Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de la Telecomunicación

Uso de Kerberos en un entorno WSO2

Autor: David Barranco Alfonso
Tutor: Isabel Román Martínez

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de la Telecomunicación

Uso de Kerberos en un entorno WSO2

Autor:
David Barranco Alfonso

Tutor:
Isabel Román Martínez
Profesor colaborador

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo de Fin de Grado: Uso de Kerberos en un entorno WSO2

Autor: David Barranco Alfonso

Tutor: Isabel Román Martínez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:
Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

Agradecimientos

Quisiera empezar este espacio agradeciendo a mi tutora, Isabel, por cada momento que me ha dedicado durante estos años para el desarrollo de este proyecto. Por toda su ayuda y ánimo.

Por supuesto agradecer a mi familia, principalmente a mis padres su confianza en mí y su apoyo incondicional. Gracias por toda esa fuerza para realizar este proyecto.

Por último, pero no por ello menos importante, a todos los compañeros que he conocido durante estos años en la escuela de ingenieros y a los amigos que me han acompañado durante toda mi vida, sin los cuales no habría llegado hasta aquí.

Resumen

En el Máster en ciber seguridad de la Universidad de Sevilla se imparten dos sesiones basadas en la autenticación y el control de acceso, cuestiones estrechamente relacionadas. En las mismas existen ejercicios prácticos que, actualmente, están desconectados.

En este trabajo de fin de grado se da el primer paso para integrar ambas prácticas, adaptando la práctica de autenticación al servidor Kerberos embebido en el servidor de identidad (IS) de WSO2. Se configura también el bus de servicios de empresa (ESB) de WSO2 y se despliega un servicio de eco que requiere un token Kerberos de identidad. Para terminar se desarrolla un cliente que utiliza dicho servicio, previa autenticación en el IS.

La intención es continuar con la adaptación de la práctica de control de acceso para utilizar los mismos recursos en la misma.

Abstract

The University of Seville offers a Master degree focused on Cibersecurity in data networks. In this Master degree there are two lessons dedicated to user authentication and user authorization, concepts that are strictly related. In those lessons there are some practical sessions which, currently, are disconnected one from the other.

This thesis takes the first step to integrate both practical sessions, adapting the first session that is focused on user authentication to use the Kerberos server that is embedded in the WSO2 Identity Server (IS). Furthermore, the WSO2 Enterprise Service Bus (ESB) is configured to deploy an echo service that requires a Kerberos token to authenticate a user. Along with this, it has been developed a client tool that uses the echo service, authenticating each user that access to the server with the IS Kerberos server.

The intention is to continue adapting the practical session that is focused on user authorization to use the same resources that have been used in this thesis.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xiii**

Índice **xiv**

Índice de Tablas **xvii**

Índice de Figuras **xix**

1 Introducción y objetivo **10**
1.1. Objetivo *10*

2 Organización del documento **12**

3 Estado del arte y tecnologías empleadas **14**
3.1. Plataforma WSO2 *14*
3.2. Middleware *19*
3.2.1. Definición 19
3.2.2. Carbon 19
3.3. Arquitectura Orientada a Servicios (Service – Oriented Architecture) *20*
3.3.1. Diseño y desarrollo de SOA 20
3.3.2. Tipos de Servicios 20
3.4. WSO2 Servidor de Identidad – Identity Server *23*
3.5. WSO2 Bus de Servicios de Empresa – Enterprise Service Bus *24*
3.6. Protocolo Kerberos *25*
3.6.1. Definición 25
3.6.2. Motivación 25
3.6.3. Conceptos importantes 25
3.6.4. Ventajas de Kerberos 26
3.6.5. Desventajas de Kerberos 26
3.6.6. Paso de mensajes 27

4 Desarrollo del proyecto **32**
4.1. Escenario *32*
4.2. Virtualbox *34*
4.2.1. Información sobre máquinas virtuales 36
4.3. Configuración de direcciones de red de las máquinas virtuales *37*
4.4. Máquinas virtuales de WSO2 *38*
4.4.1 Requisitos de *software* a instalar 38
4.5. Máquina virtual IS *44*
4.5.1. Configuración del IS 46
4.6. Máquina virtual ESB *54*
4.6.1. Configuración del ESB y servicio de eco 57
4.7. Máquina virtual del cliente *64*
4.7.1. Cliente del servicio eco 65
4.7.1.1. Archivo de configuración de cliente – build.xml 66

4.7.1.2. Archivo de configuración de cliente – krb5.conf 66
4.7.1.3. Archivo de configuración de cliente – Jaas.conf 67
4.7.1.4. Archivo de configuración de cliente – policy.xml 68
4.7.1.5. Código Java del cliente 69
4.7.2. Ejecución *70*
4.8. Análisis trazas de ejecución del cliente *72*

5 Conclusiones y trabajo futuro **81**
5.1. Trabajo futuro *81*

6 Bibliografía **83**

7 Glosario **86**

Anexo A: Código del cliente del servicio eco **88**

ÍNDICE DE TABLAS

Tabla 1 – Información sobre máquinas virtuales 36
Tabla 2 – Características de *Carbon* de WSO2 IS 45
Tabla 3 - Características de *Carbon* de WSO2 ESB 57

ÍNDICE DE FIGURAS

Figura 3-1 Esquema de productos de WSO2 14
Figura 3-2 Esquema funcional del Gestor de Interfaces Públicas 15
Figura 3-3 Esquema funcional del servidor de Integración 16
Figura 3-4 Esquema funcional del Servidor de Identidad 17
Figura 3-5 Esquema de características del Servidor del Internet de Las Cosas 18
Figura 3-6 Esquema de servicios instalables sobre *Carbon* 19
Figura 3-7 Esquema de tipo de servicios SOA 21
Figura 3-8 Ejemplo de servicio *SOA* de utilidad 21
Figura 3-9 Ejemplo de servicio *SOA* de entidad 22
Figura 3-10 Ejemplo de servicio SOA de tarea 22
Figura 3-11 Componentes incluidos en el Servidor de Identidad 23
Figura 3-12 Diagrama de paso de mensajes de Kerberos 27
Figura 4-1 Diagrama de paso de mensajes de Kerberos con productos WSO2 33
Figura 4-2 Creacion de red interna para máquinas virtuales en VirtualBox (I) 34
Figura 4-3 Creación de red interna para máquinas virtuales en VirtualBox (II) 35
Figura 4-4 Importación de máquinas virtuales en VirtualBox 35
Figura 4-5 Ajuste de red interna para máquinas virtuales en VirtualBox 36
Figura 4-6 Menu de reenvío de puertos 37
Figura 4-7 Lista de reenvío de puertos para máquinas virtuales 37
Figura 4-8 Archivo de sistema /etc/hosts 38
Figura 4-9 Descargas del *jdk 6u45* 39
Figura 4-10 Instalación de *jdk 6u45* 39
Figura 4-11 Descarga de *Carbon* 40
Figura 4-12 Descompresión de *Carbon* en ordenador local 40
Figura 4-13 Inicio de *Carbon* 41
Figura 4-14 Inicio de sesión en *Carbon* 42
Figura 4-15 Menú de características de *Carbon* 42
Figura 4-16 Asistente de configuración de repositorios de características de *Carbon* 43
Figura 4-17 Instalación de caraterísticas de *Carbon* 43
Figura 4-18 Menú de gestión de *Carbon* 46
Figura 4-19 Interfaz de inicio de WSO2 IS 46
Figura 4-20 Activación del KDC en el IS 47
Figura 4-21 Activación del KDC en el IS (2) 48
Figura 4-22 Activación del KDC en el IS (3) 49
Figura 4-23 Inicio del servicio de Kerberos en el IS 49

