Trabajo Fin de Grado

DESARROLLO DE UNA HERRAMIENTA EN PYTHON PARA LA MONITORIZACIÓN DE ACTIVIDAD DE USUARIOS DE WHATSAPP

Autor:

Fernando Alegre Ojer

Director:

Álvaro Alesanco Iglesias

Escuela de Ingeniería y Arquitectura
Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
Septiembre de 2014

Agradecimientos

Varias han sido las personas que me han ayudado en la realización de este trabajo, por ello escribo estas líneas de agradecimiento a su apoyo durante la realización del mismo.

A Álvaro Alesanco, por su ayuda y sus consejos durante el desarrollo del proyecto, presentándome varias opciones cuando me estancaba y gracias a ellas me permitían volver a avanzar, por sus comentarios y correcciones durante la redacción de la memoria y por estar disponible en todo momento.

A mis padres y mi hermana, por apoyarme durante todos estos años, sufriendo y alegrándose conmigo, soportando mis cambios de humor cuando las cosas no salían bien y siendo una fuente constante de ánimo.

Al resto de mi familia, en especial a mi abuela, por preocuparse siempre por mi e interesarse por mi trabajo.

A los amigos y compañeros de clase, en especial a Álex, Miguel Ángel, Miguel, Javi, Jorge, Nacho y Alberto por estar siempre en disposición de ayudarme y por aguantarme diariamente.

Al resto de mis amigos, por interesarse por mi proyecto y animarme en todo momento a terminarlo.

DESARROLLO DE UNA HERRAMIENTA EN PYTHON PARA LA MONITORIZACIÓN DE ACTIVIDAD DE USUARIOS DE WHATSAPP

RESUMEN

Este Trabajo de Fin de Grado consiste en el desarrollo de una plataforma que permite la monitorización de la actividad de los usuarios en la aplicación de mensajería instantánea para smartphones *Whatsapp*. Los módulos principales de la plataforma son: un Agente SNMP que ejecuta la aplicación utilizada para la monitorización, ambos escritos en el lenguaje de programación *Python*, y un Gestor SNMP que ejecuta el software de *MG-SOFT (MIB Browser, MIB Builder, MIB Compiler)* utilizado en Gestión de Red.

La aplicación utilizada para la monitorización recoge las horas de última conexión de los usuarios y las envía mediante el protocolo de gestión *SNMP* al agente. El agente SNMP, que implementa una MIB privada creada con el programa MIB Builder, es gestionado a través del programa MIB Browser, de modo que permita su acceso remotamente. Esta MIB privada consiste en una tabla que almacena los distintos números telefónicos, los nombres de sus propietarios, sus fechas de últimas conexiones y el número de veces que han accedido a Whatsapp durante los últimos 5 minutos.

Por último, la aplicación de generación de gráficos en red *Cacti* nos permite una representación temporal de los datos almacenados en el agente.

DEVELOPMENT OF A PYTHON TOOL FOR MONITORING WHATSAPP USER ACTIVITY

SUMMARY

This Final Degree Project consists in developing a tool that allows the monitoring of *Whatsapp* users activity. Their main modules are a SNMP Agent and the monitoring application. They both are written in *Python*. Besides the *MG-SOFT (MIB Browser, MIB Builder, MIB Compiler)* sofware used in network monitoring.

The application used for monitoring collects last hour connection and sends them through the management protocol *SNMP* to the SNMP Agent. This Agent, that implements a private MIB created with MIB Builder software, is managed through the MIB Browser to get remote access. This private MIB is a table that stores the different phone numbers, their owner's name, date of last connection and the number of times connected in last 5 minutes.

Finally, *Cacti*, that is a network graph generator, allows us a temporary representation of the data stored in the Agent.

Índice general

Índice de figuras

Capítulo 1

Introducción

Hoy en día la comunicación electrónica se ha convertido en una parte esencial en la vida de las personas. El avance de la tecnología ha permitido la integración de Internet con los teléfonos móviles aparenciendo así los *smarthphones*. La proliferación de estos terminales ha favorecido el desarrollo de una gran cantidad de aplicaciones de mensajería instantánea que sustituyen a los antiguos mensajes cortos o *SMS*. *WhatsApp* se convirtió en la pionera de estas aplicaciones siendo la que actualmente ostenta el mayor número de usuarios, superando en abril de 2014 los 500 millones. Con tal cantidad de personas que utilizan la aplicación a diario cabría suponer que cuenta con unas medidas de seguridad suficientes que protejan la intimidad de sus clientes. En sus orígenes, enero de 2009, la aplicación enviaba los mensajes sin cifrar de modo que cualquier persona, con unos mínimos conocimientos de seguridad, que estuviera conectada a una red Wifi era capaz de interceptarlos y leerlos comprometiendo la privacidad del resto de las personas conectadas a la misma. En mayo de 2012, una actualización solucionaba el anterior problema cifrando todas las conversaciones. Desde entonces hasta ahora, la compañía ha ido publicando varias actualizaciones para resolver algunos problemas de seguridad tales como la suplantación de usuarios, cambiar el estado de otros usuarios, etc.

Otro de los problemas que pueden comprometer la privacidad es que simplemente sabiendo el número de teléfono de una persona se puede acceder

a su foto de perfil y saber la última vez que accedió a la aplicación. Actualmente existe la posibilidad en la propia aplicación de ocultar la última hora de conexión pero muy poca gente lo tiene activado.

Un sistema de monitorización de red consiste en el uso de un terminal que remotamente monitoriza la actividad un dispositivo. Para ello, habitualmente se utiliza el protocolo de gestión de red SNMP que permite obtener datos tales como el ancho de banda, la temperatura del equipo, el número de paquetes enviados o recibidos por un interfaz y enviarlos al administrador de esa red (vía email, teléfono móvil o alarmas). La integración de este protocolo junto con la herramienta Cacti permite la generación de gráficos de red de estos datos.

Como hemos comentado, en WhatsApp es posible saber si un usuario se encuentra en línea o no, por lo tanto ¿Y si se desarrollara un sistema para que de forma automática obtuviera la hora de la última conexión, se integrara en un sistema de gestión SNMP y se realizara una representación gráfica de los resultados obtenidos con Cacti? Se podría monitorizar diariamente a los usuarios quedando su privacidad expuesta...

En resumen, esta plataforma quiere mostrar que es posible realizar un registro de la actividad de un determinado número de teléfono combinando la aplicación WhatsApp con el protocolo de gestión SNMP y la herramienta de generación de gráficos de red Cacti.

1.1 Motivación y objetivos

El presente Trabajo de Fin de Grado tiene como objetivo el desarrollo de un agente basado en la arquitectura SNMP capaz de monitorizar la actividad de los usuarios de WhatsApp.

Para lograr el objetivo, el trabajo se divide en dos bloques:

- El primero consiste en el desarrollo de una aplicación que permite obtener la hora de la última conexión del número deseado y de un agente SNMP capaz de almacenar los resultados.

- El segundo se conforma de un gestor SNMP que permite tanto configurar como monitorizar al agente.

El primer bloque consiste en dos scripts escritos en el lenguaje de programación *Python*. El segundo está compuesto por dos softwares que permiten la configuración (MIB Browser) y la monitorización (Cacti).

