VNiVERSiDAD D SALAMANCA
CAMPUS DE EXCELENCIA INTERNACIONAL

DPTO. INFORMÁTICA Y AUTOMÁTICA
Plaza de los Caídos, s/n. 37008 Salamanca
Tel.: +34 923 29 46 53 Fax:+34 923 29 45 14
diaweb.usal.es sec_dia@usal.es

MEMORIA FINAL DE EJECUCIÓN

Código del proyecto: ID2019/032

Diseño de materiales prácticos para la administración de redes de computadores

Responsable: Ángeles Mª Moreno Montero

Miembros del equipo de trabajo:

Francisco Javier Blanco Rodríguez

Belén Curto Diego

Vidal Moreno Rodilla

María José Polo Martín

Salamanca, 30 de junio de 2021

Contenido

1. Introducción

Este documento constituye el informe final de ejecución del proyecto de Innovación Docente realizado durante los cursos académicos 2019-2020 y 2020-2021 en el Departamento de Informática de la Facultad de Ciencias. Ha sido financiado mediante la convocatoria de 2018-2019 de Ayudas de la Universidad de Salamanca para los Proyecto de Innovación y Mejora Docente. El proyecto se ha retrasado 1 año debido a la Covid-19.

En la actualidad es difícil encontrar una empresa o una entidad que no disponga de una infraestructura de red compuesta por innumerables equipos. Administrar grandes redes es una tarea compleja puesto que los recursos de red son muchos y hay probabilidad de que fallen más elementos. Un administrador necesita monitorizar entre otros: la utilización del ancho de banda, el estado de funcionamiento de los enlaces, mantener un registro estadístico del tráfico que circula por los enlaces a redes de área extensa (WAN) o con la Internet, la detección de cuellos de botella, detectar y solventar problemas con el cableado, administrar la información de encaminamiento entre equipos, etc.

La disponibilidad de herramientas software que ayuden en las tareas enumeradas se hace vital para el administrador de redes. Existen numerosas técnicas y herramientas para la gestión de redes, pero en este trabajo nos hemos centrado en el protocolo estándar en Internet de la capa de aplicación SNMP (*Simple Network Management Protocol*) en el que de alguna forma u otra se basan otras herramientas.

En el ámbito del Grado en Ingeniería Informática, la materia Redes está formada por las asignaturas: Señales y Sistemas, Redes de Computadores I, Redes de Computadores II y Seguridad en Sistemas Informáticos. Estas asignaturas constan de partes teóricas y de partes prácticas, donde se pretende que el alumno aplique lo aprendido en la teoría a escenarios cercanos a la realidad.

En el caso de la asignatura Redes de Computadores II, la gestión en redes es solo un apartado de esta. Debido al hecho de que se aborda la gestión desde un punto de vista general no es posible profundizar en este aspecto ni practicar con distintas configuraciones de red. Es importante destacar que la parte práctica de la asignatura se realiza en el Laboratorio de Informática de la Facultad de Ciencias, cuya configuración no puede ser alterada por el alumno.

Por ello, las aplicaciones de simulación son una buena opción, ya que ofrecen la posibilidad de crear topologías de red completamente simuladas sin limitaciones físicas ni económicas. Existen varias herramientas de simulación que permiten una experiencia real y cercana a entornos que pueden darse en cualquier organización. Estas herramientas simulan los elementos de red (switches, routers, etc.) como elementos de Cisco Systems, uno de los fabricantes más importantes en esta área. Este aspecto constituye un beneficio para el alumno, que podrá experimentar cómo se configuran equipos Cisco ampliamente utilizados en el ámbito empresarial.

Actualmente se dispone de un prototipo de laboratorio de redes y de materiales educativos para el estudio de las redes fruto de los siguientes proyectos de innovación docente:

- Proyecto: PL11/023. Elementos de Interconexión de Redes: Integrando Voz y Datos. Innovación Docente USAL 2011
- Proyecto ID2013/311. Diseño de materiales para el laboratorio de interconexión de redes: Integrando voz y datos. Innovación docente USAL 2013
- ID2017/029. Diseño de materiales prácticos para la asignatura Seguridad en Sistemas Informáticos del grado en Ingeniera Informática. Innovación docente USAL 2017

Los equipos adquiridos con la financiación conseguida en estos proyectos son de gama media-alta, siendo similares a los utilizados en medianas y grandes empresas. La disponibilidad de estos equipos va a dotarnos de un entorno real sobre el que implementar protocolos de gestión de redes, permitiendo realizar una demostración de un laboratorio centrado en la gestión sobre una red avanzada.

Así, este proyecto de innovación se basa en la idea de elaborar material didáctico para que el estudiante pueda profundizar en el ámbito de la gestión de redes mediante guiones teóricos y prácticos, utilizando software de simulación, además de aplicarse todos los conocimientos adquiridos durante la realización de los materiales didácticos sobre un laboratorio real.

El resto del documento se organiza como sigue. En primer lugar, se enumeran los objetivos del proyecto. A continuación, se examina el proceso de diseño del material docente que se desea generar. En el apartado cuarto se describen brevemente todos los materiales elaborados, tanto los materiales para entornos simulados, como para nuestro laboratorio de redes con equipos reales. En el apartado cinco se muestran los equipos adquiridos y reacondicionados, sus principales características y se explica su rol en este proyecto. Para finalizar con las principales conclusiones y ampliaciones futuras.

2. Objetivos

El objetivo principal de este proyecto es la elaboración de materiales didácticos que permitan a los estudiantes de Ingeniería Informática adquirir competencias en el ámbito de la administración de redes de computadores. La administración o gestión de redes abarca tareas de integración y coordinación del hardware y software para supervisar el correcto funcionamiento de una red garantizando así niveles de servicio adecuados a los objetivos de una instalación y de una organización. Esto enlaza directamente con el perfil profesional de administrador de redes.

También, como objetivo paralelo, se propondrá un marco de gestión aplicado a un prototipo de laboratorio de redes que integra voz y datos, aplicando técnicas más avanzadas que solo son posibles con la utilización de equipos reales de interconexión de redes. El prototipo ha sido ampliado con equipos reacondicionados para este proyecto que han sido cedidos por el CPD de la Universidad de Salamanca. También se han recuperado equipos antiguos perteneciente al Departamento de Informática y Automática pero que son perfectamente válidos para nuestras necesidades a la hora de disponer de equipos finales.

Estos objetivos se conseguirán mediante una adecuada selección de contenidos y la elaboración de documentación que permita al alumno realizar las prácticas tanto en

entornos simulados como reales. Esta documentación estará compuesta por lecciones teóricas y sus guiones prácticos (junto a sus soluciones óptimas) asociados.

Como ya se ha comentado en la introducción en la actualidad se realizan prácticas de gestión de redes en las aulas de informática limitadas por la infraestructura operativa de la red de la Universidad de Salamanca. En este sentido, las aplicaciones de simulación son una buena opción, ya que ofrecen la posibilidad de crear topologías de red completamente simuladas sin limitaciones físicas ni económicas. En nuestra propuesta los supuestos se realizarán en entornos de simulación o en nuestro laboratorio de redes.

3. Diseño del material docente

Para conseguir los objetivos planteados ha sido necesario realizar una adecuada selección de contenidos y la elaboración de los materiales que permitan al alumno realizar las prácticas tanto en entornos simulados como reales.

A continuación, se describe el proceso de diseño del material elaborado. El primer paso es definir qué contenidos se van a abordar en las prácticas y cómo clasificarlos de forma lógica. Una vez realizado este paso se han elaborado los guiones, los cuales cuentan con dos versiones, una con las soluciones del escenario y otra para el alumno, donde se marcarán los objetivos, los pasos, las pruebas y evaluaciones necesarias para comprobar que la solución al escenario propuesta por el alumno es correcta y que se han comprendido los conceptos teóricos. Para cada uno de los contenidos se escogerá el mejor entorno para su implementación práctica bien sea en simulaciones, emulaciones o en equipos reales.

Los materiales han sido realizados siguiendo los mismos pasos y estructura que se puede observar en la Tabla 1. Se realiza un primer apartado teórico que introduzca los conceptos de la gestión de redes. El siguiente apartado describe los objetivos pretendidos. En el tercero y cuarto se explica lo más relevante del concepto a tratar apoyándose en las imágenes más esclarecedoras. El quinto es la elección del entorno real o simulado en el que realizar la práctica. Y los últimos apartados serán meramente prácticos, dividido en dos guiones; el guion dirigido al alumno y el guion con la solución al escenario presentado. Se finaliza con bibliografía y enlaces de interés para ampliar el tema.

1. Selección y clasificación de contenidos
2. Definición de objetivos del apartado
3. Elaboración de los apartados teóricos
4. Elaboración de figuras que ayuden a comprender los conceptos teóricos
5. Elección de la herramienta real o en simulación (Cisco Packet Tracer o GNS3)
6. Guion práctico para el alumno
   - 6.1. Elaboración de escenarios preconfigurados y prediseñados.
   - 6.2. Conjunto de pasos a realizar.
   - 6.3. Elaboración de preguntas de evaluación.
7. Guion con la solución óptima
   - 7.1. Elaboración de los escenarios y configuraciones que den solución a la práctica en cuestión.
   - 7.2. Sugerencias para la evaluación de la práctica.
8. Enlaces para la ampliación de los temas tratados.

Tabla 1: Pasos seguidos para la elaboración de los materiales didácticos.

4. Resultados: Material elaborado

Es este apartado se va a presentar el material elaborado durante la realización de este proyecto de innovación comenzando con los conceptos teóricos mínimos para poder trabajar con SNMP. En el manual elaborado sobre los conceptos teóricos de SNMP se incluyen numerosas capturas del tráfico que este protocolo genera con la idea de que el estudio de sus mensajes sea visto por el estudiante desde el punto de vista práctico y de esta forma comprendan mejor el funcionamiento del protocolo. También se han elaborado manuales de cómo activar SNMP en las distintas plataformas: equipos Linux, Windows y CISCO IOS.

Y finalmente se ha elaborado un guion de prácticas en su versión para el alumno y su solución para el profesor que abarca el despliegue de la gestión de una red tanto en entorno simulado con la herramienta GNS3 como con los equipos CISCO que disponemos de otros proyectos en nuestro laboratorio de redes junto con los reacondicionados para este que han sido cedidos por el CPD de la Universidad de Salamanca.

En la Figura 1 se muestra el laboratorio de redes con los equipos reales y la ejecución del escenario simulado en GNS3 tanto en un equipo Windows como en un Linux.

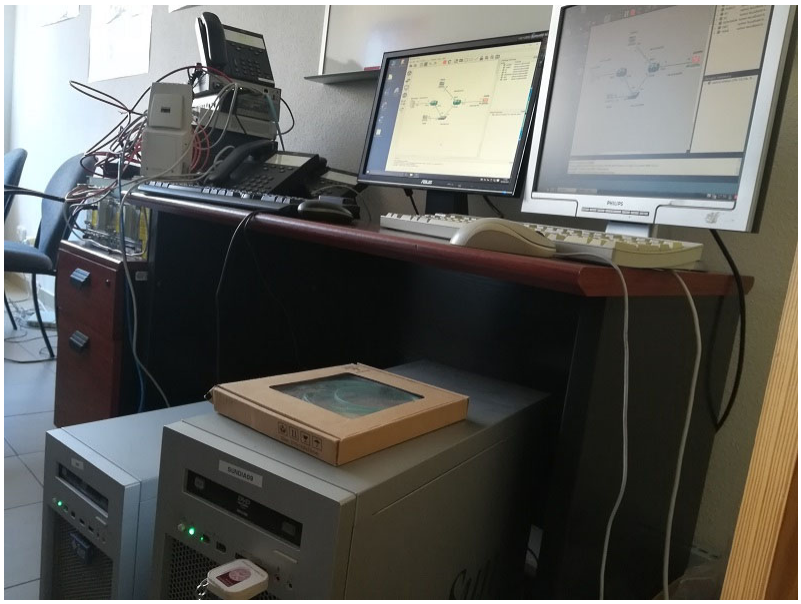


Figura 1: Vista general del prototipo de laboratorio de redes.

Este material se encuentra publicado en la plataforma de enseñanza virtual Diaweb [1] tal como se muestra en la Figura 2.

Figura 2: Práctica sobre gestión de redes publicada en Diaweb.

A continuación, se explica cómo se han elaborado estos materiales.

4.1. Conceptos teóricos del Protocolo Simple de Gestión de Red

Un sistema de gestión de red está formado por un conjunto de herramientas que permiten la supervisión y el control de la red, de forma que los diferentes dispositivos informan de su estado a un equipo central. Dicho de otra forma, son herramientas que observan el funcionamiento de los equipos y de los servicios que tienen instalados, y cuando algo empieza a ir mal informan al administrador.