Figura 4-24 Menú lateral de configuración de Kerberos 50
Figura 4-25 Configuración de ID de servicio de eco 51
Figura 4-26 Creación de ID de servicio de eco 52
Figura 4-27 Menú de configuración de usuarios y roles 52
Figura 4-28 Configuración de usuario y contraseña para el servicio de eco 53
Figura 4-29 Creación de usuario para el servicio de eco 53
Figura 4-30 Interfaz de inicio de WSO2 ESB 57
Figura 4-31 Mensajes de inicio del ESB con dirección de red de administración 59
Figura 4-32 Inicio de sesión en la consola de administración web del servicio 59
Figura 4-33 Interfaz de inicio del ESB 60
Figura 4-34 Menú lateral de configuración de servicios web 61
Figura 4-35 Listado de servicios web incluidos en el ESB 61
Figura 4-36 Consola de administración del servicio web de eco 62
Figura 4-37 Métodos de securización del servicio de eco 63
Figura 4-38 Configuración del ID del servicio de eco 64
Figura 4-39 – Contenido del archivo de configuración de Ant 66
Figura 4-40 – Configuración de políticas de seguridad para el cliente del servicio eco 68
Figura 4-41- Configuración de Kerberos para el cliente del servicio de eco 68
Figura 4-42 – Configuración del contexto del cliente del servicio de eco 69
Figura 4-43 Bucle principal de código del cliente del servicio de eco 70
Figura 4-44 Traza de ejecución del cliente 71
Figura 4-45 Métricas del tiempo de respuesta del servicio de eco 72
Figura 4-46 – Acceso a la configuración de trazas de sistema del IS 73
Figura 4-47 – Cambio de configuración de trazas de sistema en el IS 73
Figura 4-48 – Acceso a la sección de monitorización del IS 74
Figura 4-49 – Búsqueda de usuario en base de datos LDAP 75
Figura 4-50 – Petición de ticket de servicio para el servicio de eco 75
Figura 4-51 – Respuesta de creación de TGS para el servicio de eco 75
Figura 4-52 – Inicio de sesión del cliente contra el servicio de eco 76
Figura 4-53 – Creación de sesión encriptada con el protocolo Kerberos entre cliente y servicio de eco 76
Figura 4-54 – Comprobación de marcas de tiempo de los tickets de Kerberos 77
Figura 4-55 – Creación marcas de tiempo para respuesta del servicio de eco al cliente 77
Figura 4-56 – Añadido de marcas de tiempo a trafico de servicio de eco a cliente 78
Figura 4-57 – Envío de codigo de estado OK del servicio de eco al cliente 79
Figura 4-58 – Diagrama de paso de mensajes entre entidades del caso práctico 79

1 INTRODUCCIÓN Y OBJETIVO

El Máster en ciberseguridad ofertado por la Universidad de Sevilla es una alternativa muy interesante para alumnos que terminan el grado universitario. En él se aporta una visión muy extensa sobre todos los aspectos implicados en la ciber seguridad hoy en día.

Actualmente en el módulo dedicado a la securización de servicios y sistemas de información se imparten dos sesiones sobre autenticación y control de acceso muy relacionadas entre sí, pero que presentan un problema de desconexión en los ejercicios de ambas sesiones prácticas.

En un esfuerzo por parte del profesorado por mejorar estas sesiones prácticas surge este proyecto, el cual realiza una propuesta de modificación de la sesión de autenticación, para desarrollarla utilizando los componentes y servicios proporcionados por WSO2:

- **Gestión de autenticación**. En esta sesión práctica se realizan diferentes ejercicios sobre un escenario compuesto por 3 equipos en el que se despliega un servicio y se securiza bajo el protocolo de Kerberos. También se utiliza el servicio *LDAP* como repositorio de los diferentes usuarios y respectivos roles.

1.1. Objetivo

El objetivo de este proyecto es el de realizar una propuesta de mejora de esta sesión práctica haciendo uso de las tecnologías que se utilizan en las siguientes sesiones prácticas, creando así un marco común sobre el que desarrollar esta serie de prácticas del Máster de ciberseguridad.

Dado que estos ejercicios están basados en el uso del protocolo Kerberos y de la plataforma WSO2 se propone realizar un caso práctico sobre la plataforma WSO2 utilizando el Servidor de Identidad y el Bus de Servicios de Empresa para securizar un servicio haciendo uso del protocolo Kerberos. De este modo, los diferentes equipos que participan en este caso práctico se autentican entre ellos gracias a Kerberos y se le otorga acceso al servicio publicado en la plataforma de WSO2.

Gracias a WSO2, la realización de la práctica es sencilla y rápida, de manera que es posible el planteamiento de ejercicios sobre el escenario desarrollado orientados a profundizar los conocimientos enseñados en las clases teóricas.

2 ORGANIZACIÓN DEL DOCUMENTO

Antes de comenzar con el desarrollo de la solución práctica, es conveniente estudiar por qué se ha elegido WSO2, el estado actual de la plataforma, de dónde viene y hacia dónde va. Por ello, en el siguiente capítulo se desarrolla el estado del arte de WSO2 y el paradigma bajo el que se creó esta plataforma.

Una vez finalizada esta visión global sobre WSO2 y sus productos el siguiente capítulo se centra en introducir las tecnologías empleadas en este proyecto. Abarcando desde un punto de vista más general hasta uno más específico para entender correctamente el sistema sobre el que se trabajará.