Las tareas necesarias para conseguirlo son:

- El desarrollo de un sistema de monitorización de actividad de usuarios de WhatsApp basado en Python.
- La intregación del anterior sistema con la arquitectura de gestión SNMP.
- La integración de estos dos bloques con la herramienta de generación de gráficos en red Cacti.

1.2 Software utilizado

Para el desarrollo de la plataforma se utilizaron los siguientes elementos:

- **Python:** Es el lenguaje de programación utilizado en el desarrollo de los scripts. Elegimos este lenguaje por su fácil aprendizaje y por ser ampliamente utilizado en la creación de herramientas de seguridad.
- **IDLE Python:** Es el entorno de desarrollo de *Python*, la interfaz donde se realiza la programación de los scripts.
- **Oracle VM VirtualBox:** Programa utilizado para trabajar con máquinas virtuales en Windows. Se utilizó una distribución de Linux, *Debian*, para el desarrollo y ejecución tanto del agente como del programa que obtenía los registros de actividad de los usuarios.
- **MIB Browser:** Software de MG-SOFT que permite supervisar y gestionar cualquier dispositivo SNMP en la red utilizando los protocolos SNMPv1, SNMPv2c y SNMPv3 sobre las redes IPv4 e IPv6.

- **MIB Builder:** Software de MG-SOFT que permite el diseño y edición de módulos MIB.
- **MIB Compiler:** Software de MG-SOFT que permite la validación de módulos MIB.
- **Cacti:** Herramienta desarrollada en PHP para la generación de gráficos en red.

1.3 Organización de la memoria

El contenido de la memoria se estructura de la siguiente forma:

- En el **Capítulo 1** se realiza una breve introducción del TFG, el objetivo final que quiere alcanzar y el software que lo ha permitido.
- El **Capítulo 2** está dedicado a las herramientas utilizadas como son WhatsApp, introduciendo la aplicación y sus debilidades, la arquitectura de gestión de red SNMP, exponiéndola brevemente y la herramienta de generación de gráficos en red Cacti.
- En el **Capítulo 3** se presenta el escenario del proyecto, mostrando en detalle los bloques que lo forman.
- En el **Capítulo 4** se enumeran los pasos a seguir hasta la generación de las gráficas y se muestran los resultados obtenidos.
- En el **Capítulo 5** se expone información personal del trabajo.

Además, en la parte final de la memoria pueden encontrarse los siguientes anexos con información complementaria a la definida en la memoria principal:

- En el **Anexo A** se describen los acrónimos utilizados en la memoria.
- En el **Anexo B** se exponen de forma más detallada algunos aspectos de la arquitectura SNMP.

- En el **Anexo C** se adjunta la MIB desarrollada.
- En el **Anexo D** se explica el proceso de instalación de la plataforma Yowsup.
- En el **Anexo E** se adjunta el archivo XML completo que Cacti utiliza para recoger los datos.
- En el **Anexo F** se muestra el proceso de configuración de Cacti.

Capítulo 2

Materiales y herramientas utilizadas

2.1 WhatsApp

WhatsApp es un software propietario multiplataforma de mensajería instantánea para smartphones utilizado para la transmisión de datos. Además de aprovechar la mensajería en modo texto, los usuarios pueden crear grupos, de hasta 50 integrantes, y enviar entre ellos un número ilimitado de imágenes, video y audio, así como la localización del usuario si el terminal dispone de GPS. La aplicación utiliza la red de datos del dispositivo móvil en el que se esté ejecutando o la red Wifi a la que esté conectada, por lo tanto funciona conectada a Internet, a diferencia de los servicios tradicionales de mensajes cortos o el sistema de mensajería multimedia.

La aplicación está disponible para los sistemas operativos *iOS*, *Android*, *Windows Phone*, *BlackBerry OS*, *Symbian*, y *Asha*.

2.1.1 Protocolo de nivel de aplicación: FunXMPP

WhatsApp utiliza una versión propia que consiste en una modificación del protocolo de aplicación *XMPP (eXtensible Messaging and Presence Protocol)* llamada *FunXMPP*.

XMPP es un protocolo abierto, basado en el estándar XML para el intercambio en tiempo real de mensajes y presencia entre dos puntos en Internet. La principal aplicación de la tecnología XMPP es una plataforma extensible de mensajería y una red de MI (Mensajería Instantánea).

Los datos de FunXMPP son enviados a bin-short.whatsapp.net bajo el puerto 5222.

Las principales características de este protocolo son:

- **Abierto:** Se trata de un protocolo gratuito, abierto al público y comprensible.
- **Libre:** Ya que se puede ver cómo funciona y el usuario puede trabajar con él.
- **Extensible:** Al usar el lenguaje XML, cualquiera puede extender el protocolo de XMPP para una funcionalidad personalizada.
- **Descentralizado:** Cualquier persona puede montar su propio servidor de XMPP con libertad de uso.
- **Seguro:** Ya que soporta seguridad en la capa de transporte y cualquier servidor de XMPP puede ser aislado de la red pública XMPP. Utiliza TSL para las comunicaciones cliente-servidor. Además, está en desarrollo una seguridad más robusta gracias al uso de SASL y contraseñas de sesión.

Los tipos de etiquetas XML para transmitir mensajes son los siguientes:

- **Stream:** Contiene una secuencia de mensajes en XMPP.
- **Message:** Envía mensajes entre usuarios.
- **Presence:** Informa de la presencia de usuarios con metadatos como hora de conexión, desconexión, estado o nombre.
- **Iq:** Se usa para transmitir comandos internos como nueva conexión, informe de errores, y demás aspectos de señalización.
- **Xep:** Conjunto de especificaciones que extienden las diferentes etiquetas que el protocolo soporta. Es donde se definen cosas como la seguridad, intercambio de fichero, etc.

Las RFCs que definen el actual protocolo XMPP son las siguientes: RFC 3920, RFC 3921, RFC 3922 y RFC 3923.

Al ser una aplicación para móviles, WhatsApp ha reducido el tamaño de los mensajes y la carga del protocolo a lo mínimo, dando lugar al protocolo FunXMPP.

En la *Figura 2.1* se representa un mensaje en cada uno de los dos protocolos, mostrando así sus diferencias.

MENSAJE DEL PROTOCOLO XMPP

```
<message from="01234567890@s.whatsapp.net"
         id="1339831077-7"
         type="chat"
         timestamp="1339848755">
    <notify xmlns="urn:xmpp:whatsapp"
            name="NcN" />
    <request xmlns="urn:xmpp:receipts" />
    <body>Hello</body>
</message>
```

MENSAJE DEL PROTOCOLO FunXMPP

```
<\x5d \x38="01234567890@\x8a"
      \x43="1339831077-7"
      \xa2="\x1b"
      \x9d="1339848755">
   <\x65 \xbd="\xae"
         \x61="NcN" />
   <\x83 \xbd="\xad" />
   <\x16>Hello</\x16>
</\x5d>
```

Figura 2.1: Diferencias entre los protocolos XMPP y FunXMPP

Para ello hay una serie de palabras reservadas utilizadas (como message, from, body etc.) que se sustituyen por un byte, logrando reducir considerablemente la carga. FunXMPP utiliza una tabla hash de conversión para todas las palabras reservadas.

