy7yjx30KCcC8EPiF6jI/2kUn06y74mxu44JW6nRJNtE4P1m7yA46fWG5F
53iF1kYxnZN7t+zKY1ouhU9J1UES4E519RZNuBUbM/h4LspGPA0acozfa0m7tFOA
+0dt4RQJQ64gR0Z7HCzuApmY0ognqgrIu1y6i/p91hItJd/kOzQMTmgoijZdiny
zPf1
-----END CERTIFICATE-----
Response verify OK
/certs/voip-cert.pem: good
This Update: Dec 17 19:01:33 2017 GMT

```

Podemos observar que recibimos una respuesta satisfactoria por parte del OCSP Responder. Analizando las trazas, podemos comprobar que es un protocolo muy simple:

No.	Time	Source	Destination	Protocol	Length	Info
→ 8	8.975929682	192.168.251.2	192.168.251.99	OCSP	259	Request
← 10	8.993740362	192.168.251.99	192.168.251.2	OCSP	1581	Response


```
+ Frame 8: 259 bytes on wire (2072 bits), 259 bytes captured (2072 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:02 (02:42:c0:a8:fb:02), Dst: 02:42:c0:a8:fb:63 (02:42:c0:a8:fb:63)
+ Internet Protocol Version 4, Src: 192.168.251.2, Dst: 192.168.251.99
+ Transmission Control Protocol, Src Port: 44526, Dst Port: 8080, Seq: 1, Ack: 1, Len: 193
+ Hypertext Transfer Protocol
| Online Certificate Status Protocol
|   tbsRequest
|     requestList: 1 item
|       Request
|         reqCert
|           hashAlgorithm (SHA-1)
|             issuerNameHash: dc8838bd5e0a7c34dbe1ee691012b6c3fb741fa6
|             issuerKeyHash: 890f04586ed954ba13fe0ed2503c281f9b07cc5c
|             serialNumber: 16427902131081052394
|   requestExtensions: 1 item
|     Extension
|       Id: 1.3.6.1.5.5.7.48.1.2 (id-pkix-ocsp-nonce)
|       ReOcspNonce: 9d8e0779f7ddec2130ff847d82cd3e9b
```

La solicitud realizada por el cliente. En ella nos encontramos con el certificado (y CA(s)) que estamos solicitando y un atributo **nonce**, para evitar ataques de réplica.

El servidor nos responde con:

No.	Time	Source	Destination	Protocol	Length	Info
→ 8	8.975929682	192.168.251.2	192.168.251.99	OCSP	259	Request
← 10	8.993740362	192.168.251.99	192.168.251.2	OCSP	1581	Response


```
+ Frame 10: 1581 bytes on wire (12648 bits), 1581 bytes captured (12648 bits) on interface 0
+ Ethernet II, Src: 02:42:c0:a8:fb:63 (02:42:c0:a8:fb:63), Dst: 02:42:c0:a8:fb:02 (02:42:c0:a8:fb:02)
+ Internet Protocol Version 4, Src: 192.168.251.99, Dst: 192.168.251.2
+ Transmission Control Protocol, Src Port: 8080, Dst Port: 44526, Seq: 1, Ack: 194, Len: 1515
+ Hypertext Transfer Protocol
| Online Certificate Status Protocol
|   responseStatus: successful (0)
|   responseBytes
|     ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
|       BasicOCSPResponse
|         tbsResponseData
|           responderID: byName (1)
|             producedAt: 2017-12-17 19:01:33 (UTC)
|           responses: 1 item
|             SingleResponse
|               certID
|                 hashAlgorithm (SHA-1)
|                   issuerNameHash: dc8838bd5e0a7c34dbe1ee691012b6c3fb741fa6
|                   issuerKeyHash: 890f04586ed954ba13fe0ed2503c281f9b07cc5c
|                   serialNumber: 16427902131081052394
|                 certStatus: good (0)
|                   thisUpdate: 2017-12-17 19:01:33 (UTC)
|             responseExtensions: 1 item
|               Extension
|                 Id: 1.3.6.1.5.5.7.48.1.2 (id-pkix-ocsp-nonce)
|                 ReOcspNonce: 9d8e0779f7ddec2130ff847d82cd3e9b
|             signatureAlgorithm (sha256WithRSAEncryption)
|               Padding: 0
|               signature: 6a72c2d83b90d8c5f6308a3f37cc16bd5440b92a2aa3c013...
|             certs: 1 item
```

En ella incluye la información que hemos solicitado con su respuesta (good) junto a otros campos que validan su autoridad.

4 Mejoras pendientes

A continuación se relatará una serie de mejoras que estaban previstas para el proyecto, pero por cuestiones de tiempo y carga de trabajo, no he podido realizar:

1. Migrar a DNS en Docker

Durante el desarrollo de la práctica, hubo algunos problemas con la resolución de dominios. Por un lado, Docker tiene un DNS embebido en su SDN, que permite resolver la dirección IP de los hosts mediante *domainname*, *hostname* y *alias*.