Llegados a este punto, se dedica el siguiente capítulo al desarrollo del proyecto. Este capítulo muestra un escenario clásico sobre el que se despliega Kerberos y cómo este escenario cambia cuando se introducen los componentes de la plataforma WSO2. Después se continúa con la instalación e implementación del servicio, para terminar hablando del cliente que accede al servicio y sus pruebas de ejecución.

Se finaliza la memoria de este proyecto con el análisis del caso práctico en base a la información recibida en el IS y el ESB, las conclusiones principales extraidas y las posibles líneas futuras de trabajo.

3 ESTADO DEL ARTE Y TECNOLOGÍAS EMPLEADAS

Este capítulo está dedicado a presentar el estado del arte de la plataforma WSO2 y todas las tecnologías que se han utilizado a la hora de desarrollar este proyecto.

3.1. Plataforma WSO2

WSO2 [1, 2] es una compañía tecnológica que desarrolla y reutiliza software libre. Fue fundada en 2005 por Sanjiva Weerawarana y actualmente se dedicada al desarrollo de aplicaciones middleware, las cuales proveen de una arquitectura orientada a servicios. Es una compañía de renombre, con clientes como Ebay o Boeing.

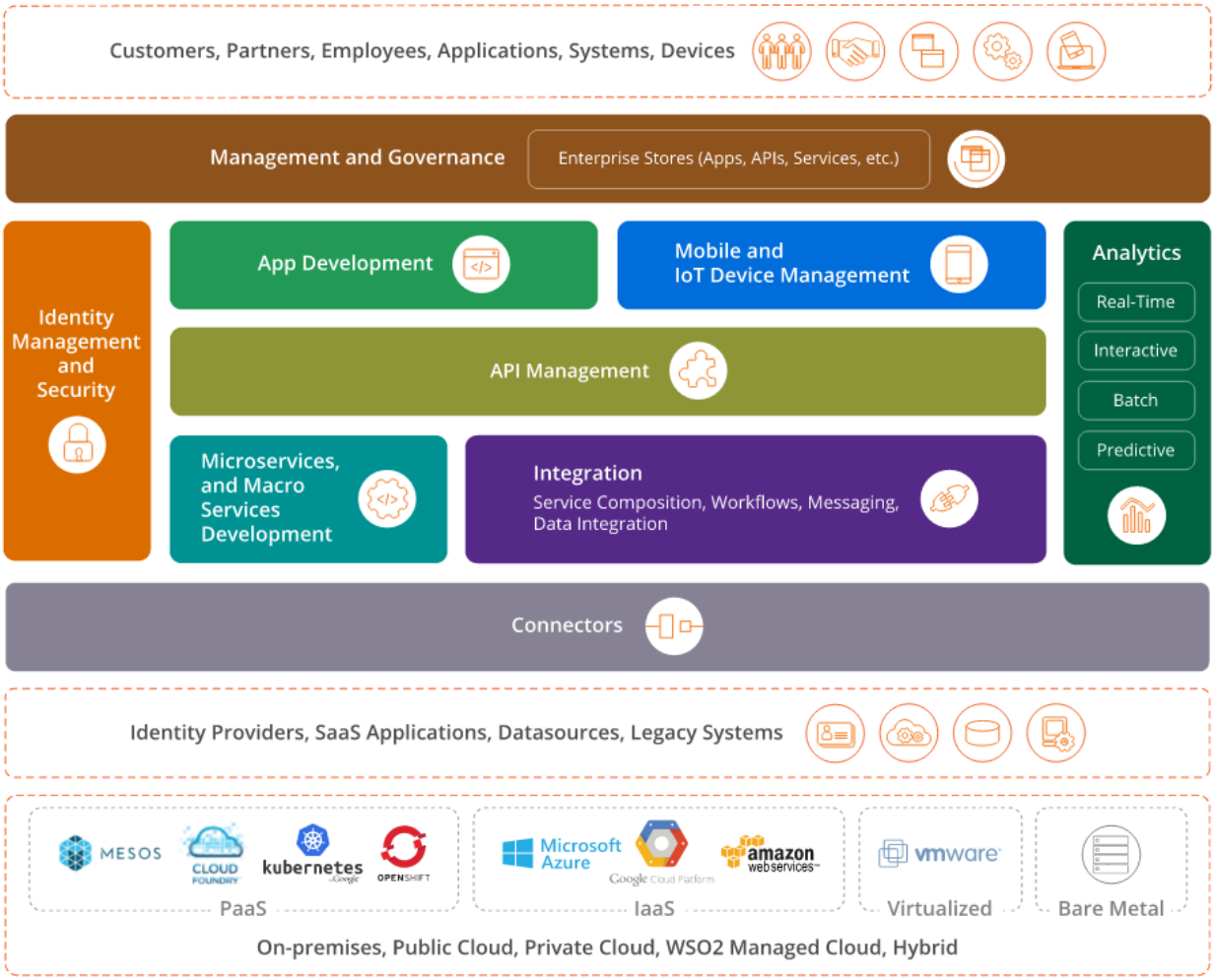


Figura 3-1 Esquema de productos de WSO2

En esta imagen se observan las diferentes soluciones que componen la plataforma de WSO2, disponiendo desde conectores para sistemas heredados hasta gestores de interfaces públicas (API's) para los servicios que se desplieguen sobre la plataforma de WSO2.

Hoy en día WSO2 ofrece una plataforma completa sobre la que desplegar y gestionar aplicaciones distribuidas de alta disponibilidad, escalables y seguras.

Si se realiza un repaso general a los productos de WSO2 se pueden englobar cinco grandes categorías:

- **Gestión de interfaces públicas (API)[3]**

  Esta solución incluye las herrramientas necesarias para el diseño, creación, mantenimiento y monitorización de las interfaces públicas de los servicios disponibles en la plataforma.