2.1.2 Arquitectura

Generalmente, XMPP se implementa y se usa como una arquitectura cliente-servidor descentralizada, pero puede emplearse XMPP para establecer una comunicación directa, de extremo a extremo peer-to-peer (P2P), entre los clientes.

Cuando enviamos un mensaje XMPP a algún contacto en otro dominio, el cliente XMPP se conecta a nuestro servidor XMPP, y éste se conecta directamente al servidor XMPP de nuestro contacto, sin realizar múltiples saltos. Una vez entregado el mensaje al servidor, permanece en ellos durante un "período de tiempo corto". Si la otra persona no está en línea, el mensaje se queda en el servidor de la compañía hasta que pueda ser entregado durante 30 días. Si no ha podido entregarse, se elimina del servidor. Esta implementación de la arquitectura es mucho más segura porque previene la suplantación de identidad, y hasta cierta punto, el spam.

En cuanto a los protocolos de transporte que sustentan la comunicación en XMPP, normalmente tenemos las conexiones puras de TCP, aunque podrían emplearse otros protocolos que se basan en usos muy particulares del protocolo de aplicación HTTP.

2.1.3 Seguridad

El protocolo FunXMPP incluye el protocolo de seguridad TLS (Transport Layer Security) utilizado también para transacciones electrónicas seguras. El problema radicaba en las primeras implementaciones ya que estaba configurado por defecto para NO utilizar cifrado, con lo que todos los datos intercambiados estaban en texto llano, accesibles a cualquiera con programas para capturar paquetes de datos. Tras una serie de críticas acerca de su seguridad, WhatsApp incluyó cifrado en sus mensajes. La solución al cifrado de WhatsApp para los móviles con sistema operativo Android fue tomar el IMEI (un código de identificación único para cada teléfono), invertirlo y aplicarle la función hash MD5 (Message-Digest Algorithm 5). En el caso de iOS el cifrado se obtuvo haciendo el mismo hash MD5 de la dirección MAC del terminal repetida dos veces. El resultado era la clave de cifrado, en cuanto al nombre de usuario, el número de teléfono.

Este sistema de generación de claves era altamente inseguro ya que cualquier persona con acceso a un teléfono ajeno puede obtener su IMEI sólo con pulsar la secuencia *#06#.

Actualmente ya no se usa el IMEI o la MAC (códigos que identifican cada teléfono) como contraseñas para el cifrado, sino que se utiliza el protocolo SSL para la comunicación entre el servidor de WhatsApp y el smartphone, es decir, un protocolo criptográfico para realizar conexiones seguras entre cliente y servidor.

2.1.4 Registro y autenticación

Para poder utilizar WhatsApp es necesario realizar un registro del número de teléfono en el servicio. Durante este registro se establece la clave de cifrado de los mensajes. Esta clave la envía el servidor como respuesta al código recibido en un SMS (mensaje corto que nos envía el servidor de WhatsApp tras la petición de registro). Un vez realizado el registro, la aplicación realiza la autenticación correspondiente previa al envío de datos. Tras el registro y autenticación el usuario es capaz de enviar mensajes a través de la aplicación cuyos datos están encriptados.

En la *Figura 2.2* se representa el flujo de datos del registro y la autenticación.

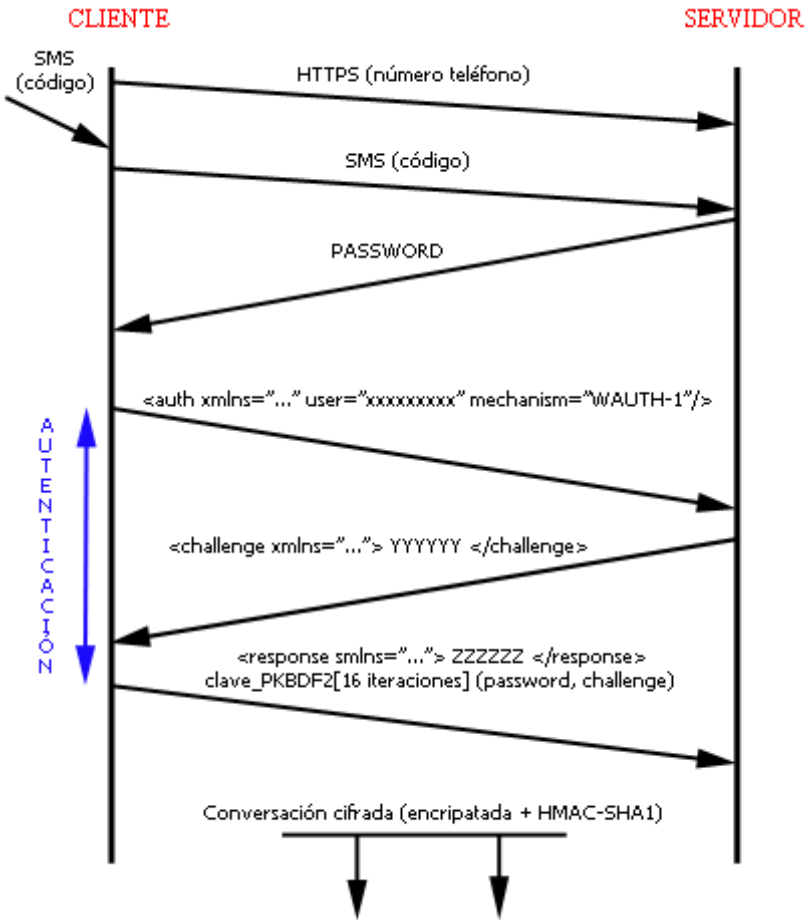

Figura 2.2: Flujo de datos del registro y autenticación entre el cliente y el servidor de WhatsApp.

2.1.5 Agujeros de WhatsApp por resolver

La seguridad es muy importante hoy en día y más si estamos hablando de una aplicación que utiliza la mayoría de la población. WhatsApp almacena gran cantidad de información de forma insegura. Datos como los logs de todos los mensajes enviados y recibidos, la información de los contactos y hasta las coordenadas de geolocalización (si el GPS está activado) se guardan en la tarjeta Micro SD del dispositivo del usuario. Estas copias de seguridad (backup), que se realizan cada 24 horas, permiten recuperar conversaciones borradas. Supuestamente están cifradas con el algoritmo AES (Advanced Encryption

Standard) mediante una clave de 192 bits pero esta clave está insertada en el propio paquete de software de WhatsApp y, además, todos los móviles Android utilizan la misma clave.

La clave de cifrado es la siguiente:

346a23652a46392b4d73257c67317e352e3372482177652c

Evidentemente, no es una contraseña fácil de adivinar, pero buscando por Internet se obtiene fácilmente y mediante ciertas aplicaciones es posible recuperar los mensajes si se dispone de la base de datos cifrada.

Además está el problema de la privacidad antes mencionado que compromete tanto la foto del usuario como su hora de última conexión. Es cierto que WhatsApp incluyó la posibilidad de ocultar la hora de la última conexión pero poca gente conoce esta función.

2.2 SNMP (Simple Network Management Protocol)

El Protocolo utilizado en Gestión de Red, *SNMP*, es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Entre los dispositivos que normalmente soportan SNMP se incluyen routers, switches, servidores o impresoras. Este protocolo permite a los administradores supervisar el funcionamiento de la red, buscar y resolver sus problemas, y planear su crecimiento.