El protocolo de gestión de red estándar en Internet se denomina SNMP (*Simple Network Management Protocol*). Su modelo de gestión de red, definido en el RFC 1157 [2], especifica los siguientes elementos (Ver Figura 3):

- **Equipo de gestión**: host donde el administrador de la red realiza la gestión.
- **Agente de gestión**: los diferentes dispositivos como switches, routers, host que tienen implementado el protocolo SNMP podrán ser gestionados desde el equipo de gestión.
- **Base de información de gestión (MIB)**: el conjunto de recursos administrados representados mediante un nombre y un valor.
- **Protocolo de gestión de red**: el protocolo que utilizan los diferentes elementos para comunicarse.

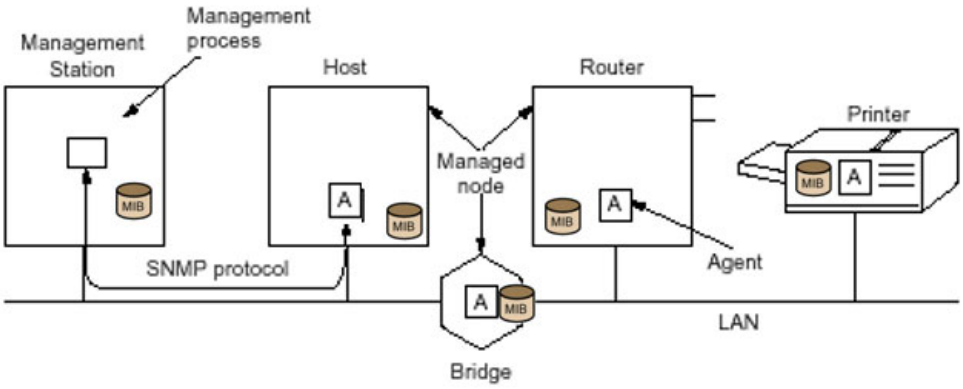


Figura 3: Modelo de gestión en SNMP.

En 1998 se publicó la especificación para SNMP y rápidamente se convirtió en el estándar predominante de gestión de red. Una serie de distribuidores ofrecen estaciones de gestión de red autónomas basadas en SNMP, y la mayoría de los vendedores de switches, routers, estaciones de trabajo y computadores personales ofrecen paquetes del agente SNMP que permiten que sus propios productos sean gestionados por una estación de gestión. A estos equipos se les conoce como gestionables y hemos de estar atentos a esta característica cuando se adquiera un equipo de red.

SNMP se diseñó como protocolo de nivel de aplicación para formar parte de los protocolos TCP/IP. Fue pensado para operar sobre UDP (*User Datagram Protocol*) como capa de transporte. En el equipo de gestión, un proceso administrador controla el acceso a la MIB central y proporciona una interfaz al administrador de red. Cada agente debe tener un proceso capaz de interpretar los mensajes SNMP y de controlar la MIB. Los puertos que se utilizan son el 161 y el 162, este último utilizado para la recepción de avisos o *traps*.

Como se puede observar en la Figura 4, desde un equipo gestor se pueden generar tres tipos de mensajes:

- *GetRequest*
- *GetNextRequest*
- *SetRequest*

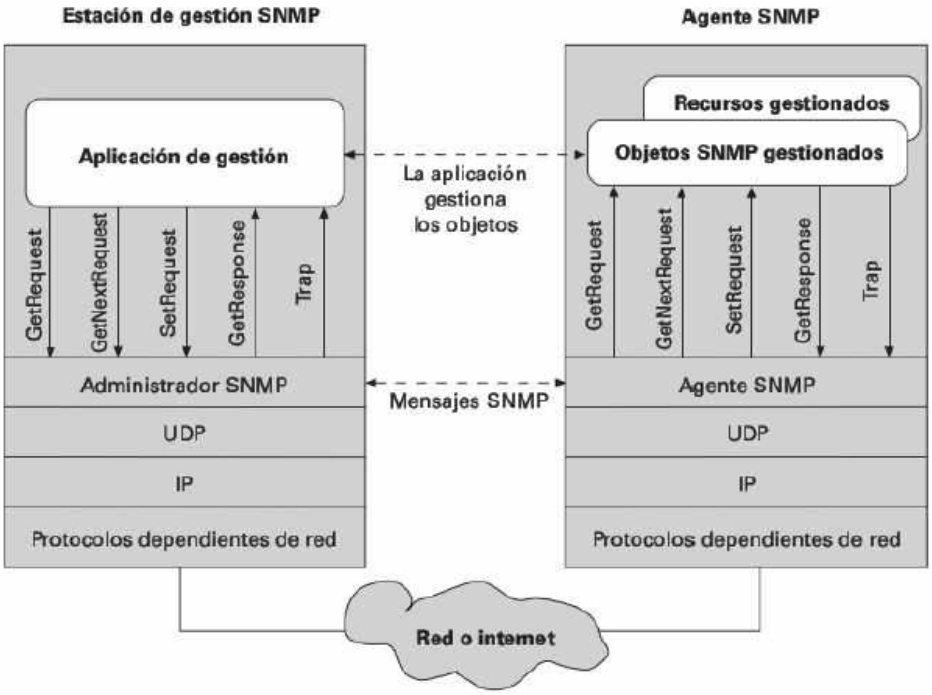


Figura 4: Contexto de una aplicación SNMP

Los dos primeros son variaciones de la función *Get*. Los tres mensajes son reconocidos por el agente mediante un mensaje de tipo *GetResponse*, que se entrega a la aplicación de gestión. Además, un agente puede emitir un mensaje *Trap* en respuesta a un hecho que afecte a los recursos gestionados de la MIB.

En el estudio de cualquier protocolo es importante entender los diferentes mensajes que se intercambian los elementos presentes, en este caso en una estructura SNMP. Por ello, en el manual elaborado se mostrarán los mensajes de SNMP en sus tres versiones (SNMPv1, SNMPv2c y SNMPv3) desde un punto de vista práctico apoyándose en capturas de tráfico realizadas en este proyecto de innovación. No se van a incluir en esta memoria todos los mensajes del protocolo, pero a modo de ejemplo se muestra a continuación el mensaje *GetRequest*.

A través de este mensaje el gestor solicita al agente un valor de un objeto de interés. En la Figura 5 se muestra un mensaje *GetRequest* enviado en la versión SNMPv1.

```
Simple Network Management Protocol
    version: version-1 (0)
    community: public
  data: get-request (0)
    get-request
        request-id: 826437480
        error-status: noError (0)
        error-index: 0
      variable-bindings: 1 item
        1.3.6.1.2.1.1.3.0: Value (Null)
            Object Name: 1.3.6.1.2.1.1.3.0 (iso.3.6.1.2.1.1.3.0)
            Value (Null)
```

Figura 5: Mensaje GetRequest SNMPv1.

La versión SNMPv2c lanza el mismo mensaje, sin embargo, la versión SNMPv3 tiene más campos (Figura 6).

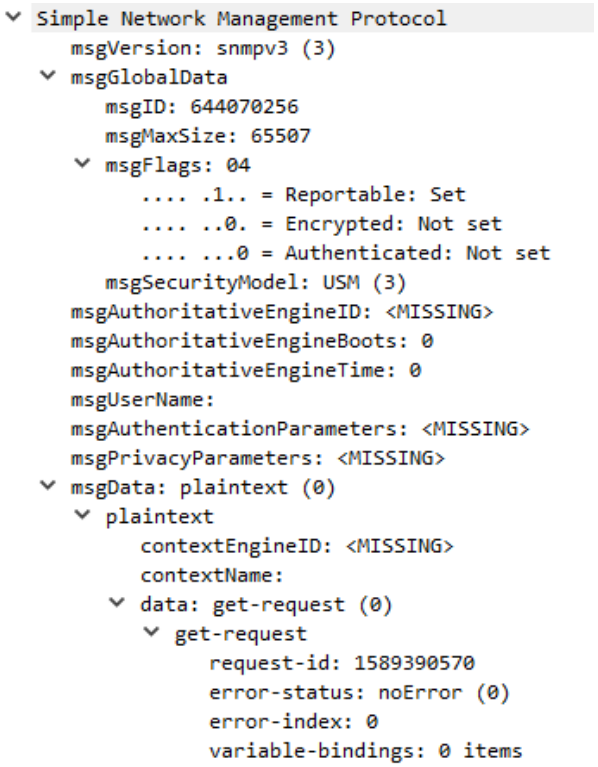


Figura 6: Mensaje *GetRequest* SNMPv3.

4.2. Guion para la Gestión de una red

En el guion elaborado aparecen los siguientes objetivos:

1. Conocer los aspectos básicos de la administración y gestión de una red.
2. Adquirir la capacidad de analizar la información más relevante del protocolo SNMP.
3. Familiarizarse con la configuración y funcionamiento de todas las versiones del protocolo SNMP.
4. Conocer el manejo de la herramienta NET-SNMP y los fundamentos en los que se basa.
5. Explorar e identificar partes estándares o empresariales del árbol MIB
6. Instalar MIB específicas de fabricantes y utilizarlas con comandos SNMP.

La realización de los escenarios propuestos ocupará de dos a cuatro sesiones de prácticas de 2 horas de duración.

No vamos a incluir en esta memoria el guion completo por su elevada extensión, pero mostraremos integra la primera parte de este dónde se trabaja con un escenario en el entorno de simulación *GNS3* [3] en la configuración de SNMPv2c. El guion continua en el escenario de simulación con la versión SNMPv3 y finaliza trabajando con los equipos reales donde el alumno ha de aplicar las técnicas aprendidas en las prácticas realizadas previamente en los entornos de simulación. Las respuestas que debe proporcionar el alumno se muestran en color verde y constituyen el guion para el profesor.

4.2.1. Materiales para la versión SNMPv2c

Configuración

El escenario sobre el que vamos a trabajar es el que se muestra en la Figura 7. El equipo con nombre **SERVIDOR** contiene la información sensible de la empresa a la que pertenece esta red. A él pueden acceder desde diferentes equipos. Uno de ellos es **ADMIN** que es el correspondiente al gestor de la red. Este equipo debe tener completos derechos de acceso al servidor e incluso poder modificar elementos de su configuración desde una subred distinta a la que pertenece el equipo **SERVIDOR**. Todos los equipos finales son equipos Linux Ubuntu y los routers son CISCO 7200.

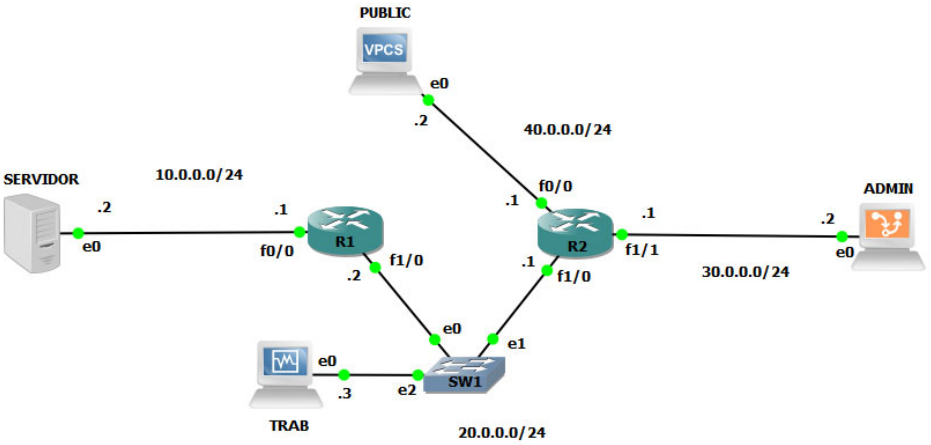


Figura 7: Escenario de red para su gestión con SNMP en GNS3.

Utilizando la versión SNMPv2c configura el equipo **SERVIDOR** de una forma segura para que solo pueda acceder a la totalidad de la MIB el equipo **ADMIN** y el propio equipo en *localhost*.

Para que tan solo el equipo ADMIN tenga total acceso al equipo SERVIDOR es necesario configurar adecuadamente el archivo /etc/snmp/snmpd.conf del lado del SERVIDOR. Esta configuración debe basarse en el modelo VACM (*View Access Control Model*). Para ello vamos a crear una ACL (*Access Control List*) solo para la dirección IP de ADMIN. Una vez creada la asociamos a un grupo, que según el RFC 3415 [4] es un conjunto de cero o más tuplas <*securityModel, securityName*> en cuyo nombre se pueden acceder a los objetos de gestión SNMP. Este grupo define los derechos de acceso otorgados a todos los *securityName* que pertenecen a ese grupo.