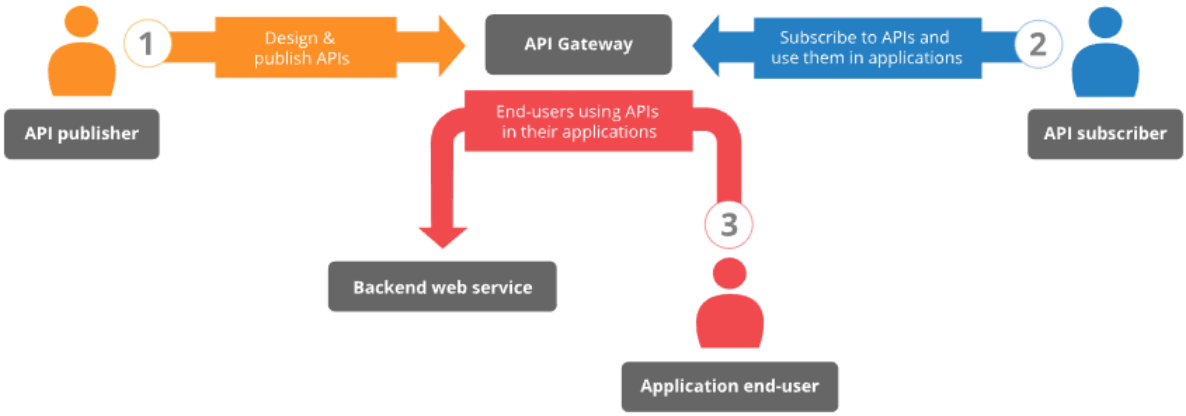


Figura 3-2 Esquema funcional del Gestor de Interfaces Públicas

Las características más importantes de este producto son:

- Permite realizar un acceso a interfaces públicas basado en roles (RBAC – Role Based Access Control) y atributos (ABAC – Atribute Based Access Control).
- Herramientas para la creación de documentación para las interfaces públicas.
- Funciones de calidad de servicio.
- Monitorización del uso de las interfaces públicas y comportamiento general del sistema
- Posibilidad de añadir estándares de autenticación como OAuth, credenciales del cliente o SAML.

Esta solución puede ser muy útil dependiendo de la infraestructura de una empresa.

- **Servidor de integración [4]**
  Este producto se centra en la integración de toda clase de sistemas que puedan existir en el negocio, desde sistemas heredados hasta sistemas de computación en nube. El principal propósito es proveer de las capacidades básicas de una Arquitectura Orientada a Servicios (SOA).

  Para lograr esta funcionalidad se reunen aquí varios subproductos como el Bus de Servicios de Empresa, el servidor de aplicaciones, el repartidor de mensajes,el servidor de datos y el servidor de procesamiento.

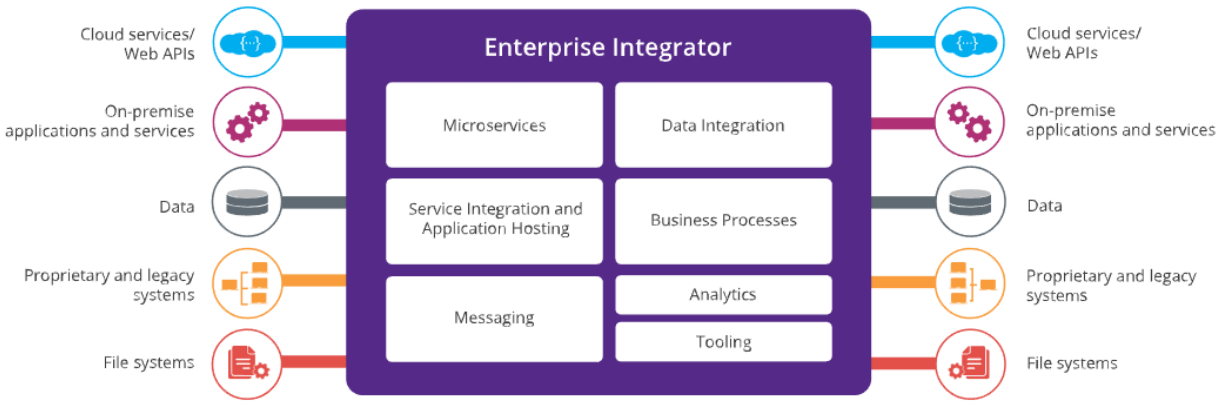


Figura 3-3 Esquema funcional del servidor de Integración

Principales caracterísitcas de esta solución:

- Optimizar procesos, reducir costes y cuellos de botellas
- Integración con sistemas legados evitando el coste de migraciones a nuevas tecnologías o arquitecturas.
- Integración de los sistemas de computación en nube.
- Creación de nuevos servicios web.
- Creación de alertas de monitorización para todos los recursos asociados al componente de integación.

- **Servidor de identidad [5]**

  Este producto se centra en la gestión de la identidad de los usuarios a través de aplicaciones de ámbito empresarial, servicios e interfaces públicas. Gracias a la gran cantidad de funcionalidades que implementa es muy sencillo definir políticas para el control de acceso de usuarios.

Figura 3-4 Esquema funcional del Servidor de Identidad

Las características básicas de este producto son:

- Implementación de single sign-on y federación de identidad mejorando la experiencia de uso al mantener las sesiones de usuarios entre distintos servicios.
- Integración de los últimos protocolos de seguridad, como el doble factor de autenticación.
- Control de acceso basando en roles y atributos (RBAC y ABAC).
- Monitorización, informes y auditorías.

- **Internet de las cosas (IOT) [6]**

Este módulo se centra en la integración de dispositivos móviles, creando una plataforma flexible sobre la que trabajar.