Las versiones de SNMP más utilizadas son SNMP versión 1 (SNMPv1) y SNMP versión 2 (SNMPv2). En su última versión, SNMPv3, posee cambios significativos con relación a sus predecesores, sobre todo en aspectos de seguridad; sin embargo no ha sido mayoritariamente aceptado en la industria.

Hay dos elementos principales en esta arquitectura: el *gestor* y el *agente*. En cada sistema gestionado se ejecuta, en todo momento, un componente de software llamado agente donde la información a manipular reside. El agente es el que reporta la información a través de SNMP al gestor (como respuesta a una petición *get*).

Los agentes exponen los datos de gestión en los sistemas administrados como variables, pudiendo el gestor realizar tareas de gestión, tales como la modificación y la aplicación de una nueva configuración a través de la modificación remota de estas variables (mediante peticiones *set*). Las variables accesibles a través de SNMP están organizadas en jerarquías.

Estas jerarquías (estructuradas en forma de árbol), y otros metadatos, como el tipo y la descripción de la variable, se describen en las Bases de Información de Gestión (MIB). Las MIBs son un tipo de bases de datos donde la información a gestionar se indexa por medio de los OIDs (Object Identifiers) siguiendo el estándar SMIv2. Hay dos tipos de MIBs: las estándares y las privadas.

La principal MIB estándar es la MIB-II, que es la base de datos común para la gestión de equipos en internet, y es de implementación obligatoria. Por otra parte, las empresas y desarrolladores pueden definir ramas privadas que incluyen los objetos administrados para sus propios productos constituyendo las MIB privadas.

Tras explicar de forma general el funcionamiento de la arquitectura SNMP, en la *Figura 2.3* podemos ver un esquema simplificado.

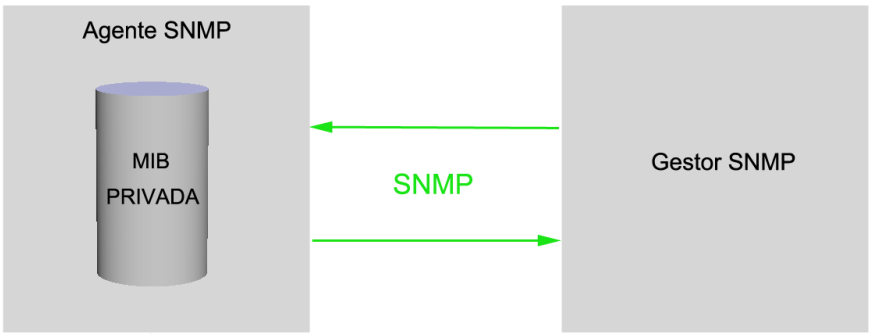

Figura 2.3: Arquitectura general SNMP.

Una explicación más detallada de la arquitectura SNMP se encuentra en el **anexo B**.

2.3 Cacti

Es la herramienta que se va a utilizar para que de forma gráfica se vea la actividad de los usuarios en WhatsApp. Cacti es una herramienta para la generación de gráficos en red, diseñada para aprovechar el poder de almacenamiento y la funcionalidad para gráficas que poseen las aplicaciones RRDtool. Está desarrollada en PHP y provee un pooler ágil, plantillas de gráficos avanzadas, múltiples métodos para la recopilación de datos, y manejo de usuarios. Además tiene una interfaz de usuario fácil de usar a través del navegador web.

Esta herramienta se comunica y obtiene los datos de los dispositivos a monitorizar a través del protocolo de gestión SNMP y permite realizar gráficas de muchos parámetros del dispositivo que se gestiona como, por ejemplo, la carga media del sistema.

En la *Figura 2.4* se muestra un ejemplo de las gráficas que se pueden realizar.

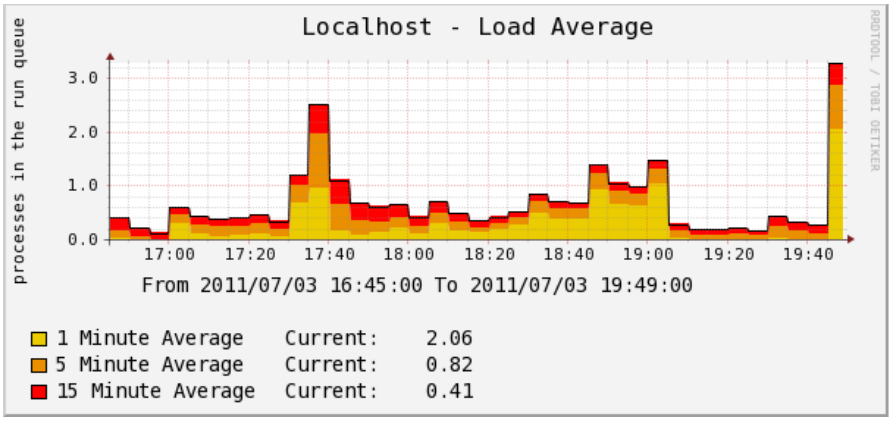

Figura 2.4: Carga media de un sistema.

Capítulo 3

Arquitectura del sistema

El bloque principal del que se compone este TFG es el *agente SNMP*. Para su configuración y monitorización se utilizará un *gestor SNMP*. En este capítulo se va a explicar cómo está compuesto cada bloque, las funciones que realiza y cómo se desarrolló el agente.

En la *Figura 3.1* se representa la arquitectura del sistema, mostrando la interacción entre los distintos bloques que forman. En los siguientes apartados se entrará en detalle en cada uno de ellos.

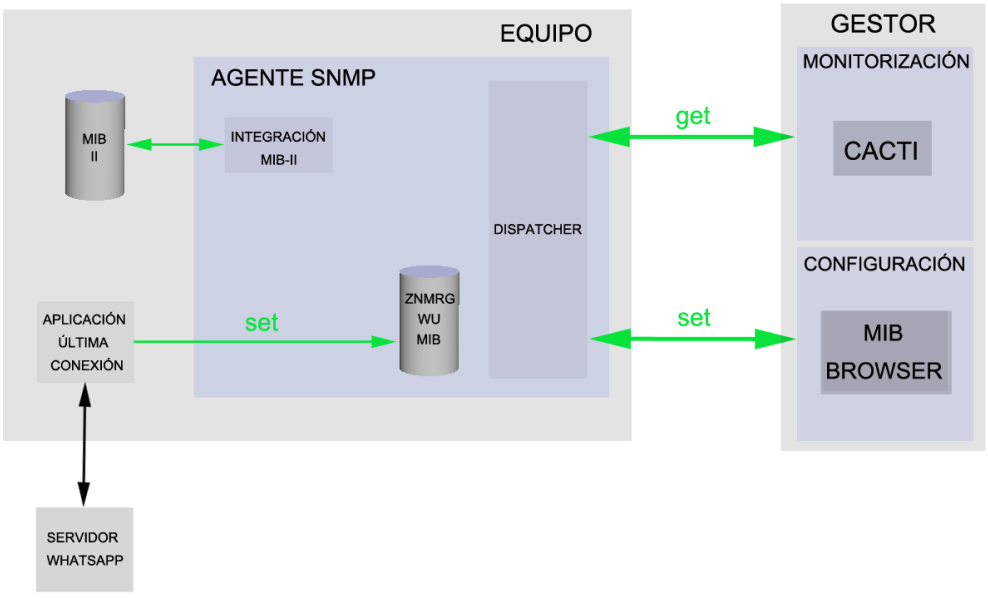

Figura 3.1: Arquitectura de la plataforma.