Después es necesario crear las vistas del modelo. En este caso, el equipo ADMIN debe tener total acceso a la MIB del SERVIDOR para poder leer o modificar cualquiera de sus objetos. Asociamos al grupo los derechos que posee, tanto de lectura como de escritura y sobre qué vista del modelo. El formato que utiliza es:

access – grupo – modelo de seguridad – nivel de seguridad – prefijo – lectura – escritura – notif

El fichero snmpd.conf del SERVIDOR quedaría de la siguiente manera:

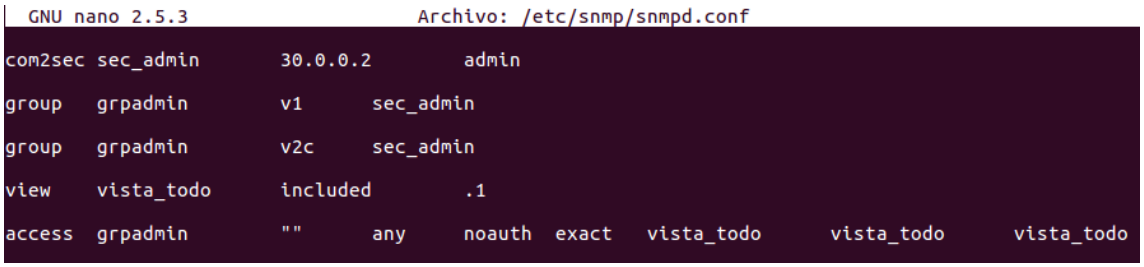


Figura 8: Configuración del agente en SERVIDOR con los accesos de ADMIN.

El grupo *grpadmin* tendría permisos de lectura, escritura y recibirá notificaciones de todo el árbol MIB.

Esta empresa necesita dar acceso a una serie de técnicos encargados del mantenimiento del **SERVIDOR** y otros elementos críticos de la red de la empresa. Para ello configura el equipo **TRAB** de tal forma que tenga acceso a la información de la MIB del equipo **SERVIDOR**. ¿Sería recomendable que estos técnicos tuviesen acceso a toda la MIB? ¿Qué ramas serian interesantes? ¿Deberían tener permisos de escritura?

No es recomendable que alguien que no sea el administrador de la red tenga acceso a poder modificar todos los elementos de la MIB. Los técnicos solo deben conocer si el equipo está funcionando y datos del servidor como saber dónde está situado o quien es la persona de contacto. Por ello, es interesante crear una vista nueva del árbol MIB del equipo SERVIDOR en la que solo puedan tener acceso a ello. Como pueden estar encargados de moverlos o actualizarlos, deben tener permiso de lectura y escritura sobre ellos. No es necesario que reciban notificaciones.

Para ello seguimos el proceso descrito anteriormente, pero en vez de crear una ACL solo de una IP, ahora la crearemos para el rango de la subred 20.0.0.0/24 en la que están alojados los ordenadores de los técnicos de la empresa.

También es necesario crear el grupo asociado a ese rango y la vista que le vamos a permitir acceder, en este caso la correspondiente a .1.3.6.1.2.1 que son todos los detalles del sistema como el tipo, contacto, localización, etc. Por último, asociamos a ese grupo los permisos de lectura y escritura.

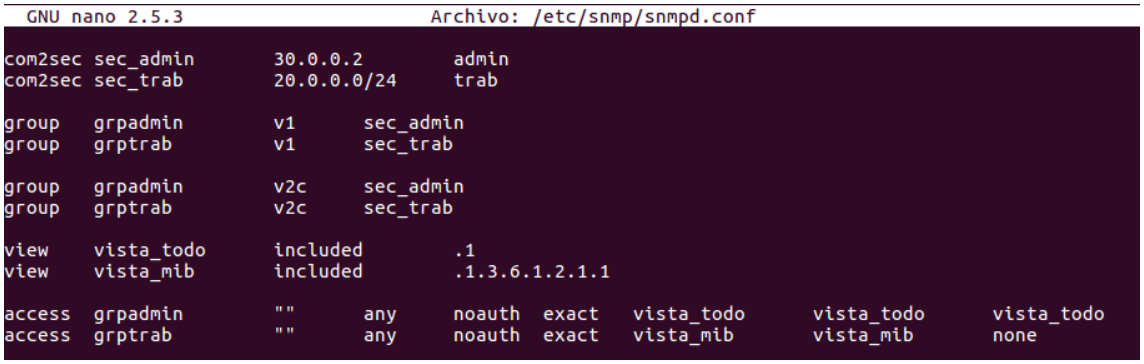


Figura 9: Configuración del agente en SERVIDOR con los accesos de TRAB.

Un elemento importante de la red son los routers y switches. Es necesario configurar también SNMP en ellos. ¿Cómo debe ser esta configuración? ¿Debe ser la misma para todos?

Sí, es necesario realizar una configuración de SNMP sobre ellos, aparte de para poder comprobar su tráfico u otras variables de memoria o procesamiento, es necesario para

que tenga una buena configuración y no sea susceptible de ser atacado y comprometer la seguridad de la red.

La configuración debe basarse en el mismo modelo que el utilizado para el equipo SERVIDOR, el modelo VACM, es necesario crear una ACL, asociarla a un grupo y una vista para que pueda trabajar el módulo de control de acceso.

Para configurar SNMP en ambos routers, abrimos la terminal de uno de ellos y seguimos el mismo esquema que para los equipos Ubuntu. Como nos estamos basando en el modelo VACM lo primero es crear una ACL, en este caso dos, una para los equipos como TRAB y otra para ADMIN, para ello escribimos los siguientes comandos:

```
#conf t
#access-list 11 permit host 30.0.0.2
#access-list 11 deny any log
#access-list 22 permit host 20.0.0.2
#access-list 22 deny any log
#snmp-server view vista_todo 1 included
#snmp-server view vista_mib system included
#snmp-server community admin view vista_todo rw 11
#snmp-server community trab view vista_mib rw 22
```

Si comprobamos con un *snmpget* desde ADMIN, podremos acceder a toda la MIB del router configurado, sin embargo, desde TRAB solo podremos entrar al subárbol *system*.

Por último, esta empresa tiene miedo de que alguien ajeno a la jerarquía de trabajadores pueda obtener información sensible de los equipos de su red. ¿Cómo debería ser la configuración de los equipos para que no pueda acceder a ella nadie ajeno a la empresa?

Cuando un equipo que no pertenezca a ninguna de las ACL anteriores lo consideraremos externo de la empresa y será del grupo *grppublic*. A este grupo no le vamos a permitir acceder a ningún elemento del árbol MIB del equipo SERVIDOR y por extensión, de ningún equipo de la red para no comprometer su seguridad. Para ello en el módulo de control de acceso no le daremos ni permisos de lectura, ni de escritura ni para poder recibir notificaciones.

```
  GNU nano 2.5.3                    Archivo: /etc/snmp/snmpd.conf

com2sec sec_admin       30.0.0.2        admin
com2sec sec_trab        20.0.0.0/24     trab
com2sec sec_public      default         public


group   grpadmin        v1      sec_admin
group   grptrab         v1      sec_trab
group   grppublic       v1      sec_public

group   grpadmin        v2c     sec_admin
group   grptrab         v2c     sec_trab
group   grppublic       v2c     sec_public


view    vista_todo      included        .1
view    vista_mib       included        .1.3.6.1.2.1.1


access  grpadmin        ""      any     noauth  exact   vista_todo      vista_todo      vista_todo
access  grptrab         ""      any     noauth  exact   vista_mib       vista_mib       none
access  grppublic       ""      any     noauth  exact   none            none            none
```

Figura 10: Configuración del agente en SERVIDOR con los accesos públicos.

La máquina **SERVIDOR** por sí misma ¿Qué acceso tendía al árbol MIB? ¿Qué derechos debería tener sobre ella?

Con la configuración actual no podría ver nada puesto que su IP, a efectos prácticos la 127.0.0.1 (*localhost*), no pertenece a ningún elemento de la ACL así que cualquier petición sería rehusada por el demonio *snmpd*.

Para evitar esto es necesario añadir un elemento más a la ACL para dar acceso al propio equipo, crear un grupo y darle acceso a una vista de la MIB, de esta forma el módulo de control de acceso no rechazará las peticiones con origen en *localhost*.

Puede ser necesario que, por motivos de seguridad o de algún tipo de impedimento como la caída de la red, se pueda trabajar como administrador en la propia máquina SERVIDOR, por ello a su grupo, el *grplocal*, se le conceden permisos de lectura y escritura en toda la MIB.

```
  GNU nano 2.5.3                    Archivo: /etc/snmp/snmpd.conf

com2sec sec_local       localhost       private
com2sec sec_admin       30.0.0.2        admin
com2sec sec_trab        20.0.0.0/24     trab
com2sec sec_public      default         public


group   grplocal        v1      sec_local
group   grpadmin        v1      sec_admin
group   grptrab         v1      sec_trab
group   grppublic       v1      sec_public

group   grplocal        v2c     sec_local
group   grpadmin        v2c     sec_admin
group   grptrab         v2c     sec_trab
group   grppublic       v2c     sec_public


view    vista_todo      included        .1
view    vista_mib       included        .1.3.6.1.2.1.1


access  grplocal        ""      any     noauth  exact   vista_todo      vista_todo      none
access  grpadmin        ""      any     noauth  exact   vista_todo      vista_todo      vista_todo
access  grptrab         ""      any     noauth  exact   vista_mib       vista_mib       none
access  grppublic       ""      any     noauth  exact   none            none            none
```

Figura 11: Configuración del agente en SERVIDOR con los accesos locales.

Enviando y analizando los mensajes del protocolo

Ahora el **SERVIDOR** está configurado para dar soporte en SNMPv2c, vamos a analizar el tráfico SNMP generando distintos mensajes sobre este equipo:

1. Envía un ***snmpget*** desde ADMIN hasta SERVIDOR de cualquier elemento de la MIB. ¿Cuántos mensajes se intercambian? ¿En qué orden? Y ¿de qué tipo? Comenta los aspectos destacados de la captura.

   Para este ejemplo vamos a preguntar al equipo SERVIDOR por su localización, para ello utilizaremos el siguiente comando:

   ```
   snmpget -v2c -c admin 10.0.0.2 sysLocation.0
   ```

   El comando hará una petición GET en la versión 2c del protocolo SNMP identificado como miembro de la comunidad “admin” a la dirección IP 10.0.0.2 por el elemento sysLocation.0 de su árbol MIB. La captura de tráfico de este mensaje es la que se muestra en la Figura 12.

No.	Time	Source	Destination	Protocol	Length	Info
11	34.425853	30.0.0.2	10.0.0.2	SNMP	84	get-request 1.3.6.1.2.1.1.6.0
12	34.461157	10.0.0.2	30.0.0.2	SNMP	93	get-response 1.3.6.1.2.1.1.6.0

Figura 12: Mensajes *GetRequest* y *GetResponse* desde ADMIN a SERVIDOR.

Podemos comprobar que para una petición GET con un resultado exitoso se intercambian dos mensajes. Uno desde ADMIN (30.0.0.2) a SERVIDOR (10.0.0.2) de tipo *get-request*, es decir, la petición y otro desde SERVIDOR para ADMIN que contiene la respuesta de tipo *get-request*. También vemos que la variable por la que preguntamos en el árbol MIB es la 1.3.6.1.2.1.1.6.0 que se corresponde con *sysLocation*.

Analizando más a fondo el mensaje *get-request* cuyo detalle se muestra en la Figura 13.

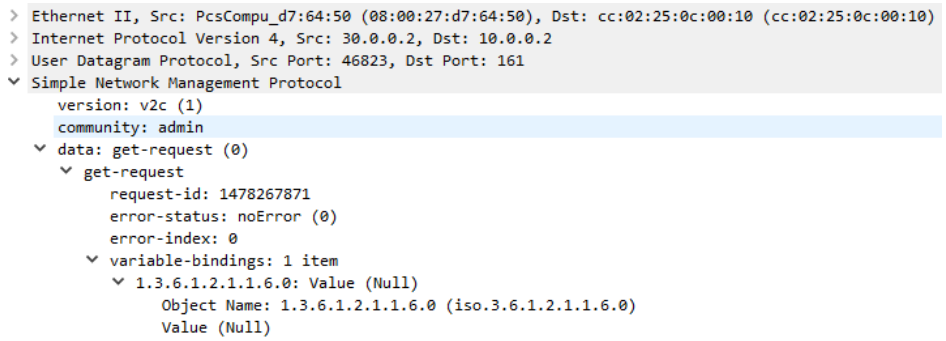


Figura 13: Detalle del mensaje *GetRequest* desde ADMIN a SERVIDOR.