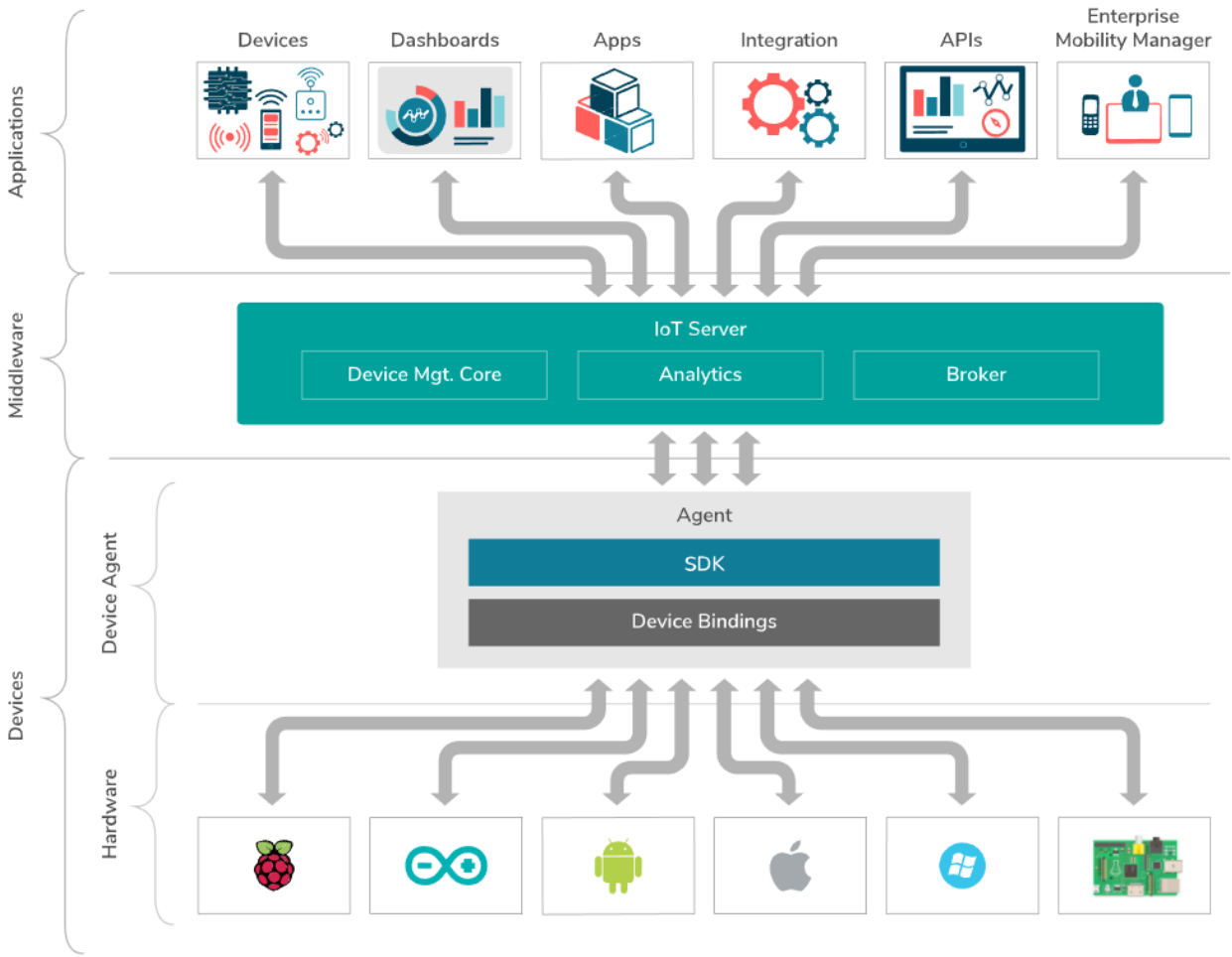


Figura 3-5 Esquema de características del Servidor del Internet de Las Cosas

Las características de esta solución son:

- Gestión automática de dispositivos de iOS, Android y Windows
- Control de acceso basado en permisos y monitorización del mismo.
- Procesamiento de eventos de protocolos IOT.

A modo de ejemplo, esta solución se utiliza actualmente por empresas de telecomunicaciones como *Verizon* para la gestión de pagos en dispositivos móviles.

3.2. Middleware

3.2.1. Definición

El término *middleware* [7] aplica al software que provee una capa de abstracción a las aplicaciones que lo utilizan, enmascarando así la heterogeneidad de las capas subyacentes y reduciendo la complejidad del desarrollo de sistemas distribuidos.

La principal característica que tienen todos los tipos de *middlewares* disponibles hoy en día es la de abstraer funcionalidades de otras capas de *software*, por lo que hace que el desarrollo de sistemas distribuidos complejos resulte una tarea mucho más asequible.

3.2.2. Carbon

Carbon [8] es el *middleware* que ofrece *WSO2* sobre el que se pueden instalar todos los servicios disponibles.

Dado que éste es el componente base de la arquitectura de WSO2, está diseñado para ser flexible dependiendo de la infraestructura donde se quiera desplegar.

Estas son algunas de sus características:

- Diseño escalable (vertical y horizontalmente) en topologías de cluster
- Descubrimiento dinámico de servicios
- Despliegues con topologías de alta disponibilidad
- Autenticación de usuarios basada en roles y atributos

Estos son los diferentes servicios disponibles para instalar sobre *Carbon*:

Figura 3-6 Esquema de servicios instalables sobre *Carbon*

3.3. Arquitectura Orientada a Servicios (Service – Oriented Architecture)

La Arquitectura Orientada a Servicios (*SOA* por sus siglas en inglés) [9, 10, 11, 12] es un modelo de arquitectura basado en **el paradigma de diseño orientado a servicios**. Las soluciones SOA han sido creadas para satisfacer los nuevos objetivos de negocio que han ido surgiendo a medida que el desarollo y arquitectura de las aplicaciones distribuidas ha ido evolucionando.

3.3.1. Diseño y desarrollo de SOA

La metodología de modelado y diseño para aplicaciones *SOA* se conoce como el análisis y diseño orientado a servicios. La arquitectura orientada a servicios, generalmente basada en tecnologías web, es un marco trabajo conceptual que establece una estructura de diseño para el desarrollo e integración de aplicaciones distribuidas.

Cuando nos referimos a una arquitectura orientada a servicios se está hablando de un juego de servicios web residentes en una red de computación. Existen diversos estándares relacionados con los servicios web, incluyendo los siguientes:

- **XML:** Siglas en inglés de eXtensible Markup Language. Es un lenguaje de marcas desarrollado por el World Wide Web Consortium utilizado para almacenar datos de forma legible.
- **HTTP:** Protocolo de comunicación que permite las transferencias de información en la World Wide Web.
- **SOAP:** Protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
- **REST:** Estilo de arquitectura de software para servicios web.

Hay que considerar , sin embargo, que un sistema SOA no necesariamente utiliza estos estándares para ser orientado a servicios, aunque lo más habitual es su uso.

SOA apuesta por los servicios como base de la arquitectura. Estos diferentes servicios tienen características en común como: reutilización, capacidad de composición (un servicio hace uso de otro servicio o viceversa), estandarización, abstracción, autonomía, etc...

3.3.2. Tipos de Servicios

Antes de nada es necesario aclarar que existen diferentes clasificaciones (o modelos) de servicios. La que vamos a exponer a continuación es probablemente la más sencilla y el resto de modelos suelen ser derivados de ésta.