3.1 El Agente SNMP

El agente SNMP consiste en un componente de software (en este caso un script escrito en el lenguaje de programación Python) que se ejecuta en el equipo que va a ser gestionado. La interacción con el gestor, la aplicación y con Cacti se realiza por medio de SNMP.

El agente está compuesto a su vez por distintos bloques:

- El Dispatcher.
- La MIB privada creada para la aplicación.
- Integración de la MIB-II.
- Aplicación para la obtención de la hora de la última conexión.

3.1.1 Dispatcher

El dispatcher se trata de un socket que gestiona tanto las peticiones entrantes como las salientes. En el script que representa al agente SNMP viene definido como la variable *transportDispatcher*. Este socket se encuentra continuamente escuchando en la dirección privada del equipo donde se aloja el agente en el puerto 161, que es el utilizado por el protocolo SNMP, a la espera de nuevas peticiones o respuestas.

Las peticiones recibidas se pasan al agente para que éste, según el tipo de paquete que sea (para pedir o manipular la información), actúe en consecuencia. Por otra parte, las respuestas se encapsulan como paquetes *GetResponse* para los tres tipos de peticiones (get, getnext o set) y se envían al gestor.

3.1.2 La MIB privada: ZNMRG-WU-MIB

Antes de empezar a desarrollar el agente hay que tener claro como va a ser su funcionamiento y la MIB que va a implementar. Por lo tanto, antes de empezar con la programación del agente hay que diseñar la MIB, en este caso la MIB

privada *ZNMRG-WU-MIB*. Para el desarrollo de esta MIB se siguieron los pasos aprendidos en la asignatura *Gestión de Red*.

La MIB se construye a partir de un modelo de datos. Este modelo permite obtener la MIB con una sintaxis SMIv2 de manera sencilla. Para ello se utilizó el software *MG-SOFT Visual MIB Builder*.

Al tratarse de una MIB privada, ésta colgará del nodo *entreprises*. Como el grupo de investigación de gestión de red de la Universidad de Zaragoza ya había pedido a la IANA un número para el ZNMRG (Zaragoza Network Management Research Group), el 28308, se aprovechó para colgar de éste nuestra MIB.

En la *figura 3.2* se muestra el árbol de la MIB ZNMRG-WU-MIB.

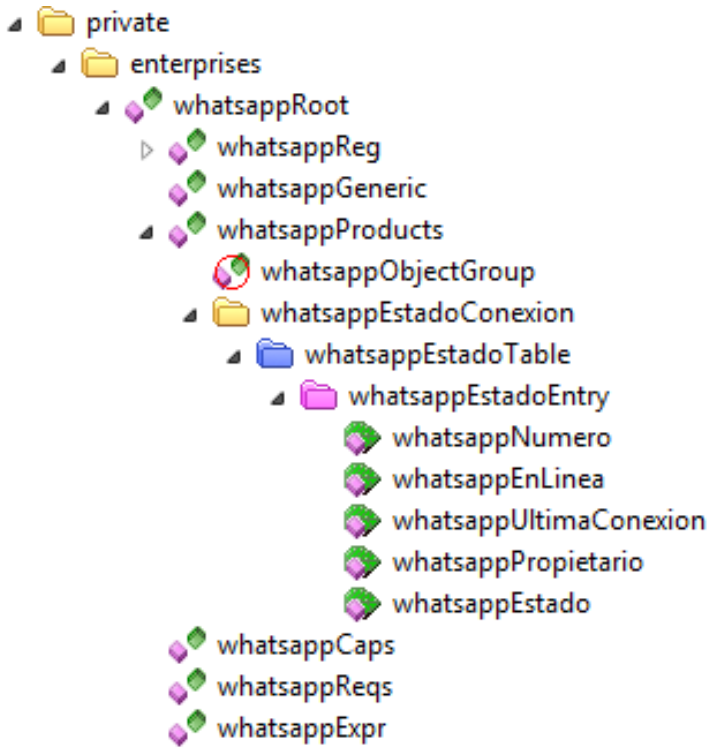

Figura 3.2: *Árbol de la MIB creado con el programa MG-SOFT Visual MIB Builder.*

Para la creación de la MIB hay que agregar elementos al nodo *Products* del grupo ZNMRG, ya que bajo este nodo cuelgan los productos de la empresa. En este trabajo se va a implementar una tabla cuyas columnas corresponden a los datos a gestionar:

- **whatsappNumero:** representa el número de teléfono de la persona a monitorizar. Es un tipo de dato *entero*. Su OID es 1.3.6.1.4.1.28308.3.2.1.1.1.

- **whatsappEnLinea:** en un principio se pensó esta columna para representar con un 1 si estaba en línea y con 0 cuando estuviera desconectado. Posteriormente, viendo las limitaciones que imponía *Cacti* (recoge datos cada 5 minutos), se utilizó esta columna para contabilizar las veces que había estado conectado durante esos 5 minutos. Es un tipo de dato *entero*. Su OID es 1.3.6.1.4.1.28308.3.2.1.1.2.

- **whatsappUltimaConexion:** representa la fecha de la última conexión. Es un tipo de dato *octet string* cuyo formato es dd/mm/yyyy-hh:mm. Su OID es 1.3.6.1.4.1.28308.3.2.1.1.3.

En todas las tablas de las MIBs hay dos columnas que siempre aparecen: el propietario de la fila y una columna estado.

- **whatsappPropietario:** normalmente esta columna es utilizada para contener el nombre del creador de la fila, no obstante, en este caso se le da un uso del nombre de la persona a la que estamos monitorizando. Es un tipo de dato *octet string*. Su OID es 1.3.6.1.4.1.28308.3.2.1.1.4.

- **whatsappEstado:** indica el estado en el que se encuentra la fila. Si es una petición de crear una fila(2), si está en creación(3), si está validada(1) o cuando se borra(4). Es un tipo de dato *entero*. Su OID es 1.3.6.1.4.1.28308.3.2.1.1.5.

Toda tabla tiene un índice que permite su indexación. Éste se elige entre las columnas de la misma. En este caso se eligió indexar por número de teléfono. Por lo tanto, para acceder a un valor concreto de la tabla, se necesita el OID de esa columna y el número de teléfono. Ejemplo: 1.3.6.1.4.1.28308.3.2.1.1.3.657418239 (en este caso se accedería a la fecha de la última conexión, por ser la columna 3, del número indicado).

Una vez definida la tabla, seleccionados todos los tipos de datos y su acceso (sólo escritura, sólo lectura o lectura y escritura), si no hay ningún error, se compila la MIB con el software *MIB Compiler*. El resultado es la MIB **ZNMRG-WU-MIB** que se adjunta en el **anexo C**.