Lo primero es comprobar las distintas cabeceras de las capas OSI. En este caso hemos utilizado Ethernet a nivel de enlace. También nos apoyamos en IP en capa de red y sobre UDP en la capa de transporte. SNMP pertenece a la capa de aplicación. Este protocolo envía estos mensajes al puerto 161.

Entrando más a fondo en la información de SNMP, comprobamos que la versión utilizada para la petición es la v2c y va dirigida a la comunidad (*Community*) *admin*. Esto se corresponde con el comando que hemos escrito para hacer este *get-request*.

Dentro del campo *data* de este paquete va la información importante de esta petición. Comprobamos que se añade un *request-id* para identificar el mensaje. *Error-status* y *error-index* nos proporcionan la información necesaria para conocer el porqué de un fallo en una comunicación de este tipo. *Variable-bindings* contiene los datos de la variable del árbol MIB que solicitamos. En este caso solo hemos pedido la información de *sysLocation*.0 (1.3.6.1.2.1.1.6.0) por eso la captura nos muestra que solo contiene 1 ítem. Si desplegamos este OID vemos que no tiene valor, aún está a *null* puesto que es una petición.

Ahora estudiaremos la captura del *get-response* (Figura 14)

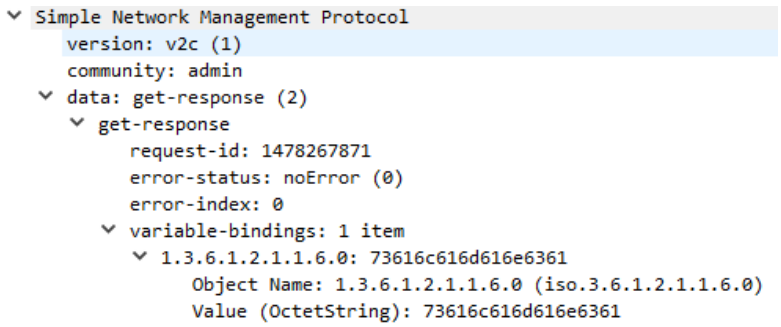


Figura 14: Detalle del mensaje *GetResponse* desde SERVIDOR a ADMIN.

Los primeros campos se corresponden con los de la petición *get-request*, la versión es la v2c, la comunidad a la que va dirigida es *admin*. Ahora el campo *data* contiene la información de *get-response*. El *request-id* es el mismo que en el primer mensaje intercambiado y como en esta comunicación no ha habido ningún error, los campos *error-status* y *error-index* van con *no-error* y 0 respectivamente. En *variable-bindings* está la información que solicitamos. Si desplegamos este campo volvemos a ver que solo contiene 1 ítem ya que preguntamos únicamente por una variable. Ahora vemos que el campo *Value* no va a *null* si no que contiene información de tipo *OctetString*. Este tipo de dato no nos permite ver en un lenguaje entendible para nosotros cuál es el contenido de esa variable, pero en la terminal sale formateado a un String normal, por tanto, la salida del comando *snmpget* descrito en este apartado es la que se muestra en la Figura 15.

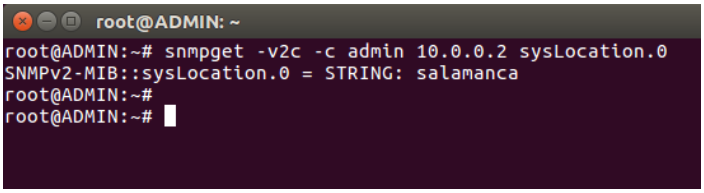


Figura 15: Respuesta a una petición SNMP.

2. Envía dos ***snmpget*** desde TRAB hasta SERVIDOR. Uno sobre un elemento de la MIB del que tenga permiso y otro sobre el que no. ¿Qué es lo que sucede? Comenta la captura de los mensajes de este intercambio.

El primer comando es el que no tiene que recibir la información, es decir, aquel que no tiene acceso en el árbol MIB, para ello vamos a acceder a la variable que se aloja en 1.3.6.1.2.1.2.1.0 que se corresponde con *ifNumber.0* dentro de la rama *Interfaces*.

```
snmpget -v2c -c trab 10.0.0.2 1.3.6.1.2.1.2.1.0
```

La versión que utilizamos vuelve a ser las v2c, ahora sobre la comunidad "trab" a la que pertenece este equipo. Vamos a preguntar al SERVIDOR que tiene la IP 10.0.0.2 por la variable de la MIB 1.3.6.1.2.1.2.1.0.

El otro comando solicitando la información de una variable del árbol MIB al que sí tiene acceso es el siguiente:

```
snmpget -v2c -c trab 10.0.0.2 1.3.6.1.2.1.1.3.0
```

En este caso preguntamos por la variable *sysUpTime.0*.

La captura de esta comunicación es la siguiente:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	20.0.0.3	10.0.0.2	SNMP	83	get-request 1.3.6.1.2.1.2.1.0
2	0.020616	10.0.0.2	20.0.0.3	SNMP	83	get-response 1.3.6.1.2.1.2.1.0
3	5.799959	20.0.0.3	10.0.0.2	SNMP	83	get-request 1.3.6.1.2.1.1.3.0
4	5.814970	10.0.0.2	20.0.0.3	SNMP	85	get-response 1.3.6.1.2.1.1.3.0

Figura 16: Solicitud de información accesible y no accesible.

Se intercambian dos mensajes igual para cada comando. La diferencia está en el *get-response*, en uno tiene una longitud (campo *Length* de 83 y otro de 85). Eso es porque en el campo *value* del segundo comando sí va con información, es decir, con

algún tipo de contenido. El detalle del primer *get-response* se muestra en la Figura 17.

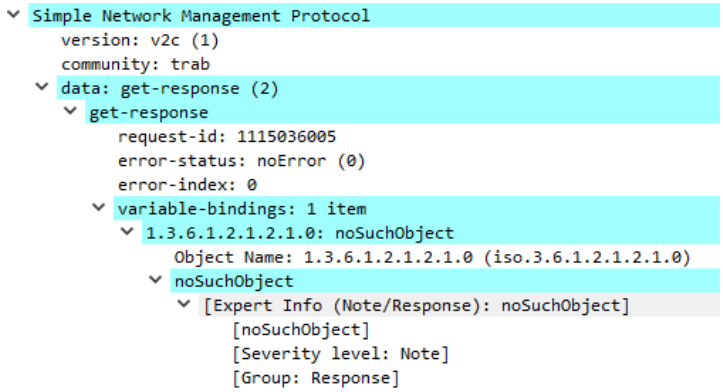


Figura 17: Respuesta a una solicitud de información sobre la que no se tiene acceso.

Podemos comprobar que no nos devuelve información de ningún tipo, tan solo nos dice que no hay ningún objeto como ese, *noSuchObject*, que es lo que devuelve cuando el equipo que realiza la petición no tiene accesible el OID que ha solicitado.

Por otro lado, en el segundo *get-response* (Figura 18) vemos que el campo *variable-bindings* contiene la información sobre el OID que preguntamos. En este caso vemos el tiempo que lleva el sistema arrancado en *Timeticks* que se corresponde con segundos.

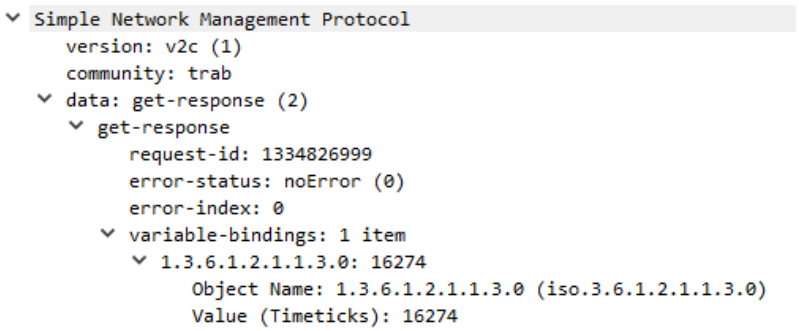


Figura 18: Respuesta a una solicitud de información sobre la que se tiene acceso

Esto se produce por la configuración que hicimos en el equipo SERVIDOR, la comunidad trab solo es accedida por los equipos de la subred 20.0.0.0/24 y tiene acceso a la vista "vista_mib" que solo tiene acceso a las ramas que van después de 1.3.6.1.2.1.1.

3. Envía un ***snmpget*** desde PUBLIC a SERVIDOR. ¿Qué sucede? ¿Qué mensajes intercambian? Comenta la captura de los mensajes de este intercambio.

El comando que vamos a utilizar es el siguiente:

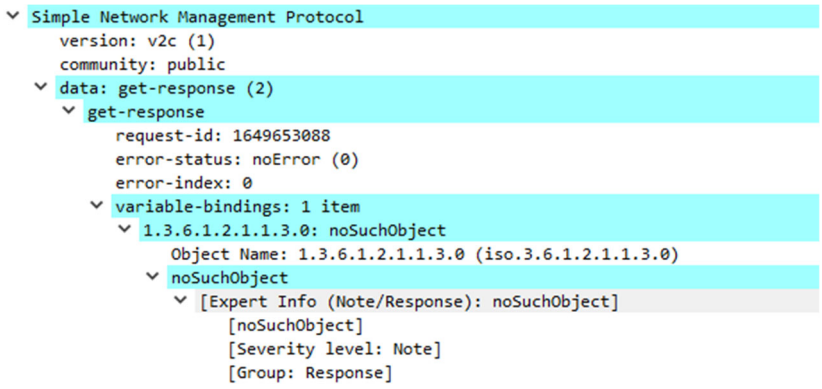
```
snmpget -v2c -c public 10.0.0.2 sysUpTime.0
```

La captura de este intercambio es la siguiente:

No.	Time	Source	Destination	Protocol	Length	Info
2	3.917624	40.0.0.2	10.0.0.2	SNMP	85	get-request 1.3.6.1.2.1.1.3.0
3	3.950984	10.0.0.2	40.0.0.2	SNMP	85	get-response 1.3.6.1.2.1.1.3.0

No cambia el esquema seguido por las anteriores peticiones GET, aunque la comunidad *public* no pueda acceder a ningún elemento de la MIB.

El *get-response* es el siguiente



Es de la misma forma que en el apartado anterior cuando un equipo de la red 20.0.0.0/24 TRAB quiso acceder a un elemento de la MIB al cual no tenía permiso. En este caso pasa lo mismo solo que la comunidad *public* no tiene acceso de ningún tipo a la MIB.

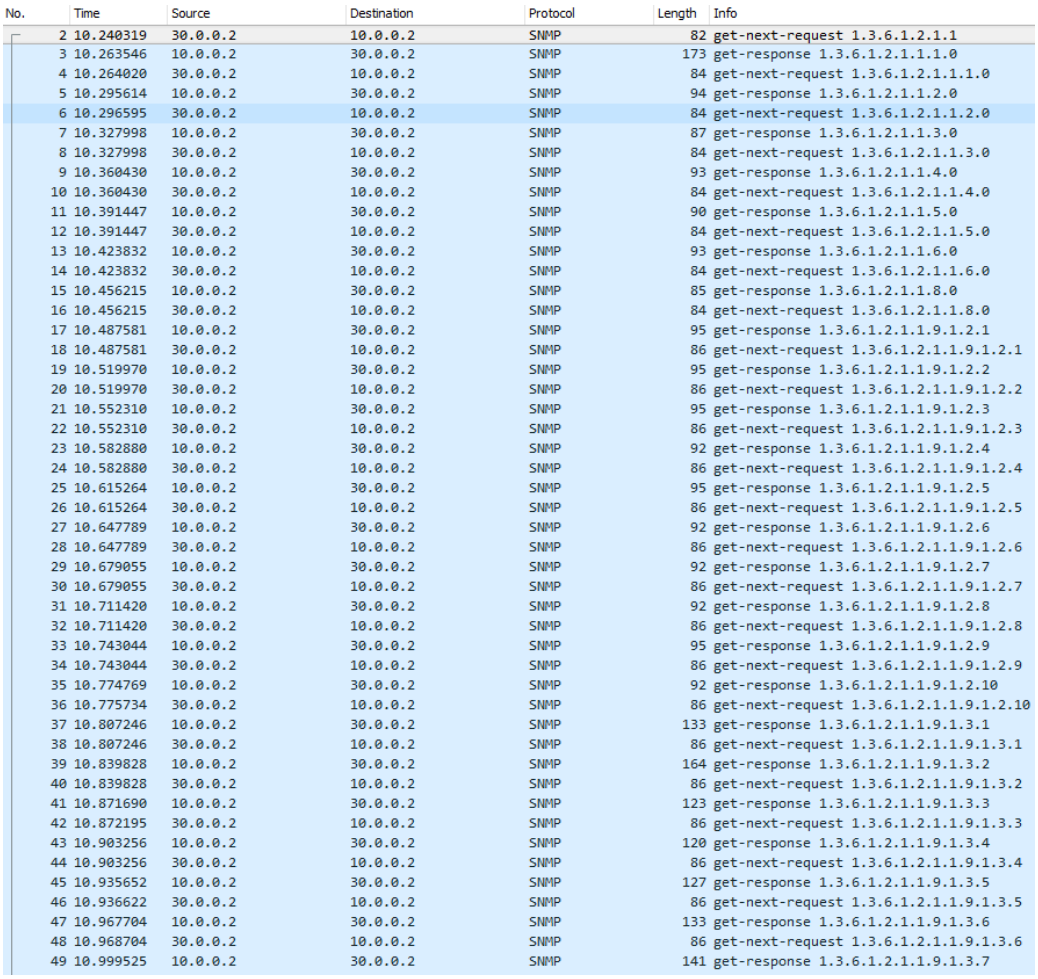
4. Para intercambiar más información de una sola vez que con *snmpget*, *Net-SNMP* nos proporciona la herramienta ***snmpwalk***. Utilízala desde ADMIN para conseguir toda una rama de la MIB del SERVIDOR. ¿Cuántos mensajes han sido necesarios para la comunicación? Comenta las diferencias entre la captura de un mensaje enviado con *snmpget* y otro enviado con *snmpwalk*.