Distinguiremos 3 tipos de servicios:

- Servicios de utilidad
- Servicios de entidad
- Servicios de tarea

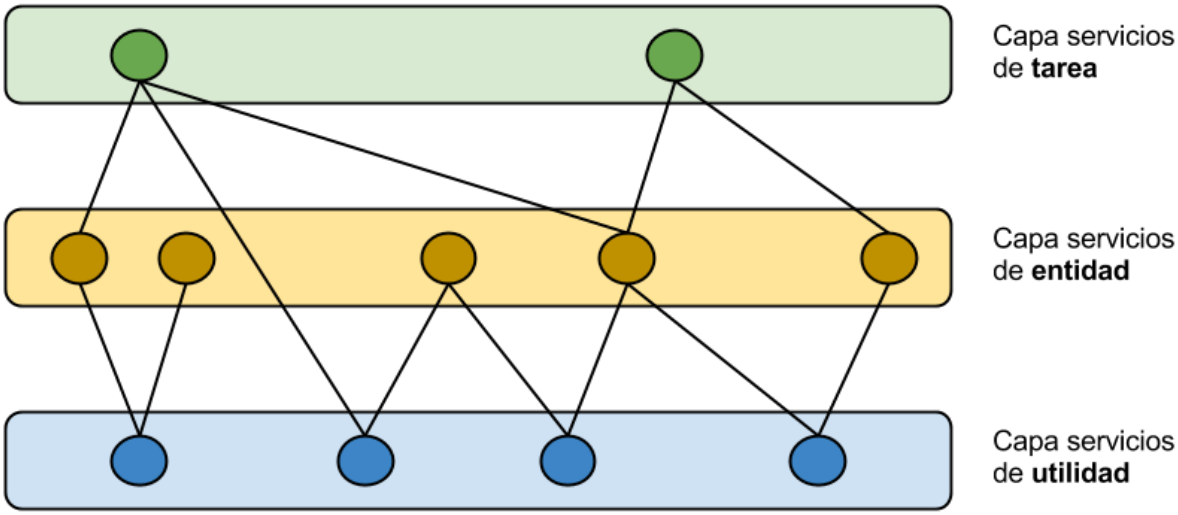


Figura 3-7 Esquema de tipo de servicios SOA

3.3.2.1. Servicios de utilidad

Los servicios de utilidad son aquellos que encapsulan una funcionalidad concreta, pudiendo ser válidos para un gran número de casos de uso**.** Son servicios que no tienen por qué cubrir una necesidad concreta de negocio.

Idealmente, estos servicios deben de ser altamente reusables.

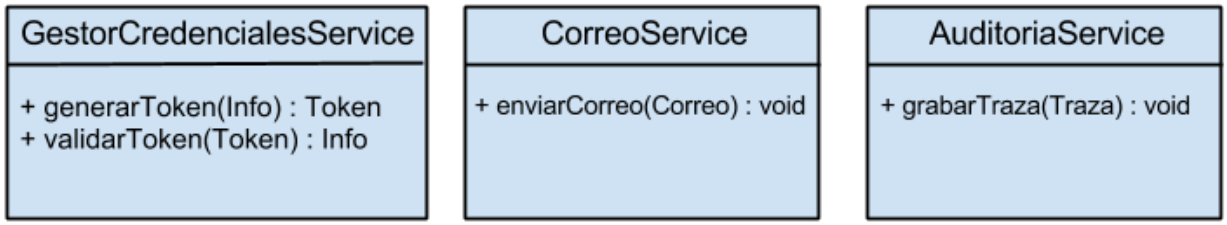


Figura 3-8 Ejemplo de servicio *SOA* de utilidad

3.3.2.2. Servicios de entidad

Los servicios de entidad son aquellos que están centrados en el contexto de las **entidades de negocio.** Al contrario que ocurre con los servicios de utilidad, son servicios que están expresamente diseñados para ser aplicados a contextos de negocio, aportando así funcionalidades muy concretas.

Evidentemente, estos servicios variarán en función del negocio concreto y las entidades que representa su actividad. A continuación, se muestra un ejemplo:

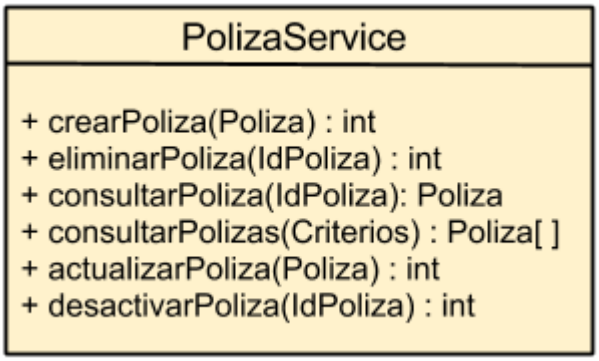


Figura 3-9 Ejemplo de servicio *SOA* de entidad

3.3.2.3. Servicios de tarea

Los servicios de tarea son aquellos que engloban un proceso de negocio apoyándose (normalmente) en servicios de más bajo nivel como los servicios de utilidad o de entidad. Suelen implementar un flujo de trabajo entre los servicios que subyacen a este. Una característica importante de estos servicios es que no suelen ser servicios con un grado de reutilización tan alto como los servicios de entidad o utilidad.

Es muy importante tener una buena base de servicios de entidad y utilidad sobre los que apoyar los servicios de tarea, pudiendo así ofrecer una respuesta al cambio de forma ágil.