```
um_voip: #VoIP
  build: ./um/voip/
  hostname: voip
  domainname: um.es
  depends_on:
    - "um_snmp"
  volumes:
    - um-certs-voip:/certs
    - lego_ca:/ca_ssl
  networks:
    red1:
      aliases:
        - voip.um.es
      ipv4_address: 192.168.251.2
```

Sin embargo, ha habido una serie de impedimentos para que esto se adapte completamente al proyecto.

Por un lado, las redes virtuales que se crean dentro del proyecto (red1 y red2), están en modo bridge con la interfaz de red principal (salida a Internet). El comportamiento por defecto, es que un nombre de dominio se intenta resolver hacia Internet. Y si eso no es posible, enruta con los hosts del proyecto.

De esta forma, el dominio **ldap.upm.es** EXISTE en internet, por lo que a veces parecía haber un comportamiento arbitrario para resolver la IP.

Por otro lado, la resolución de dominio funcionaba únicamente dentro de cada red, pero no entre ella. Es decir, en el caso de querer resolver el dominio **owncloud.um.es** desde la organización de la UM, no funcionaría (pero si podemos hacer ping a la IP de ese máquina).

Mi idea era solventar este inconveniente con un DNS en cada organización, con forwarding de peticiones.

2. TLS over SNMP

A pesar de haber conseguido la combinación de parámetros adecuada para compilar el software con soporte TLS, me daba algunos errores, por lo que al final desistí para no perder el ritmo de las demás prácticas.

3. PKI

Tenía previsto mejorar la infraestructura PKI. Actualmente se despliega mediante scripts y “apaños”. Mi idea era usar algo más serio, como *cfssl*

5. Manual de uso

Para desplegar la infraestructura en modo bridge, nos colocamos en la raíz del proyecto y ejecutamos:

```
$ ./up.sh -b
```

En caso de realizarlo en el escenario real (con el router), es imprescindible realizar previamente:

1. Cargar el archivo de configuración **lego_config_final.bin** en el firmware DD-WRT (Administration -> Backup)
2. Configurar nuestra máquina host con las VLAN. Es preferible marcar que no use las rutas obtenidas para salir a Internet. Una posible configuración resultante sería:
 - a. **vlan3** en la interfaz **enp2s0.3** (UM)
 - b. **vlan4** en la interfaz **enp2s0.4** (UPM)

Si podemos realizar ping al router a través de la VLAN, podemos lanzar el proyecto con:

```
$ ./up.sh -m enp2s0.3 -m enp2s0.4
```

NOTA: El orden de los parámetros es importante.

Tras esperar un tiempo, podemos comprobar que los servidores están desplegados con el siguiente comando:

```
$ docker-compose ps
      Name           Command       State    Ports
-----  
lego_um_pc1_1   /usr/bin/supervisord   Up
lego_um_snmp_1   /usr/bin/supervisord   Up
lego_um_voip_1   /usr/bin/supervisord   Up
```

...

Todos los servidores se despliegan con **supervisor**, para asegurar que sigan activos con sus múltiples servicios.

Listamos los servicios disponibles mediante:

```
$ docker-compose config --services
um_crl
um_ca
upm_ca
upm_crl
...
```

O usando el autocompletado de Bash (*Tabulador*) en órdenes específicas.

Podemos “entrar” dentro de cualquier servidor con el siguiente comando:

```
$ docker-compose exec um_snmp bash
root@um_snmp:/#
```

A partir de este punto, estamos en una subshell del servidor virtualizado.

También podemos realizar órdenes concretas mediante:

```
$ docker-compose exec um_voip asterisk -vr
Asterisk 14.6.2, Copyright (C) 1999 - 2016, Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty'
for details.
This is free software, with components licensed under the GNU General
Public
License version 2 and other licenses; you are welcome to redistribute it
under
certain conditions. Type 'core show license' for details.
=====
=
Connected to Asterisk 14.6.2 currently running on voip (pid = 10)
voip*CLI>
```

Es importante señalar que el despliegue del proyecto con el modo **macvlan** (escenario real), no permite que el host se comunique directamente con los servicios virtuales por su IP. Para ello, es necesario conectarse a la red con una interfaz que no sea la que se está usando con el router.

Destacar que en la ruta **/var/logs/supervisor** de cada contenedor se encuentra la salida de todos los servicios que ha levantado *supervisor*.

6. Bibliografía

- Transparencias TCI
- Transparencias STA
- https://es.wikipedia.org/wiki/Simple_Network_Management_Protocol
- http://www.personales.ulpgc.es/nramos.dit/?q=system/files/Tutorial_de_NET-SNMP.pdf
- <http://net-snmp.sourceforge.net/tutorial/tutorial-5/commands/snmptranslate.html>
- <http://net-snmp.sourceforge.net/wiki/index.php/TUT:snmptable>
- <http://net-snmp.sourceforge.net/wiki/index.php/TUT:snmptrap>
- <http://www.zeroshell.net/listing/Instalando-Asterisk-con-GUI-en-Zeroshell.pdf>
- <http://www.net-snmp.org/tutorial/tutorial-5/commands/snmptrap.html>
- <https://es.wikipedia.org/wiki/Asterisk>
- <https://www.gsp.com/cgi-bin/man.cgi?section=5&topic=snmpd.conf#5>
- http://tuan.nguoianphu.com/Net_SNMP_compile_for_Linux_Solaris_Windows
- http://net-snmp.sourceforge.net/wiki/index.php/Using_DTLS
- <https://blog.sinologic.net/2013-07/asterisk-101-como-solucionar-problema-audio-voip.html>
- <https://www.somosbinarios.es/primeros-pasos-con-docker-compose/>
- [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software))
- https://access.redhat.com/documentation/en-us/red_hat_update_infrastructure/2.1/html/administration_guide/chap-red_hat_update_infrastructure-administration_guide-certification_revocation_list_crl
- <https://blog.didierstevens.com/2013/05/08/howto-make-your-own-cert-and-revocation-list-with-openssl/>
- <http://thenubbyadmin.com/2015/02/19/fixing-unable-to-get-local-issuer-certificate-and-certificate-verify-failed-in-syslog-ng/>
- <https://www.tbs-certificates.co.uk/FAQ/en/200.html>
- https://es.wikipedia.org/wiki/Protocolo_de_iniciaci%C3%B3n_de_sesi%C3%B3n
- <https://www.voipmechanic.com/sip-call-example.htm>
- <https://en.wikipedia.org/wiki/FreeRADIUS>
- Tarea 5 TCI
- <https://es.wikipedia.org/wiki/OpenLDAP>
- <https://serverfault.com/questions/49146/ldap-structure-dc-example-dc-com-vs-o-example>
- <https://linux.die.net/man/1/ldapsearch>
- <https://superuser.com/questions/592650/what-does-binding-to-a-ldap-server-mean>
- <https://docs.oracle.com/javase/jndi/tutorial/ldap/misc/aliases.html>
- <https://wwwldap.com/the-ldap-search-operation>
- <https://linux.die.net/man/1/ldapadd>
- <https://linux.die.net/man/1/ldapdelete>
- <https://www.openssl.org/docs/manmaster/man1/ca.html>
- https://raymii.org/s/tutorials/OpenSSL_command_line_Root_and_Intermediate_CA_including_OCSP_CRL%20and_revocation.html
-