   El comando que vamos a utilizar es el siguiente:

   ```
   snmpwalk -v2c -c admin 10.0.0.2 1.3.6.1.2.1.1
   ```

   En este caso, el OID que solicitamos es la correspondiente a *system*, es decir, toda la información del sistema.

   La captura de tráfico de la Figura 19 no se muestra en su totalidad. En esta comunicación se intercambian 77 mensajes.



No.	Time	Source	Destination	Protocol	Length	Info
2	10.240319	30.0.0.2	10.0.0.2	SNMP	82	get-next-request 1.3.6.1.2.1.1
3	10.263546	10.0.0.2	30.0.0.2	SNMP	173	get-response 1.3.6.1.2.1.1.1.0
4	10.264020	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.1.0
5	10.295614	10.0.0.2	30.0.0.2	SNMP	94	get-response 1.3.6.1.2.1.1.2.0
6	10.296595	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.2.0
7	10.327998	10.0.0.2	30.0.0.2	SNMP	87	get-response 1.3.6.1.2.1.1.3.0
8	10.327998	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.3.0
9	10.360430	10.0.0.2	30.0.0.2	SNMP	93	get-response 1.3.6.1.2.1.1.4.0
10	10.360430	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.4.0
11	10.391447	10.0.0.2	30.0.0.2	SNMP	90	get-response 1.3.6.1.2.1.1.5.0
12	10.391447	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.5.0
13	10.423832	10.0.0.2	30.0.0.2	SNMP	93	get-response 1.3.6.1.2.1.1.6.0
14	10.423832	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.6.0
15	10.456215	10.0.0.2	30.0.0.2	SNMP	85	get-response 1.3.6.1.2.1.1.8.0
16	10.456215	30.0.0.2	10.0.0.2	SNMP	84	get-next-request 1.3.6.1.2.1.1.8.0
17	10.487581	10.0.0.2	30.0.0.2	SNMP	95	get-response 1.3.6.1.2.1.1.9.1.2.1
18	10.487581	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.1
19	10.519970	10.0.0.2	30.0.0.2	SNMP	95	get-response 1.3.6.1.2.1.1.9.1.2.2
20	10.519970	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.2
21	10.552310	10.0.0.2	30.0.0.2	SNMP	95	get-response 1.3.6.1.2.1.1.9.1.2.3
22	10.552310	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.3
23	10.582880	10.0.0.2	30.0.0.2	SNMP	92	get-response 1.3.6.1.2.1.1.9.1.2.4
24	10.582880	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.4
25	10.615264	10.0.0.2	30.0.0.2	SNMP	95	get-response 1.3.6.1.2.1.1.9.1.2.5
26	10.615264	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.5
27	10.647789	10.0.0.2	30.0.0.2	SNMP	92	get-response 1.3.6.1.2.1.1.9.1.2.6
28	10.647789	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.6
29	10.679055	10.0.0.2	30.0.0.2	SNMP	92	get-response 1.3.6.1.2.1.1.9.1.2.7
30	10.679055	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.7
31	10.711420	10.0.0.2	30.0.0.2	SNMP	92	get-response 1.3.6.1.2.1.1.9.1.2.8
32	10.711420	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.8
33	10.743044	10.0.0.2	30.0.0.2	SNMP	95	get-response 1.3.6.1.2.1.1.9.1.2.9
34	10.743044	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.9
35	10.774769	10.0.0.2	30.0.0.2	SNMP	92	get-response 1.3.6.1.2.1.1.9.1.2.10
36	10.775734	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.2.10
37	10.807246	10.0.0.2	30.0.0.2	SNMP	133	get-response 1.3.6.1.2.1.1.9.1.3.1
38	10.807246	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.3.1
39	10.839828	10.0.0.2	30.0.0.2	SNMP	164	get-response 1.3.6.1.2.1.1.9.1.3.2
40	10.839828	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.3.2
41	10.871690	10.0.0.2	30.0.0.2	SNMP	123	get-response 1.3.6.1.2.1.1.9.1.3.3
42	10.872195	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.3.3
43	10.903256	10.0.0.2	30.0.0.2	SNMP	120	get-response 1.3.6.1.2.1.1.9.1.3.4
44	10.903256	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.3.4
45	10.935652	10.0.0.2	30.0.0.2	SNMP	127	get-response 1.3.6.1.2.1.1.9.1.3.5
46	10.936622	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.3.5
47	10.967704	10.0.0.2	30.0.0.2	SNMP	133	get-response 1.3.6.1.2.1.1.9.1.3.6
48	10.968704	30.0.0.2	10.0.0.2	SNMP	86	get-next-request 1.3.6.1.2.1.1.9.1.3.6
49	10.999525	10.0.0.2	30.0.0.2	SNMP	141	get-response 1.3.6.1.2.1.1.9.1.3.7

Figura 19: Tráfico generado por snmpbalk desde ADMIN.

Lo primero que salta a la vista es que no se producen los mismos mensajes que con un *snmpget*. En esta comunicación *get-request*, es sustituido por muchos *get-next-request*. Este mensaje permite leer el siguiente objeto de un árbol MIB sin necesidad de conocer el nombre. Es muy similar al tipo GET, pero esta operación devuelve el valor del próximo objeto en el árbol MIB.

```
▼ Simple Network Management Protocol
    version: v2c (1)
    community: admin
  ▼ data: get-next-request (1)
    ▼ get-next-request
        request-id: 37283363
        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)
```

Vemos que este primer mensaje va dirigido a ese subárbol de la MIB, el 1.3.6.1.2.1.1. El otro tipo de mensaje es uno ya estudiado que es el de tipo *get-response*.

```
Simple Network Management Protocol
    version: v2c (1)
    community: admin
  data: get-response (2)
    get-response
        request-id: 37283363
        error-status: noError (0)
        error-index: 0
      variable-bindings: 1 item
        1.3.6.1.2.1.1.1.0: 4c696e757820534552564552203 42e382e302d35342d6765...
            Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)
          Value (OctetString): 4c696e757820534552564552203 42e382e302d35342d6765...
              Variable-binding-string: Linux SERVER 4.8.0-54-generic #57~16.04.1-Ubuntu SMP Wed May 24 16:22:28 UTC 2017 x86_64
```

Esta respuesta es el valor del primer elemento del subárbol que le hemos solicitado. En este caso, es el elemento 1.3.6.1.2.1.1.1.0 que se corresponde con la descripción del sistema, *sysDescr*. Los campos que trae son los mismos que en los mensajes de este tipo ya estudiados, en el campo *value* trae un objeto de tipo *OctetString* que se corresponde con la descripción del equipo.

Un detalle interesante es que el siguiente *get-next-request* es solicitado al elemento del que acabamos de recibir el response, es decir, utiliza la respuesta anterior para elegir cuál es el elemento siguiente del árbol MIB por el que tiene que preguntar.

5. Ahora utiliza ***snmpwalk*** desde TRAB hacia SERVIDOR. Úsalo con una rama a la que tenga acceso y otra a la que no. ¿Qué sucede en casa caso? Comenta la captura de tráfico realizada.

Para este apartado, sabiendo que este equipo TRAB solo tiene acceso al subárbol MIB 1.3.6.1.2.1.1, tan sólo tenemos opción a hacer un comando snmpwalk que tenga éxito. El otro comando, el preparado para que la petición falle, lo haremos al subárbol MIB 1.3.6.1.2.1.2.

```
snmpwalk -v2c -c 10.0.0.2 1.3.6.1.2.1.1
```

```
  95 9857.872309 10.0.0.2   20.0.0.3   SNMP  171 get-response 1.3.6.1.2.1.1.1.0
  96 9857.872309 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.1.0
  97 9857.903934 10.0.0.2   20.0.0.3   SNMP   93 get-response 1.3.6.1.2.1.1.2.0
  98 9857.904915 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.2.0
  99 9857.936372 10.0.0.2   20.0.0.3   SNMP   86 get-response 1.3.6.1.2.1.1.3.0
 100 9857.936372 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.3.0
 101 9857.967795 10.0.0.2   20.0.0.3   SNMP   92 get-response 1.3.6.1.2.1.1.4.0
 102 9857.967795 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.4.0
 103 9858.000177 10.0.0.2   20.0.0.3   SNMP   89 get-response 1.3.6.1.2.1.1.5.0
 104 9858.000177 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.5.0
 105 9858.032843 10.0.0.2   20.0.0.3   SNMP   92 get-response 1.3.6.1.2.1.1.6.0
 106 9858.032843 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.6.0
 107 9858.064226 10.0.0.2   20.0.0.3   SNMP   84 get-response 1.3.6.1.2.1.1.8.0
 108 9858.064226 20.0.0.3   10.0.0.2   SNMP   83 get-next-request 1.3.6.1.2.1.1.8.0
 109 9858.096606 10.0.0.2   20.0.0.3   SNMP   94 get-response 1.3.6.1.2.1.1.9.1.2.1
 110 9858.096606 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.1
 111 9858.129006 10.0.0.2   20.0.0.3   SNMP   94 get-response 1.3.6.1.2.1.1.9.1.2.2
 112 9858.129006 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.2
 113 9858.160466 10.0.0.2   20.0.0.3   SNMP   94 get-response 1.3.6.1.2.1.1.9.1.2.3
 114 9858.161474 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.3
 115 9858.192849 10.0.0.2   20.0.0.3   SNMP   91 get-response 1.3.6.1.2.1.1.9.1.2.4
 116 9858.192849 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.4
 117 9858.225302 10.0.0.2   20.0.0.3   SNMP   94 get-response 1.3.6.1.2.1.1.9.1.2.5
 118 9858.225302 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.5
 119 9858.257115 10.0.0.2   20.0.0.3   SNMP   91 get-response 1.3.6.1.2.1.1.9.1.2.6
 120 9858.257115 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.6
 121 9858.289500 10.0.0.2   20.0.0.3   SNMP   91 get-response 1.3.6.1.2.1.1.9.1.2.7
 122 9858.289500 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.7
 123 9858.321880 10.0.0.2   20.0.0.3   SNMP   91 get-response 1.3.6.1.2.1.1.9.1.2.8
 124 9858.321880 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.8
 125 9858.353569 10.0.0.2   20.0.0.3   SNMP   94 get-response 1.3.6.1.2.1.1.9.1.2.9
 126 9858.354071 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.9
 127 9858.385546 10.0.0.2   20.0.0.3   SNMP   91 get-response 1.3.6.1.2.1.1.9.1.2.10
 128 9858.385546 20.0.0.3   10.0.0.2   SNMP   85 get-next-request 1.3.6.1.2.1.1.9.1.2.10
```

Sucede lo mismo que en el apartado anterior. El funcionamiento del mensaje de tipo *get-next-request* permite recuperar la información del elemento siguiente conociendo el anterior. Como la subred 20.0.0.0/24 a la que pertenece el equipo TRAB tiene acceso a este subárbol MIB no hay problema. Sin embargo, para el comando:

```
snmpwalk -v2c -c trab 10.0.0.2 1.3.6.1.2.1.2
```