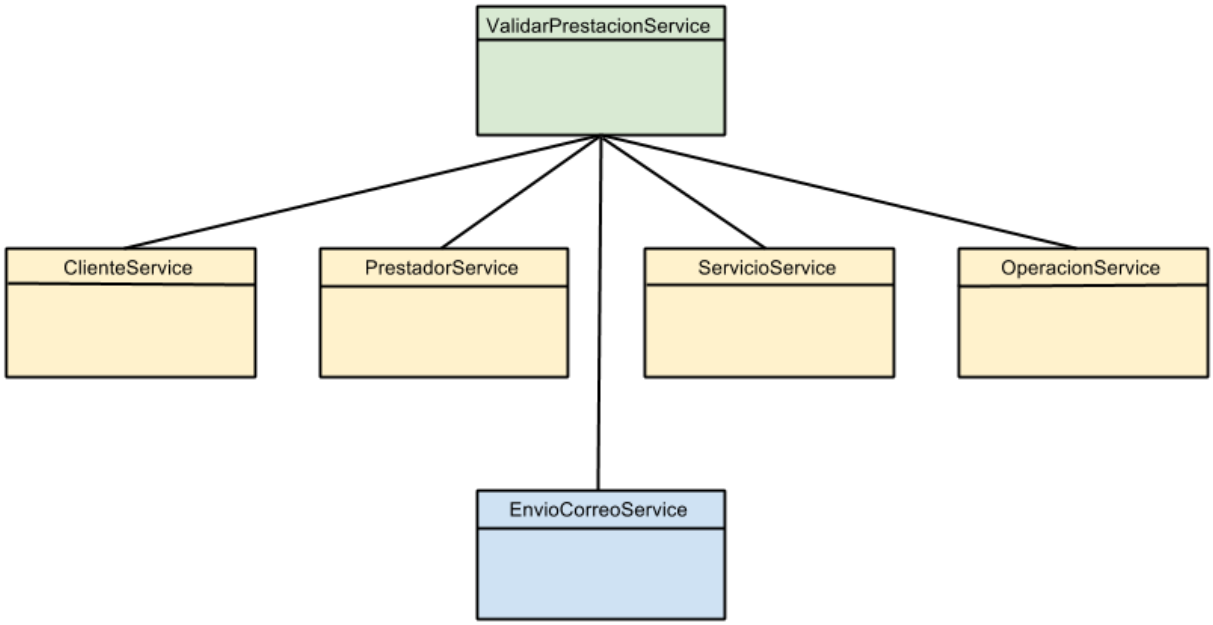


Figura 3-10 Ejemplo de servicio SOA de tarea

Como se ve en la imagen, este servicio concreto ha de estar apoyado en diferentes servicios de utilidad, en concreto, en las entidades de negocio que intervienen en el proceso y en algún servicio de utilidad (como el de enviar un correo electrónico).

3.3.2.4. Beneficios

Algunos de los principales beneficios que aporta SOA son:

- Aplicaciones flexibles, reutilizables y adaptables al cambio
- Reducción de costes

- Mejora en los tiempos de realización de cambios en procesos
- Facilidad para evolucionar a modelos de negocios basados en externalización

3.3.2.5. Diferencias con otras arquitecturas

Al contrario que las arquitecturas orientadas a objetos, las *SOA* están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma en la que estén desplegados y del lenguaje de programación (por ejemplo, WSDL).

Con esta arquitectura se pretende que los componentes software desarrollados sean muy reutilizables, ya que la interfaz se diseña en base a estándares de la industria.

3.4. WSO2 Servidor de Identidad – Identity Server

Junto con el Bus de Servicios de Empresa, el cual se introduce en el apartado 3.5, el servidor de identidad ha sido utilizado en el despliegue del caso práctico del proyecto.

Actualmente, WSO2 ofrece soluciones para el despliegue del servicio sobre servidores, plataformas de *cloud* y contenedores de microservicios.

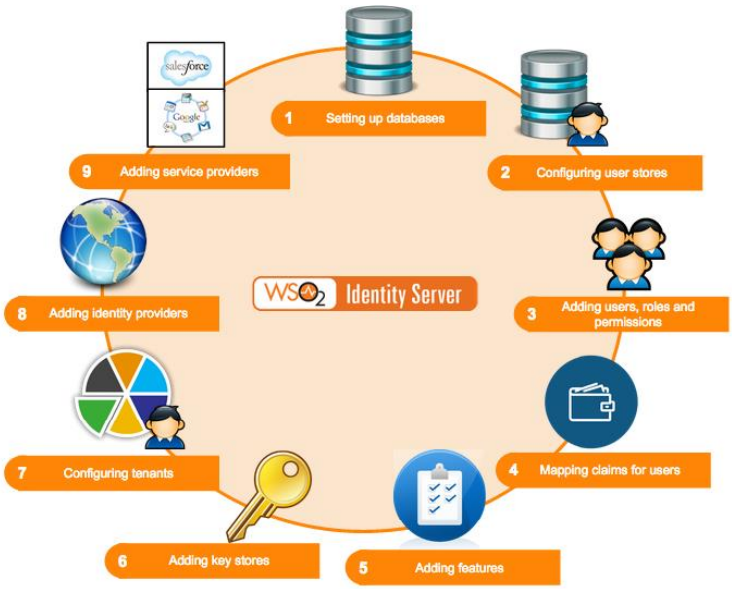


Figura 3-11 Componentes incluidos en el Servidor de Identidad

El servidor de identidad (**a partir de ahora *IS***) es completamente de código libre y está desarrollado bajo la licencia de Apache 2.0. La frecuencia de actualizaciones del servidor es mensual. El código fuente del mismo puede encontrarse aquí:

https://github.com/wso2/product-is

Las caracterísiticas más destacables de este producto son:

- Proporciona una interfaz (API) para la integración de las caracterísiticas de gestión de identidad en cualquier aplicación. Pensado para entornos SOA, optimizado para la integración con otras soluciones WSO2.
- Permite Single Sign-On (SSO) basado en OpenID, SAML2, y Kerberos
- Proporciona una aplicación web para la gestión de usuarios, perfiles, claves y proveedores de servicios
- Incluye repositorios de información de usuarios (LDAP) y puede ser configurado para utilizar otros (Microsoft Active Directory)
- Facilita la gestión de perfiles de usuarios, con múltiples perfiles por usuario
- Permite bloqueo de cuentas tras un número determinados de intentos fallidos de acceso
- Políticas para validación de claves
- Permite realizar un acceso a interfaces públicas basado en roles (RBAC – Role Based Access Control) y atributos (ABAC – Atribute Based Access Control).
- Control de acceso via XACML, WS-Trust, OpenID, y gestión de solicitudes
- El servidor de identidad puede ser configurado como un punto de administracion de políticas, el cual permite:
    - Establecer políticas de grano fino via XACML
    - Editar políticas por medio de una interfaz gráfica
    - Verificar las políticas creadas mediante herramientas
    - Soportar múltiples puntos de información de políticas (PIP)
    - Distribuir políticas a distintos puntos de decisión de políticas (PDP)
- El servidor de identidad tambien puede ser configurado como un punto de decision de políticas (PDP), el cual permite:
    - Almacenar en memorias de caché las decisiones y los atributos de políticas
    - Notificar de actualizaciones de políticas
- Include un punto de administracion de políticas para gestionar diferentes puntos de decisión de políticas

3.5. WSO2 Bus de Servicios de Empresa – Enterprise Service Bus

Por último, pero no por ello menos importante, es el turno de hablar del Bus de Servicios de Empresa [13]. Este producto ofrece capacidades de integración de datos sin necesidad de desarrollar aplicaciones específicas para ello.