3.1.3 Integración para la MIB-II

En el capítulo anterior se ha comentado que todo dispositivo debe implementar obligatoriamente la MIB-II. En este agente, en vez de implementarla entera, se ha realizado un reenvío de paquetes al servidor SNMP que incorpora linux. De este modo, si alguien realiza una consulta a la MIB-II, el agente preguntará por ella al servidor *snmpd.conf* y, una vez recibida la respuesta, se la reenviará al gestor.

3.1.4 Aplicación para la obtención de la hora de la última conexión

En primer lugar se realizó un estudio de las diferentes plataformas que permitían utilizar WhatsApp en el ordenador para ver si era posible obtener la hora de la conexión a partir de ellas. En este estudio se encontró una librería escrita en Python llamada *Yowsup*.

Yowsup es una librería open source de WhatsApp escrita en Python por Tarek Tgalal que permite emplear el protocolo WhatsApp en cualquier sistema operativo que soporte el intérprete de Python. La librería tiene toda la funcionalidad del cliente WhatsApp que instalamos en nuestros smatphones. La plataforma permite ejecutar *yowsup-cli* que permite, a través de línea de comandos, conectarse y utilizar WhatsApp. A partir de esta aplicación se puede obtener la hora de la última conexión utilizando el comando */lastseen* (este comando en realidad devuelve la fecha de la última conexión).

Como con *yowsup-cli* era posible obtener la hora de la última conexión, se utilizó esta aplicación para nuestro objetivo. No obstante, como no eran necesarias todas las funciones que la aplicación ofrecía (chatear con el número solicitado, cambiar nuestro estado, etc.), se desarrolló un script en Python con las funciones de

Yowsup que permitían obtener la hora de la última conexión. Este script se ejecuta en el mismo equipo donde se aloja el agente pero en un terminal independiente. De este modo habrá un nuevo terminal por cada número que se monitoriza realizando un seguimiento individual a cada uno de ellos. Su funcionamiento es el siguiente: obtiene cada minuto el estado de conexión y, si la persona se encuentra en línea, actualiza el campo *whatsappUltimaConexion* de la tabla del agente. Además, el script hace un sondeo cada 5 minutos de las veces que ha accedido a la aplicación de WhatsApp el usuario y, pasado este perÃodo, actualiza la columna *whatsappEnLinea* con ese valor. Esto se debe, como se ha comentado anteriormente, a las limitaciones impuestas por Cacti.

La instalación de la plataforma Yowsup (necesaria para el funcionamiento de la aplicación desarrollada) se explica en el **anexo D**.

En la *Figura 3.3* se muestra la máquina de estados describiendo el funcionamiento de la aplicación.

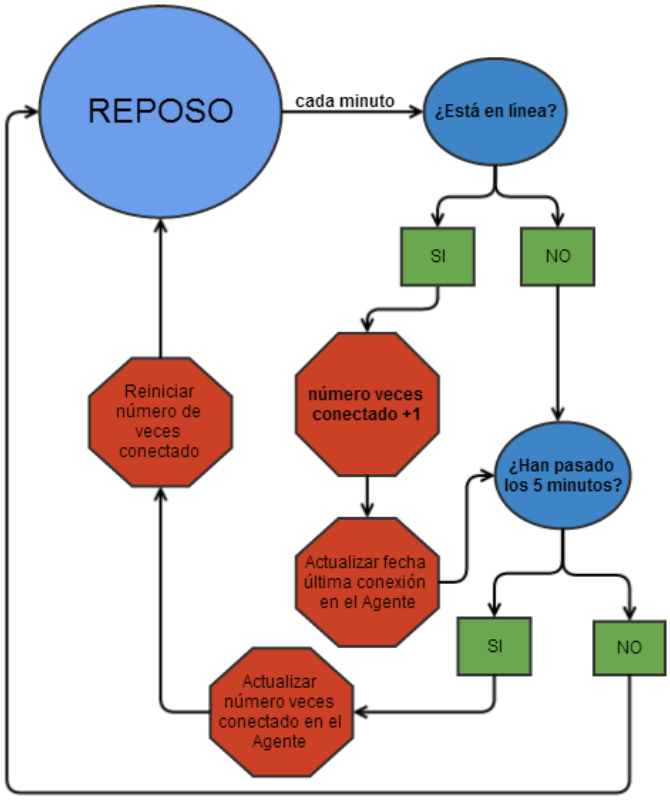

Figura 3.3: Máquina de estados de la aplicación para obtener la hora de la última conexión.

La función principal de la aplicación es *onPresenceUpdated(jid, lastSeen)*. En ella se realizan las comprobaciones del estado del número de teléfono.

- Si el número se encuentra conectado vemos en la línea 7 el comando que actualiza al Agente SNMP con la fecha actual. Ese comando es para mandar un *SetResponse* a la columna *whatsappUltimaConexion*.

```
1 def onPresenceUpdated(jid, lastSeen):
2         if((lastSeen//60) == 0):
3                 print "\nEn linea. Son las "+datetime.datetime.
                      now().strftime("%H")+":"+datetime.datetime.
                      now().strftime("%M")
4                 hora_actual = datetime.datetime.now().strftime("
                      %d/%m/%Y-%H:%M")
5                 oid = "1.3.6.1.4.1.28308.3.2.1.1.3." + numero
6                 print "\nActualizando la hora de ultima conexion
                       en el agente SNMP..."
7                 os.system("snmpset -v2c -c public " + ip + " " +
                       oid + " s " + hora_actual)
8                 veces_enlinea += 1
```

- Mientras que si no se ha conectado simplemente muestra por la pantalla del terminal el tiempo pasado desde la última conexión.

Por otra parte, pasados los 5 minutos, actualiza en el agente las veces que se ha conectado durante este perÃodo. Esta función se puede ver, en la línea 4 del fragmento de código siguiente, donde la aplicación manda un *SetResponse* a la columna *whatsappEnLinea* cuyo valor es el número de conexiones.

```
1       if multiplo(i):
2               oid = "1.3.6.1.4.1.28308.3.2.1.1.2." + numero
3               print "\nActualizando el numero de veces ""en linea""
                     durante los ultimos 5 minutos..."
4               os.system("snmpset -v2c -c public " + ip + " " + oid
                   + " i " + str(veces_enlinea))
5               enlinea = [0,0,0,0,0]
6               veces_enlinea = 0
```

Esta aplicación ha de contactar con los servidores de WhatsApp. Para hacer posible esta conexión hay que realizar un registro inicial en el servicio y, por lo tanto, es necesario utilizar un número de teléfono para poder monitorizar al resto.

El código completo de la aplicación se encuentra en el archivo **ObtenerUltimaConexion.py** en el CD adjunto.

3.1.5 Desarrollo del agente

Una vez desarrollada la MIB privada hay que programar el agente que la implemente. En vez de crear una base de datos, la tabla que contiene los números de teléfono se va a implementar como un diccionario de Python. Los diccionarios en Python son contenedores de pares clave-valor, cuyos índices son cadenas (la clave) en vez de ser números enteros. Esto permite realizar una indexación por OID. Cabe mencionar que los diccionarios no están ordenados, por lo tanto, el script contiene una función para ordenar los OID en orden creciente (ya que SNMP devuelve un error en caso contrario).