Sucede lo siguiente:

```
181 9879.834030 20.0.0.3    10.0.0.2    SNMP    81 get-next-request 1.3.6.1.2.1.2
182 9879.852695 10.0.0.2    20.0.0.3    SNMP    81 get-response 1.3.6.1.2.1.2
```

Lo primero que salta a la vista es que el *get-response* lleva el mismo OID que se envió en el *get-next-request* esto ya hace pensar que algo ha ido mal. Si analizamos más a fondo ese *get-response*:

```
Simple Network Management Protocol
    version: v2c (1)
    community: trab
  data: get-response (2)
    get-response
        request-id: 659966736
        error-status: noError (0)
        error-index: 0
      variable-bindings: 1 item
        1.3.6.1.2.1.2: endOfMibView
            Object Name: 1.3.6.1.2.1.2 (iso.3.6.1.2.1.2)
          endOfMibView
            [Expert Info (Note/Response): endOfMibView]
                [endOfMibView]
                [Severity level: Note]
                [Group: Response]
```

Vemos que pasa algo parecido al error que encontramos cuando hacemos un *snmpget* de un elemento de la MIB al que no tenemos acceso. En este caso nos aparece en el mensaje el elemento *endOfMibView*. Nos indica que no hay más variables en esta vista de la MIB, que ya hemos pasado el extremo del árbol MIB al que podemos acceder. De esto nos informa por pantalla el comando *snmpwalk*:

```
root@TRAB: ~
root@TRAB:~# snmpwalk -v2c -c trab 10.0.0.2 1.3.6.1.2.1.2
IF-MIB::interfaces = No more variables left in this MIB View (It is past th
e end of the MIB tree)
root@TRAB:~#
root@TRAB:~#
```

6. Comprueba que ***snmpwalk*** no funciona desde el equipo PUBLIC. ¿Qué ocurre en la consola de este equipo?

Para este apartado vamos a reutilizar un comando de *snmpwalk* ya visto antes:

```
snmpwalk -v2c -c public 10.0.0.2 1.3.6.1.2.1.1
```

La pantalla del equipo PUBLIC muestra lo siguiente:

```
root@PUBLIC: ~
root@PUBLIC:~# snmpwalk -v2c -c public 10.0.0.2 1.3.6.1.2.1.1
SNMPv2-MIB::system = No more variables left in this MIB View (It is past the
 end of the MIB tree)
root@PUBLIC:~#
root@PUBLIC:~#
```

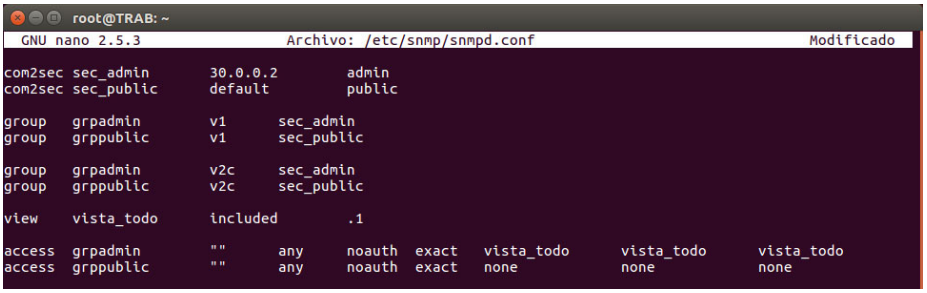
Por lo que al comprobar la captura no encontramos nada nuevo, sucede lo mismo que en el apartado anterior cuando un equipo TRAB intentaba acceder a un elemento del árbol MIB al que no tiene acceso.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000				60	<Ignored>
2	5.179000	40.0.0.2	10.0.0.2	SNMP	83	get-next-request 1.3.6.1.2.1.1
3	5.219236	10.0.0.2	40.0.0.2	SNMP	83	get-response 1.3.6.1.2.1.1

7. ¿El administrador de la red de esta empresa debe ser el único capaz de modificar los valores de la MIB del SERVIDOR e incluso de los equipos que pertenecen a dicha red? ¿Cómo se consigue esto? ¿Es una configuración óptima?

Del SERVIDOR depende. Algunas cosas sensibles sí, solo el administrador, pero de cosas más rutinarias de mantenimiento, la subred TRAB también debe tener algunos permisos. En el momento en el que configuramos el equipo SERVIDOR, dejamos esto ya preparado. Al módulo de control de acceso le añadimos una línea para la cual especificamos que para la vista que habíamos determinado, en ese caso todo el árbol MIB, fuese accedida por el grupo al que pertenece la ACL del equipo ADMIN (comunidad *admin*).

Para los demás equipos de la red, los de la subred 20.0.0.0/24, los equipos como TRAB aún no hemos realizado una configuración para su demonio *snmpd*, por ello, para conseguir que sólo el equipo ADMIN pueda modificar los elementos del árbol MIB, los equipos como TRAB deben tener un archivo de configuración como el siguiente:



Al no darle permisos para poderlo cambiar en local, garantizamos que solo el administrador de esta red podrá modificar cualquier elemento del árbol MIB.

La configuración óptima de este escenario, en cuanto a SNMPv2c, terminaría configurando el demonio *snmpd* del equipo ADMIN. Para mantener los criterios de seguridad establecidos, este equipo sólo podrá ser modificado en local. El aspecto que debe tener su fichero de configuración es el siguiente:

8. Utilizando la herramienta ***snmpset*** modifica algún valor de un elemento de la MIB de SERVIDOR desde ADMIN. Comenta la captura realizada.

Para este punto vamos a modificar el valor del campo *sysLocation* que nos indica dónde se encuentra físicamente el equipo. Para ello primeros vamos a comprobar donde está con un *snmpget*:

```
snmpget -v2c -c admin 10.0.0.2 sysLocation.0
```

El resultado es el siguiente:

```
root@ADMIN: ~
root@ADMIN:~# snmpget -v2c -c admin 10.0.0.2 sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: Valladolid, CYL
root@ADMIN:~#
root@ADMIN:~#
```

Ahora vamos a modificarlo por Salamanca para ello usamos el comando *snmpset* que tiene la misma sintaxis que *snmpget*. Solo cambia que al final del comando añadimos el tipo del objeto que vamos a modificar y el nuevo valor. Para este caso el comando sería así:

```
snmpset -v2c -c admin 10.0.0.2 system.sysLocation.0 s
"Salamanca, CYL"
```

La "s" indica que el tipo de objeto que vamos a modificar es de tipo String y lo que va detrás es el nuevo valor que le damos a ese elemento del árbol MIB. En este caso, "Salamanca, CYL". Si repetimos el comando *snmpget* de ese objeto vemos el cambio:

```
root@ADMIN: ~
root@ADMIN:~# snmpget -v2c -c admin 10.0.0.2 sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: Valladolid, CYL
root@ADMIN:~#
root@ADMIN:~# snmpset -v2c -c admin 10.0.0.2 system.sysLocation.0 s "Salamanca, CYL"
SNMPv2-MIB::sysLocation.0 = STRING: Salamanca, CYL
root@ADMIN:~#
root@ADMIN:~# snmpget -v2c -c admin 10.0.0.2 sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: Salamanca, CYL
root@ADMIN:~#
root@ADMIN:~#
```

Para profundizar más vamos a ver la captura de tráfico de este tipo de comando:

No.	Time	Source	Destination	Protocol	Length	Info
3	14.229017	30.0.0.2	10.0.0.2	SNMP	98	set-request 1.3.6.1.2.1.1.6.0
4	14.295851	10.0.0.2	30.0.0.2	SNMP	98	get-response 1.3.6.1.2.1.1.6.0

La comunicación genera dos mensajes, uno de tipo *set-request* y otro de tipo *get-response*. El contenido del primer mensaje es el siguiente:

```
Simple Network Management Protocol
    version: v2c (1)
    community: admin
  data: set-request (3)
    set-request
        request-id: 445202805
        error-status: noError (0)
        error-index: 0
      variable-bindings: 1 item
        1.3.6.1.2.1.1.6.0: 53616c616d616e63612c2043594c
            Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
            Value (OctetString): 53616c616d616e63612c2043594c
```

En forma es igual a un mensaje de tipo *get-request*, la diferencia es que el campo *Value* en un mensaje de ese tipo tiene como valor *null*. Sin embargo, en un mensaje de tipo *set-request,* tiene como valor el que hemos puesto nosotros a ese objeto del árbol MIB. El valor que vemos es la secuencia de Bytes que se corresponden con el valor del String "Salamanca, CYL".

El contenido del segundo mensaje, el de tipo *get-response* es el siguiente:

```
Simple Network Management Protocol
    version: v2c (1)
    community: admin
  data: get-response (2)
    get-response
        request-id: 445202805
        error-status: noError (0)
        error-index: 0
      variable-bindings: 1 item
        1.3.6.1.2.1.1.6.0: 53616c616d616e63612c2043594c
            Object Name: 1.3.6.1.2.1.1.6.0 (iso.3.6.1.2.1.1.6.0)
            Value (OctetString): 53616c616d616e63612c2043594c
```

Nos confirma el nuevo valor contestando como si en realidad hubiésemos hecho un mensaje con el comando *snmpget*. Por ello se muestra en la consola el mismo mensaje que si hubiese hemos hecho un GET de ese objeto a esa dirección.

9. Utilizando la herramienta ***snmpset*** modifica algún valor de un elemento de la MIB de ADMIN desde TRAB. ¿Qué sucede? ¿Pasa lo mismo enviando el comando desde ADMIN a TRAB? Comenta las capturas realizadas.

   Utilizaremos el comando visto antes:

```
snmpset -v2c -c admin 10.0.0.2 system.sysLocation.0 s
“Salamanca, CYL”
```

   Vemos en la pantalla de TRAB, tras haber pasado unos segundos aparece esto:

```
root@TRAB: ~
root@TRAB:~# snmpset -v2c -c trab 30.0.0.2 sysLocation.0 s "Salamanca, CYL"
Timeout: No Response from 30.0.0.2
root@TRAB:~#
root@TRAB:~#
```

   Analizando la captura de tráfico vemos que tardaba en responder porque estaba reintentando la petición hasta en 6 ocasiones. Como el equipo ADMIN está configurado para solo poder ser modificado desde un ámbito local, no puede ser ejecutado con éxito el comando anteriormente descrito.

```
11 66.999944   20.0.0.3   30.0.0.2   SNMP   97 set-request 1.3.6.1.2.1.1.6.0
12 68.001424   20.0.0.3   30.0.0.2   SNMP   97 set-request 1.3.6.1.2.1.1.6.0
13 69.003215   20.0.0.3   30.0.0.2   SNMP   97 set-request 1.3.6.1.2.1.1.6.0
14 70.005126   20.0.0.3   30.0.0.2   SNMP   97 set-request 1.3.6.1.2.1.1.6.0
15 71.007281   20.0.0.3   30.0.0.2   SNMP   97 set-request 1.3.6.1.2.1.1.6.0
16 72.008553   20.0.0.3   30.0.0.2   SNMP   97 set-request 1.3.6.1.2.1.1.6.0
```

10. Prueba la herramienta ***snmpbulkwalk***. ¿Representa alguna ventaja respecto a otras herramientas proporcionadas por *Net-SNMP*? Comenta la captura realizada.