El cuerpo del script consiste en una función de tratamiento de paquetes. En primer lugar se extrae el contenido del paquete y se decide qué tipo de petición es:

- **GetRequest (get):** si la petición es un *get*, se mira el OID por el que pregunta. Si pertenece a la MIB privada, prepara un paquete GetResponse con el par OID-valor pedido. Si pertenece a la MIB-II, realiza el reenvío anteriormente mencionado y responde con la respuesta obtenida. Si no pertenece a ninguna de las anteriores, envía un error como respuesta.

- **GetNextRequest (walk):** si la petición es un *walk*, se mira el OID por el que pregunta. Si pertenece a la MIB privada, busca el siguiente OID al recibido, y va devolviendo todos los pares OID-valor que estén por debajo de ese OID hasta que no encuentra más. Si pertenece a la MIB-II, realiza el reenvío anteriormente mencionado y responde con todas las respuestas obtenidas. Si no pertenece a ninguna de las anteriores, envía un error como respuesta.

- **SetRequest (set):** si la petición es un *set*, se mira el OID por el que pregunta. Si pertenece a la MIB privada, se modifica el valor de la tabla indicado por el OID, si el tipo de dato coincide con el esperado en la columna y el rango permitido. En caso de no cumplir alguna de estas dos normas, devuelve el error correspondiente. Al gestor se le envía también un paquete con el par OID-valor modificado para que tenga constancia de que el cambio se ha producido. Si pertenece a la MIB-II, devuelve un error de escritura porque son valores de sólo lectura. Si no pertenece a ninguna de las anteriores, envía un error como respuesta.

El código completo del agente se encuentra en el archivo **AgenteSNMP.py** en el CD adjunto.

3.2 Gestor SNMP

El gestor es el encargado de controlar al agente. Éste se encarga de crear las filas, que representan la monitorización de los distintos números, activarlas o borrarlas y de mostrar gráficamente los resultados. El gestor se puede divirdir también en dos bloques: *el gestor de configuración* y *el gestor de monitorización.*

3.2.1 El gestor de configuración: MIB Browser

Este bloque consiste en el software *MIB Browser* que permite gestionar remotamente un dispositivo. Con este programa, trabajando a nivel de usuario, se puede interaccionar con los módulos desarrollados pudiendo tanto manipular la tabla como ver su contenido. No obstante, se podría haber utilizado otro software comercial.

Para poder gestionarlo es necesario primero cargar la MIB creada en el MIB Browser y posteriormente conectarse a la dirección IP del equipo donde se aloja el agente. Una vez hecho esto ya es posible acceder a él.

Los pasos a seguir para crear una nueva fila para monitorizar un nuevo número son los siguientes:

1. Se manda una solicitud para crear una nueva fila. Para ello hay que mardar un *SetRequest* con el OID, **1.3.6.1.4.1.28308.3.2.1.1.5. + número**, que corresponde a la columna *whatsappEstado* con valor 2. En el momento en que se recibe en el agente, éste crea la fila asignando: el número utilizado para crear la fila a la columna *whatsappNumero*, un 0 a la columna *whatsappEnLinea*, las columnas *whatsappUltimaConexion* y *whatsappPropietario* las inicializa vacías y la columna *whatsappEstado* pasa a tomar el valor 3 (que quiere decir que la columna está en creación).

2. Se modifica el propietario de la nueva fila para saber el nombre de la persona a monitorizar. Para ello hay que mandar un *SetRequest* con el OID, **1.3.6.1.4.1.28308.3.2.1.1.4. + número**, que corresponde a la columna *whatsappPropietario*.

3. Por último se valida la fila. Para ello hay que mardar un *SetRequest* con el OID, **1.3.6.1.4.1.28308.3.2.1.1.5. + número**, a la columna *whatsappEstado* con valor 1.

En la *Figura 3.4* se muestra la tabla monitorizando a 3 números de teléfono.

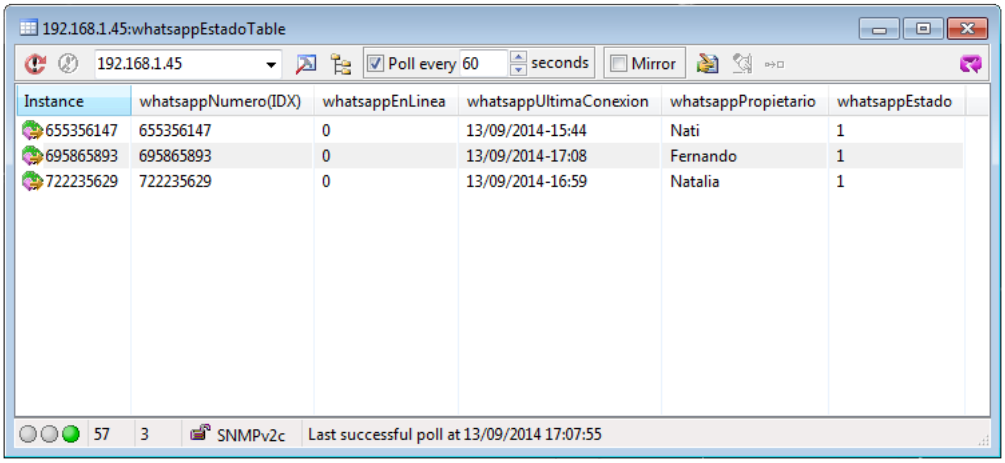

Figura 3.4: Tabla de monitorización de la MIB ZNMRG-WU-MIB.

3.2.2 El gestor de monitorización: Cacti

Este bloque consiste en la herramienta de generación de gráficos en red Cacti. Esta herramienta, que trabaja a nivel de usuario con una configuración personalizada para este proyecto, se instala sobre un servidor web (puerto 80), se accede a ella desde http://localhost/cacti y recolecta los datos mediante SNMP de dos formas distintas: utiliza un archivo XML que le indica como recoger los datos o recoge los datos lanzando un script.

En este caso se utiliza un archivo XML. En el siguiente fragmento de código se puede ver como éste le indica a la herramienta la columna de la que ha de recoger la información (identificada como output), las funciones para obtener los datos debe utilizar (get o walk), si los datos son valores para realizar la gráfica (output) o datos estáticos que pueden ser utilizados en la leyenda como comentarios (input) o a qué OIDs deben realizarse estas funciones.

```
1  <whatsappPropietario>
2    <name>Propietario</name>
3    <method>walk</method>
4    <source>value</source>
5    <direction>input</direction>
6    <oid>.1.3.6.1.4.1.28308.3.2.1.1.4</oid>
7  </whatsappPropietario>
8  <whatsappEnLinea>
9    <name>EnLinea</name>
10   <method>walk</method>
11   <source>value</source>
12   <direction>output</direction>
13   <oid>.1.3.6.1.4.1.28308.3.2.1.1.2</oid>
14 </whatsappEnLinea>
```

El código completo de este archivo XML se encuentra tanto en el **anexo E** de este trabajo como en el CD adjunto.

La configuración de Cacti realizada para este trabajo se explica en el **anexo F**.

Capítulo 4

Aplicación en escenario real: Resultados

Para poder realizar una representación gráfica de los resultados se deben seguir una serie de pasos previos que se exponen a continuación:

1. Instalar la plataforma Yowsup en Debian (pudiéndose realizar de forma manual, como se explica en el **anexo D**, o ultilizando el script **InstalarPlataforma.py** que se desarrolló para automatizar el proceso).

2. Copiar el archivo **ObtenerUltimaConexion.py** en la ruta */yowsup/src*. Esta aplicación se ha desarrollado a partir de funciones de Yowsup, por lo tanto, para que funcione se ha de incluir en la ruta creada en la instalación anterior.

3. Editar las variables *nickname*, *username* y *password* del archivo anterior. Estas variables se utilizan en la autenticación con el servidor de WhatsApp. Los campos se han de rellenar de la siguiente manera:

    - El *nickname* se puede rellenar con cualquier nombre.
    - El *username* es el número de teléfono completo (Ej. 34695852123). Como se ha comentado en el capítulo anterior, es necesario un número para poder monitorizar al resto (ya que para poder contactar con el

servidor de WhatsApp, y obtener la hora de última conexión, se ha de realizar una autenticación previa).

- El *password* es el recibido desde el servidor de WhatsApp. Si se ha utilizado el script para instalar la plataforma se puede encontrar en el archivo *whatsapp_config.txt* que se crea tras la instalación de la plataforma.

4. Lanzar el agente. Para ello hay que ejecutar el script que lo ejecuta (en Debian el comando **python AgenteSNMP.py**).

5. Crear con el software *MIB Browser* las filas a monitorizar y validarlas (como se ha comentado en el capítulo 3).

6. Configurar *Cacti* para esta aplicación (como se muestra en el **anexo F**).

El resultado de monitorizar, durante un día, los siguientes numeros de teléfono son las siguientes gráficas:

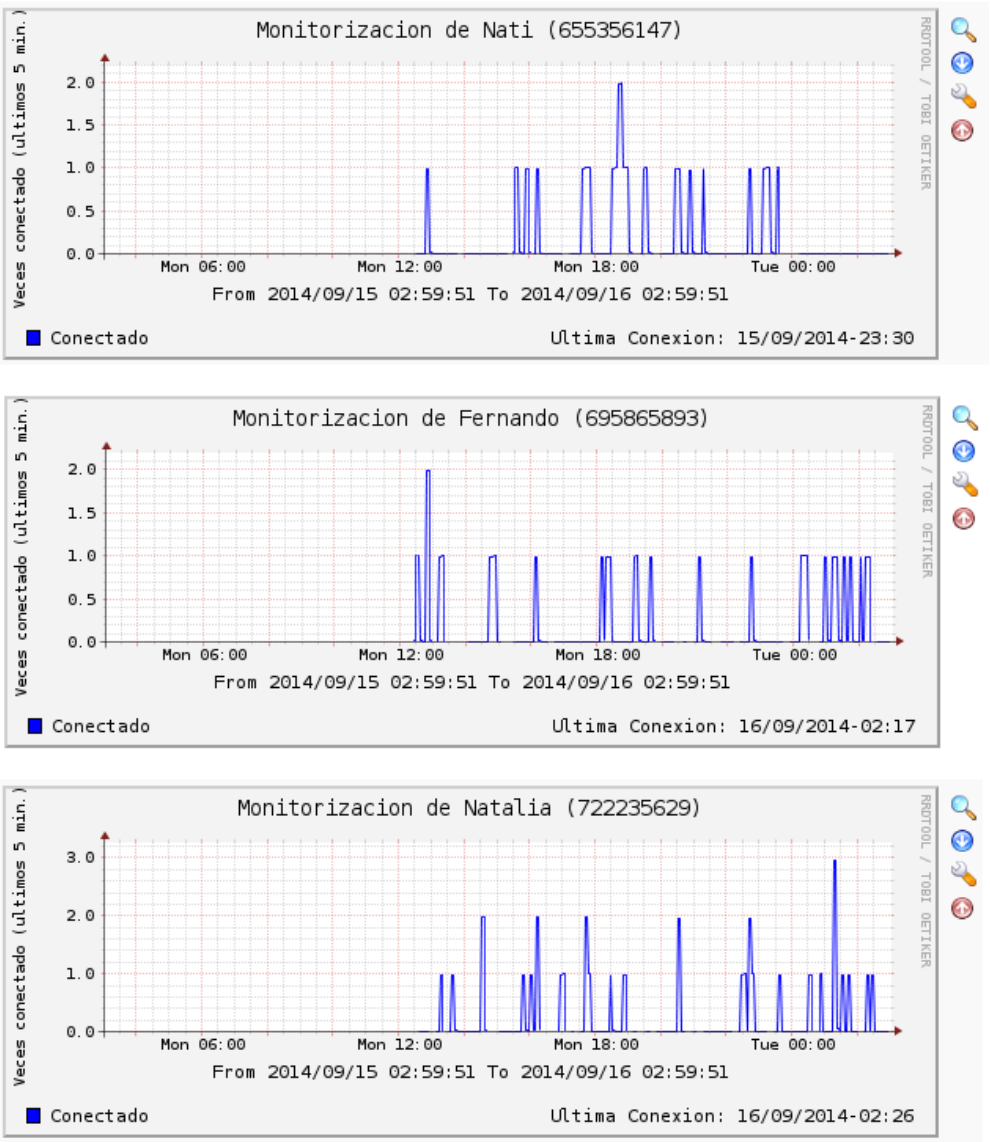

Figura 4.1: Monitorización de 3 números de teléfono durante un día.

Capítulo 5

Conclusiones y líneas futuras

5.1 Opinión personal del trabajo

La realización de este proyecto conlleva la finalización del Grado y supone un reto individual en el que poner en práctica algunos de los conocimientos aprendidos durante estos 4 años. A pesar de basarse en algunas asignaturas (como Gestión de Redes o fundamentos de programación), la mayor parte del proyecto ha sido autodidacta ya que he tenido que aprender el lenguaje de programación Python y expandir los conocimientos aprendidos en las asignaturas antes mencionadas. Además ver cómo al final acabas pudiendo hacer cosas que cuando te planteaban el proyecto te parecían complicadísimas supone una gran satisfacción.

Elegí este proyecto ya que me permitía aprender a programar en Python (lenguaje en auge con mucho potencial) que complementa con el resto de lenguajes aprendidos durante la carrera y porque me interesaba aprender como funcionaba la aplicación de moda, WhatsApp.

En conclusión, este trabajo ha sido una gran experiencia en el que no solo se aprenden lenguajes o conocimientos, sino a resolver los problemas que van surgiendo durante la realización del mismo.

5.2 Conclusiones

Este Trabajo de Fin de Grado buscaba mostrar que era posible obtener la hora de la última conexión de los números de WhatsApp y posteriormente realizar con Cacti las gráficas de los resultados mostrando así algunas de las vulnerabilidades que presenta la aplicación.

Se ha conseguido una plataforma que permite la monitorización, con un programa externo, de varios números simultáneamente. La plataforma está completamente integrada en la arquitectura SNMP que es el estándar de facto para la gestión de redes. Se ha utilizado Cacti como gestor externo pero cualquier gestor externo podría tener acceso a la información obtenida.

5.3 Líneas futuras

En un futuro, la plataforma se podría expandir para que obtuviera también la foto de usuario del número elegido (siempre y cuando WhatsApp no solucione los problemas vigentes). Además se podría realizar una aplicación móvil que mostrara las gráficas obtenidas en cualquier lugar ya que sería interesante controlar la actividad del número deseado en tiempo real.