    Vamos a utilizar este comando para recopilar toda la información del subárbol MIB 1.3.6.1.2.1.1. Para ello, desde el equipo ADMIN vamos a enviar a TRAB el siguiente comando:

```
snmpbulkwalk -v2c -c admin 20.0.0.3 1.3.6.1.2.1.1
```

    El resultado es el mismo que el del comando *snmpwalk,* pero se muestra mucho más rápido la información:

```
root@ADMIN: ~
root@ADMIN:~# snmpbulkwalk -v2c -c admin 20.0.0.3 1.3.6.1.2.1.1
SNMPv2-MIB::sysDescr.0 = STRING: Linux TRAB 4.8.0-54-generic #57~16.04.1-Ubuntu SMP Wed May 24 16:22:28 UT
C 2017 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (169490) 0:28:14.90
SNMPv2-MIB::sysContact.0 = STRING: root
SNMPv2-MIB::sysName.0 = STRING: TRAB
SNMPv2-MIB::sysLocation.0 = STRING: Unknown
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (22) 0:00:00.22
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
SNMPv2-MIB::sysORID.7 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.8 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.9 = OID: SNMP-NOTIFICATION-MIB::snmpNotifyFullCompliance
SNMPv2-MIB::sysORID.10 = OID: NOTIFICATION-LOG-MIB::notificationLogMIB
SNMPv2-MIB::sysORDescr.1 = STRING: The MIB for Message Processing and Dispatching.
SNMPv2-MIB::sysORDescr.2 = STRING: The management information definitions for the SNMP User-based Security
 Model.
SNMPv2-MIB::sysORDescr.3 = STRING: The SNMP Management Architecture MIB.
SNMPv2-MIB::sysORDescr.4 = STRING: The MIB module for SNMPv2 entities
SNMPv2-MIB::sysORDescr.5 = STRING: View-based Access Control Model for SNMP.
SNMPv2-MIB::sysORDescr.6 = STRING: The MIB module for managing TCP implementations
SNMPv2-MIB::sysORDescr.7 = STRING: The MIB module for managing IP and ICMP implementations
SNMPv2-MIB::sysORDescr.8 = STRING: The MIB module for managing UDP implementations
SNMPv2-MIB::sysORDescr.9 = STRING: The MIB modules for managing SNMP Notification, plus filtering.
SNMPv2-MIB::sysORDescr.10 = STRING: The MIB module for logging SNMP Notifications.
SNMPv2-MIB::sysORUpTime.1 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.2 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.3 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.4 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.5 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.6 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.7 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.8 = Timeticks: (21) 0:00:00.21
SNMPv2-MIB::sysORUpTime.9 = Timeticks: (22) 0:00:00.22
SNMPv2-MIB::sysORUpTime.10 = Timeticks: (22) 0:00:00.22
```

Analizando la captura de tráfico encontramos respuesta a este hecho:

No.	Time	Source	Destination	Protocol	Length	Info
4	71.384168	30.0.0.2	20.0.0.3	SNMP	82	getBulkRequest 1.3.6.1.2.1.1
5	71.401802	20.0.0.3	30.0.0.2	SNMP	364	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0 1.3.6.1.
6	71.401802	30.0.0.2	20.0.0.3	SNMP	86	getBulkRequest 1.3.6.1.2.1.1.9.1.2.3
7	71.423422	20.0.0.3	30.0.0.2	SNMP	446	get-response 1.3.6.1.2.1.1.9.1.2.4 1.3.6.1.2.1.1.9.1.2.5
8	71.423422	30.0.0.2	20.0.0.3	SNMP	86	getBulkRequest 1.3.6.1.2.1.1.9.1.3.3
9	71.444565	20.0.0.3	30.0.0.2	SNMP	572	get-response 1.3.6.1.2.1.1.9.1.3.4 1.3.6.1.2.1.1.9.1.3.5
10	71.445035	30.0.0.2	20.0.0.3	SNMP	86	getBulkRequest 1.3.6.1.2.1.1.9.1.4.3
11	71.465521	20.0.0.3	30.0.0.2	SNMP	241	get-response 1.3.6.1.2.1.1.9.1.4.4 1.3.6.1.2.1.1.9.1.4.5

Comprobamos que respecto al comando *snmpwalk* se reduce muchísimo el número de mensajes intercambiados para una comunicación de este tipo. De los 77 que necesito el comando *snmpwalk* a los 8 que ha necesitado el comando *snmpbulkwalk*. La razón de esto es la siguiente:

```
▾ Simple Network Management Protocol
    version: v2c (1)
    community: admin
  ▾ data: getBulkRequest (5)
    ▾ getBulkRequest
        request-id: 1735797644
        non-repeaters: 0
        max-repetitions: 10
      ▾ 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)
```

Figura 20: Mensaje *GetBulkRequest*.

El mensaje de tipo *GetBulkRequest* solicita el objeto 1.3.6.1.2.1.1 y la respuesta que recibe es un mensaje de tipo *get-response* con el siguiente contenido:

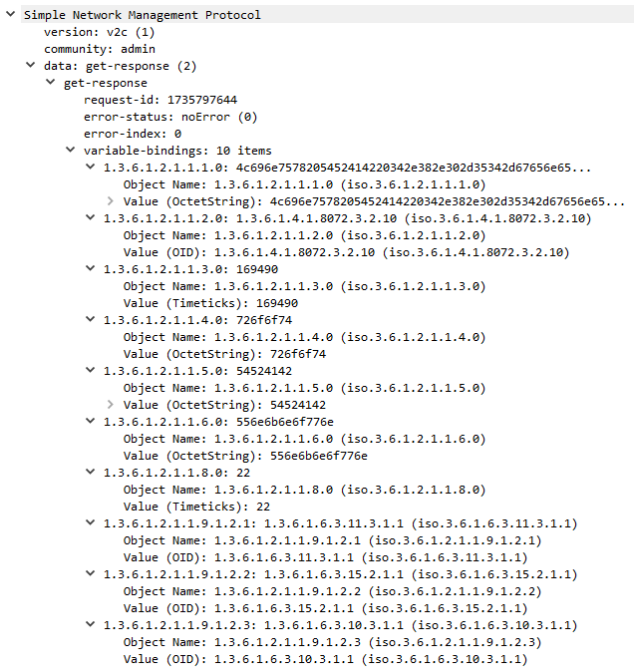


Figura 21: Mensaje *GetResponse* en respuesta a un *GetBulkRequest.*

Este mensaje no sólo contiene un ítem como hemos visto con todos los comandos hasta ahora. Esta respuesta contiene hasta 10 ítems. Este número se corresponde con el del mensaje de petición en el campo *max-repetitions*.

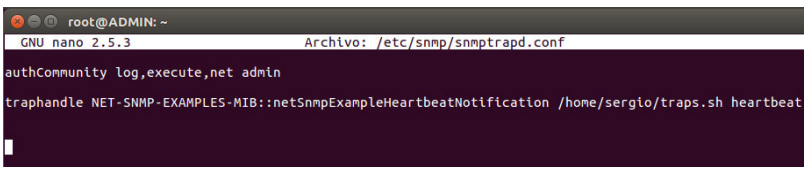
Como en una sola respuesta no ha entrado toda la información solicitada, el comando lanza otro *GetBulkRequest* desde el último objeto que entró en el mensaje de respuesta del destino de nuestro comando.

Generando aviso o *traps*

Llegados a este punto es necesaria la configuración de *traps*. Habilítalas de la forma adecuada en todos los equipos. Explica el demonio utilizado, en este caso ***snmptrapd*** y su fichero de configuración y utiliza la propiedad ***traphandle*** para asociar la llegada de un *trap* a un *script*.

Con *NET-SNMP* no es necesario realizar ninguna configuración para el envío de *traps* desde un equipo a otro, sin embargo, si es necesario realizar cambios para poder recibir este tipo de notificaciones. Para capturar los *traps* que nos envían tenemos el demonio *snmptrapd* que utiliza como fichero de configuración /etc/snmp/snmptrapd.conf. En él se describen las comunidades que escucharán *traps*. En este fichero también podemos configurar manejadores de *traps* que se ejecuten cada vez que llega una notificación asociada a un *script* o programa.

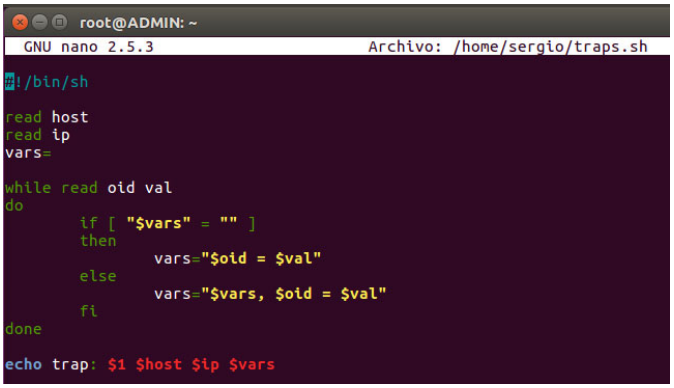
Un ejemplo de archivo de configuración es el siguiente:



En él creamos una comunidad para escuchar *traps* llamada admin y configuramos un manejador que ejecute el programa /home/sergio/traps.sh con el parámetro *heartbeat* cuando llegue un *trap* de tipo:

```
NET-SNMP-EXAMPLES-MIB::netSnmpExampleHeartbeatNotification
```

El contenido del *script* que se ejecutará con su llegada es:



Mostrará el mismo contenido que se muestra si activamos el parámetro *log* en la creación del *authCommunity* del fichero /etc/snmp/snmptrapd.conf.

1. Con la ayuda de la herramienta ***snmptrap*** envía un *trap* desde SERVIDOR a ADMIN. ¿Qué se muestra en pantalla en ADMIN? ¿Se corresponde con la captura correspondiente de tráfico de red? Comenta dicha captura.

   Para que admin pueda capturar la trama debe estar ejecutando el demonio *snmptrapd*, para ello ejecutamos este comando:

   ```
   snmptrapd -f -C -c /etc/snmp/snmptrapd.conf -Le
   ```

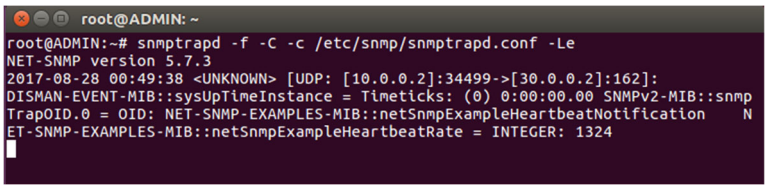
   Una vez hecho esto, el demonio se quedará escuchando a la espera de que le llegue algún *trap* al equipo ADMIN.

   Desde el equipo SERVIDOR vamos a mandar el siguiente *trap* utilizando la herramienta *snmptrap*:

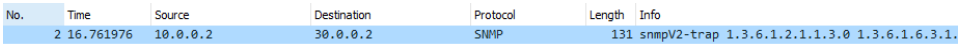
   ```
   snmptrap -v2c -c admin 30.0.0.2 0 NET-SNMP-EXAMPLES-
   MIB::netSnmpExampleHeartbeatNotification
   netSnmpExampleHeartbeatRate i 1234
   ```

   Cuando enviamos un mensaje de este tipo, solo se genera un paquete, el que envía el equipo SERVIDOR, no hay una respuesta por parte del equipo receptor del *trap*.

   En la pantalla del equipo ADMIN aparece esto:

   Toda esa información se corresponde con la enviada en el paquete, para comprobarlo vamos a analizar el tráfico de red generado:

   Comprobamos que sólo se envía un paquete con origen en la máquina SERVIDOR y con destino a ADMIN.

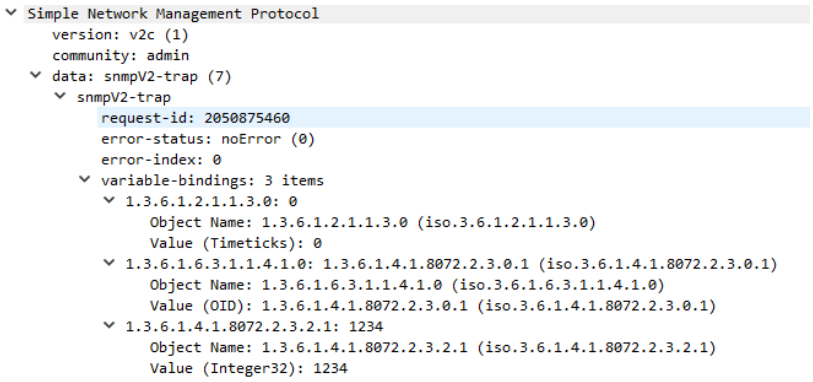


Figura 22: Mensaje *snmpv2-trap*.

Analizando más a fondo el paquete vemos que es de tipo *snmpV2-trap* y que ha sido exitoso puesto que no lleva ningún error en las variables destinadas a ello. En *variable-bindings* viajan 3 ítems. El primero es el valor que le hemos colocado detrás de la IP y se corresponde con el *uptime*. El segundo es la MIB o el OID al que pertenece el objeto de la notificación. El siguiente parámetro que aparece es el propio objeto del *trap*, en nuestro caso el objeto es *netSnmpExampleHeartbeatRate*. Los últimos parámetros del comando son el último ítem de *variable-bindings*, en nuestro ejemplo la "i" significa que el tipo del valor del objeto es Integer32 y 1234 es el valor que enviamos.

Por tanto, podemos afirmar que lo que aparece en la pantalla de ADMIN se corresponde con lo enviado por SERVIDOR y lo que la captura reconoce.

2. Para el administrador de la red ¿es interesante que reciba *traps* de todos los equipos o solo del SERVIDOR?

   Depende del grado de información que se necesite conocer de los equipos de la red. Si no es importante que un equipo de un trabajador informe al administrador, por ejemplo, que ha reiniciado el ordenador, no es recomendable que cargue la red con ese tráfico innecesario. Pero que un router se reinicie sí es una información que es importante para el administrador luego debe de estar preparado para enviar *traps* y hacerlo a ADMIN.

3. Prepara ambos *routers*, **R1** y **R2**, para el envío de *traps* importantes a **ADMIN**. ¿Cuáles consideras más importantes?

   Para que los routers puedan enviar *traps* de forma automática al equipo ADMIN debemos seguir esta configuración:

```
#conf t
#snmp-server enable traps
#snmp-server trap-source Loopback 0
#snmp-server enable traps config
#snmp-server enable traps snmp
#snmp-server enable traps entity
#snmp-server host 30.0.0.2 version 2c admin config snmp entity
```

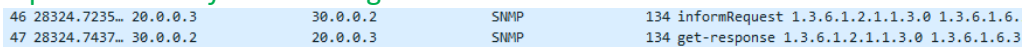
Después de esto, ambos routers enviarán los *traps* directamente al equipo ADMIN cuando suceda un hecho importante. Los más importantes son los de configuración, los del protocolo SNMP y los de modificación de la MIB.

4. ¿En qué se diferencia un mensaje ***inform*** de uno *trap*? Lanza un *inform* desde **TRAB** a **ADMIN**. Comenta las diferencias en la captura.

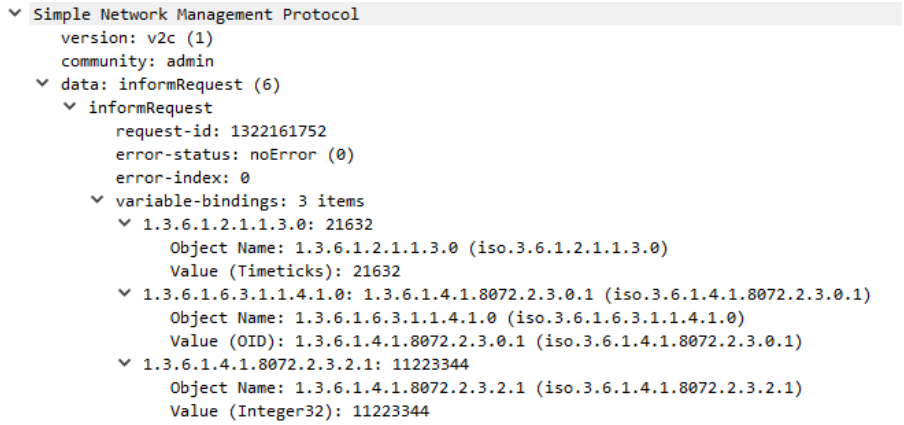
   Un *inform* es enviado por un agente a un NMS (gestor) con el objetivo de notificar algo. Una vez enviado queda a la espera de recibir una confirmación por parte del NMS de que lo ha recibido. Sin embargo, un *trap* es lanzado por un agente a un NMS y este no es contestado ni se espera respuesta. El equipo receptor es el encargado de tratar dicho *trap* como sea necesario.

   Otra diferencia se encuentra a la hora de utilizar la herramienta *snmptrap*. Para lanzar un *inform* desde el agente hay que añadir al comando el parámetro -Ci, de este modo el *trap* se convierte en un mensaje de este tipo.

   La captura de un *inform* es la siguiente:

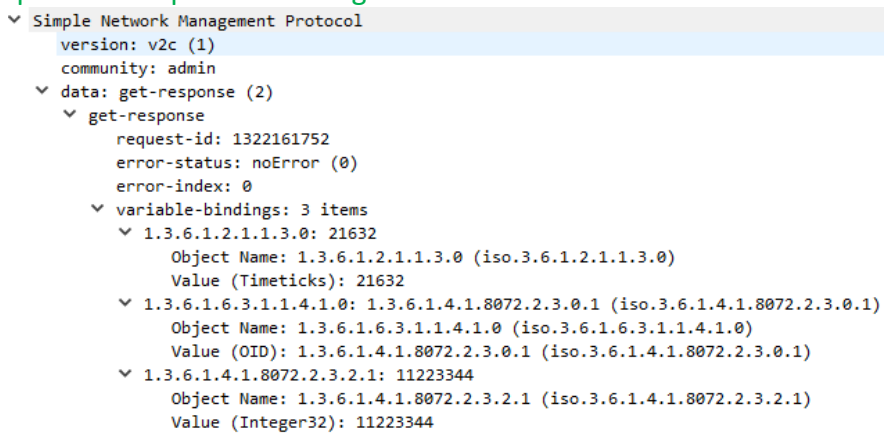


   Como podemos ver hay dos mensajes, un *InformRequest* y otro de tipo *get-response*, este último es la confirmación por parte del equipo destino del *inform* de que lo ha recibido. Analizando más a fondo el primer paquete:



   Comprobamos que el paquete es idéntico al de un *trap*, el espacio reservado para *variable-bindings* contiene los mismos tres campos que el *trap*.

   El paquete de respuesta es el siguiente:



   Es una confirmación de todos los campos enviados en la petición por eso tiene 3 ítems en *variable-bindings* y ambos tres coinciden con los valores del primer paquete analizado.

5. Equipos adquiridos e integrados

En la Tabla 2 se muestran los equipos que han sido integrados en este proyecto junto con sus principales características y el rol que desempeñan en el desarrollo de este.

Figura 23: Cisco ASA 5506-X with FirePOWER Services.

Firewall adaptativo y orientado a amenazas de la industria diseñado para una nueva era de amenazas y protección avanzada contra malware. Ofrece una defensa de amenazas integrada para todo el ataque, antes, durante y después de un ataque. Financiado con el proyecto de innovación docente ID2017/029 [5].

Figura 24: Cisco Catalyst 2960L-16TS-LL.

Conmutador Gigabit Ethernet que opera con el software Cisco IOS® y admite la gestión simple de dispositivos y la gestión de red. Conmutador administrado que ofrece funciones avanzadas de nivel 2. Esta serie ofrece seguridad de red mejorada, confiabilidad de la red y eficiencia operativa. Este conmutador pertenece a la serie *classic 2960* uno de los equipos más robustos de la marca CISCO. Financiado con el proyecto de innovación docente ID2017/029 [5]

Figura 25: Cisco Small Business SG350-10P

Los switches de esta serie proporcionan características de seguridad avanzadas, así como funcionalidades de nivel 2 y 3. El hecho de proporcionar facilidades de nivel 3 es la razón por la que ha sido elegido para este proyecto a diferencia con el conmutador de la serie classic 2969 que solo tiene facilidades de nivel 2. Financiado con el proyecto de innovación docente ID2017/029 [5]

Figura 26: Router CISCO 7200 Serie VXR.

Otorga los servicios de red básicos para construir a partir de este Router una estructura de red más compleja. Entre sus tareas están; encaminamiento de redes, servicio de parámetros de red mediante DHCP, etiquetado 802.1Q, aplicar reglas NAT e implementar listas de acceso para el tráfico en la red.Dispone de 3 puertos GigabitEthernet con 2 tipos de conexión a elegir entre uno (RJ-45/SFP), cuenta con 2 fuentes de alimentación, una principal y otra secundaria. La conexión por terminal se realiza mediante RJ-45. Equipo reacondicionado cedido por el CPD de la Universidad de Salamanca.

Figura 27: Wireless LAN Controller 4400 Series Model 4402 14 AP.

Pese a no ser de los últimos modelos realiza una labor optima a la hora de desplegar redes inalámbricas de prácticas para los alumnos. Será el encargado de administrar varias redes inalámbricas con diferentes configuraciones. Dispone de servicios propios DHCP, soporte para VLAN, base de datos local para servicios AAA, QoS, etc. Se requiere un cable serial RS-232 para su administración por consola y posee 2 puertos SFP de enlaces al sistema de distribución. Equipo reacondicionado cedido por el CPD de la Universidad de Salamanca.

Figura 28: Cisco Aironet 1000 Series Lightweight Access Points.

Es el dispositivo encargado de desplegar las redes inalámbricas, consta de 2 antenas planas en el eje vertical, admite Power-over-Ethernet, soporta 802.11a,b,g. Equipo reacondicionado cedido por el CPD de la Universidad de Salamanca.

Figura 29: SUN Microsystems Untra 20M2.

2 PC SUN Microsystems Ultra 20M2 con procesador Dual Core AMD Opteron 1218, 2600MHz, 2 procesadores y 4GB de memoria RAM. Uno de ellos con Windows 10 y el otro con Debian-10. Estos equipos se utilizaran como equipos finales de nuestra red. Ambos tienen 2 interfaces de red que se utilizan tanto para conectar con el laboratorio de redes como con a la red corporativa de la USAL.

Tabla 2: Equipos adquiridos e integrados.

6. Conclusiones y ampliaciones futuras

Con este proyecto se han elaborado materiales didácticos para iniciar a los estudiantes en la gestión de redes. El material didáctico elaborado se compone de un apartado de teoría donde se presentan los conceptos de la gestión de redes con SNMP, completándose con la elaboración de figuras que expliquen de forma visual los conceptos considerados. El apartado teórico se completa con los correspondientes manuales sobre cómo instalar y configurar SNMP en las diferentes plataformas con las que se va a trabajar en el guion: CISCO IOS, Linux y Windows. Tras el apartado de teoría, le siguen un guion donde el alumno será guiado para la correcta solución a determinadas situaciones en la gestión de la red y un guion con soluciones que resuelve detalladamente las cuestiones presentadas.

Para cumplir con el objetivo de trabajar con equipos reales y no solo en entornos de simulación se han realizado configuraciones de gestión de redes de dificultad creciente en el prototipo de laboratorio de redes que ha sido ampliado con equipos reacondicionados para este proyecto de innovación.

Consideramos que el principal grado de innovación reside en el planteamiento de unas prácticas que sin soluciones del tipo de las que planteamos en nuestra propuesta serían imposibles de realizar sin comprometer la infraestructura de la red de la USAL.

En el trabajo realizado en este proyecto han quedado varias líneas abiertas como son la instalación de un software de gestión de redes con entorno gráfico e incluso la realización de un desarrollo propio que cubra esta necesidad.

Bibliografía

[1] «Servidor del Departamento de Informática y Automática,» Universidad de Salamanca, [En línea]. Available: http://diaweb.usal.es. [Último acceso: 30 06 2021].

[2] «RFC 1157 - A Simple Network Management Protocol (SNMP),» 1990.

[3] «GNS3,» [En línea]. Available: https://www.gns3.com/. [Último acceso: 29 06 2021].

[4] «RFC 3415 - View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP),» 2002.

[5] Á. M. Moreno-Montero, F. J. Blanco Rodríguez, M. B. Curto Diego, V. Moreno Rodilla y M. J. Polo Martín, «Diseño de materiales prácticos para la asignatura Seguridad en Sistemas Informáticos del grado en Ingeniera Informática,» Salamanca, 2018.

[6] «RFC 1441 - Introduction to version 2 of the Internet-standard Network Management Framework,» 1993.

[7] «RFC 2578 - Structure of Management Information Version 2 (SMIv2),» 1999.

[8] «RFC 2579 - Textual Conventions for SMIv2,» 1999.

[9] «RFC 2580 - Conformance Statements for SMIv2,» 1999.

[10] «RFC 1445 - Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2),» 1993.

[11] «RFC 1446 - Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2),» 1993.

[12] «RFC 1447 - Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2),» 1993.

[13] «RFC 3416 - Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP),» 2002.

[14] «RFC 3417 - Transport Mappings for the Simple Network Management Protocol (SNMP),» 2002.

[15] «RFC 3418 - Management Information Base (MIB) for the Simple Network Management Protocol (SNMP),» 2002.

[16] «RFC 1451 - Manager-to-Manager Management Information Base,» 1993.

[17] «RFC 1901 - Introduction to Community-based SNMPv2,» 1996.

[18] «RFC 3584 - Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework,» 2003.

[19] «RFC 1909 - An Administrative Infrastructure for SNMPv2,» 1996.

[20] «RFC 1910 - User-based Security Model for SNMPv2,» 1996.

[21] «RFC 3410 - Introduction and Applicability Statements for Internet Standard Management Framework,» 2002.

[22] «RFC 3411 - An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,» 2002.

[23] «RFC 3412 - Message Processing and Dispatching for the Simple Network Management Protocol (SNMP),» 2002.

[24] «RFC 3413 - Simple Network Management Protocol (SNMP) Applications,» 2002.

[25] «RFC 3414 - User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3),» 2002.