Básicamente, lo que integra este servicio es un motor de mensajería configurable, haciendo muy sencilla la integración entre todo tipo de sistemas, sin necesidad de añadir más servidores destinados a la integración de datos. Además, al igual que la mayoría de productos de WSO2, incluye la monitorización de los mensajes enviados, dando la posibilidad de monitorizar de una manera muy clara las comunicaciones de redes internas.

Referenciando algunas de sus características:

- Más de 160 tipos de conectores, incluyendo categorías de pagos online, redes sociales o sistemas legados.
- Transporte: HTTP/S, webSocket, POP, IMAP, SMTP y más.

- Capacidad para utilizar servicios de almacenaje de datos, bases de datos relacionales u hojas de cálculo de Google.
- Capacidad de transformar peticiones de un estándar a otro para realizar la integración entre distintos sistemas.
- Alta disponibilidad, escalabilidad, rendimiento y estabilidad.

3.6. Protocolo Kerberos

3.6.1. Definición

Kerberos [14, 15, 16] es un protocolo de red creado por el *MIT* (Instituto Tecnológico de Massachusetts) que utiliza criptografía de claves simétricas para autenticar usuarios contra servicios de red, evitando así tener que enviar las contraseñas de los mismos a traves de una red insegura de datos.

3.6.2. Motivación

Kerberos fue diseñado para proteger los servicios de red del proyecto ***Athena*** [17], el cual fue un proyecto de cooperación del *MIT* junto con otras empresas para crear un entorno de computación distribuida. Kerberos está basado en el protocolo *Needham-Schroeder*, uno de los primeros protocolos diseñados que utilizaban claves simétricas para compartir datos por una red de datos insegura.

3.6.3. Conceptos importantes

En este apartado se comentan los detalles más importantes para un correcto entendimiento del funcionamiento del protocolo Kerberos. Un caso real en el que se detalla el paso de mensajes entre las distintas entidades se presenta al final de este capitulo.

En un escenario clásico en el que Kerberos securiza el acceso a un servicio, los elementos involucrados en la comunicación son los siguientes:

- El **cliente** es el agente de usuario que accederá al servicio web de una forma segura.
- El **Servidor**, que contiene el servicio al que el usuario desea acceder.
- El **Centro De Distribución de Claves** (*KDC* a partir de ahora). Es lo que se suele conocer tambien como el controlador del dominio. Dentro de este servidor se ejecuta el proceso principal de Kerberos, el cual provee los dos principales servicios de Kerberos:
  - **Servidor de Autenticación** (AS a partir de ahora).
  - **Servidor de Reparto de Tickets** (TGS a partir de ahora).

Es clave detallar que Kerberos trabaja con dominios o "reinos". Una vez instalamos Kerberos, en la primera configuración de este, deberemos configurar el dominio de red donde trabajará. Existen a su vez configuraciones inter-dominio más complejas, las cuales no se abarcan en el desarrollo de este trabajo.

Otros puntos importantes a destacar:

- Con cada interacción, el cliente recibe dos mensajes. Uno que puede desencriptar y otro que no.
- El servicio al que quiere acceder el cliente nunca se comunica directamente con el *KDC*.
- A cada contraseña almacenada en el *KDC*, se le realiza un *hash* y se le añade una parte generada aleatoriamente por el *KDC*. Esta parte aleatoria se conoce como "*salt*".
- El *KDC* en sí mismo está encriptado con una contraseña maestra para añadir otro punto de seguridad a la hora de obtener claves de la base de datos donde almacena las contraseñas de los usuarios
- Todas las claves que se utilizan en el paso de mensajes de una entidad a otra son encriptadas

siguiendo los algoritmos DES-CBD-MD5, con una longitud de clave de 56 caracteres. Estos son los que por defecto Kerberos debe soportar, aunque se puede configurar con otro tipo de algoritmos como RC4-HMAC, en los cuales la longitud de la clave pasa a ser 128 caracteres.

- Kerberos utiliza *principales* para identificar de forma unívoca a usuarios o servicios autenticados mediante el protocolo de Kerberos. Un *principal* sigue el siguiente formato:

```
{Nombre de usuario/servicio} / {instancia} / {Dominio de Kerberos}
```

  Donde el **nombre** será un identificador único que represente a ese usuario o servicio, la **instancia** es un identificador de la máquina donde está alojado el servicio o donde se encuentra el usuario y el **dominio** es el reino de Kerberos al que el usuario o servicio pertenece. Es importante mencionar que cada *principal* tiene asociado una contraseña, ya sea para un usuario o para un servicio.

  Durante el desarrollo del caso práctico será necesario crear un identificador del servicio que el cliente utilizará como identificador del servicio al que quiere conectarse.

3.6.4. Ventajas de Kerberos

Los servicios de red más convencionales usan esquemas de autenticación basados en contraseñas. Tales esquemas requieren que los usuarios se autentiquen en servicios proporcionando sus credenciales de usuario, que normalmente viajan por la red sin estar encriptadas. Además, con estos es posible conocer si el usuario que está accediendo al servicio es legítimo.

Por lo tanto, el primer objetivo de Kerberos es el de eliminar la transmisión a través de la red de información de autenticación sin cifrar. Basado en el uso de claves públicas/privadas para compartir claves simétricas Kerberos consigue establecer una comunicación segura.

3.6.5. Desventajas de Kerberos

A pesar de que Kerberos elimina una amenaza de seguridad común, puede ser difícil de implementar por una variedad de razones:

- La migración de contraseñas de usuarios desde una base de datos de contraseñas estándar UNIX a una base de datos de contraseñas Kerberos puede ser tediosa y no hay un mecanismo rápido para realizar esta tarea.
- Kerberos supone que cada usuario es de confianza pero que está utilizando una máquina no fiable en una red no fiable. Su principal objetivo es el de prevenir que las contraseñas no encriptadas se envíen a través de la red. Sin embargo, si cualquier otro usuario no autorizado, tiene acceso a la máquina que emite los tickets usados para la autenticación (Centro de Distribución de Claves, *KDC*), el sistema completo de Kerberos está en riesgo.
- Para que una aplicación use Kerberos, esta debe estar diseñada y programada para realizar las llamadas a los distintos servicios de *Kerberos*. Por lo tanto, adaptar una aplicación al uso de Kerberos es una tarea a considerar, en términos de tiempo de desarrollo.

3.6.6. Paso de mensajes

Sin más, vamos a ver un ejemplo de como sería una comunicación entre cliente y servidor utilizando Kerberos, basándonos en el esquema antes propuesto. En siguientes capítulos se verá como este esquema cambia al introducir en él los productos de WSO2.

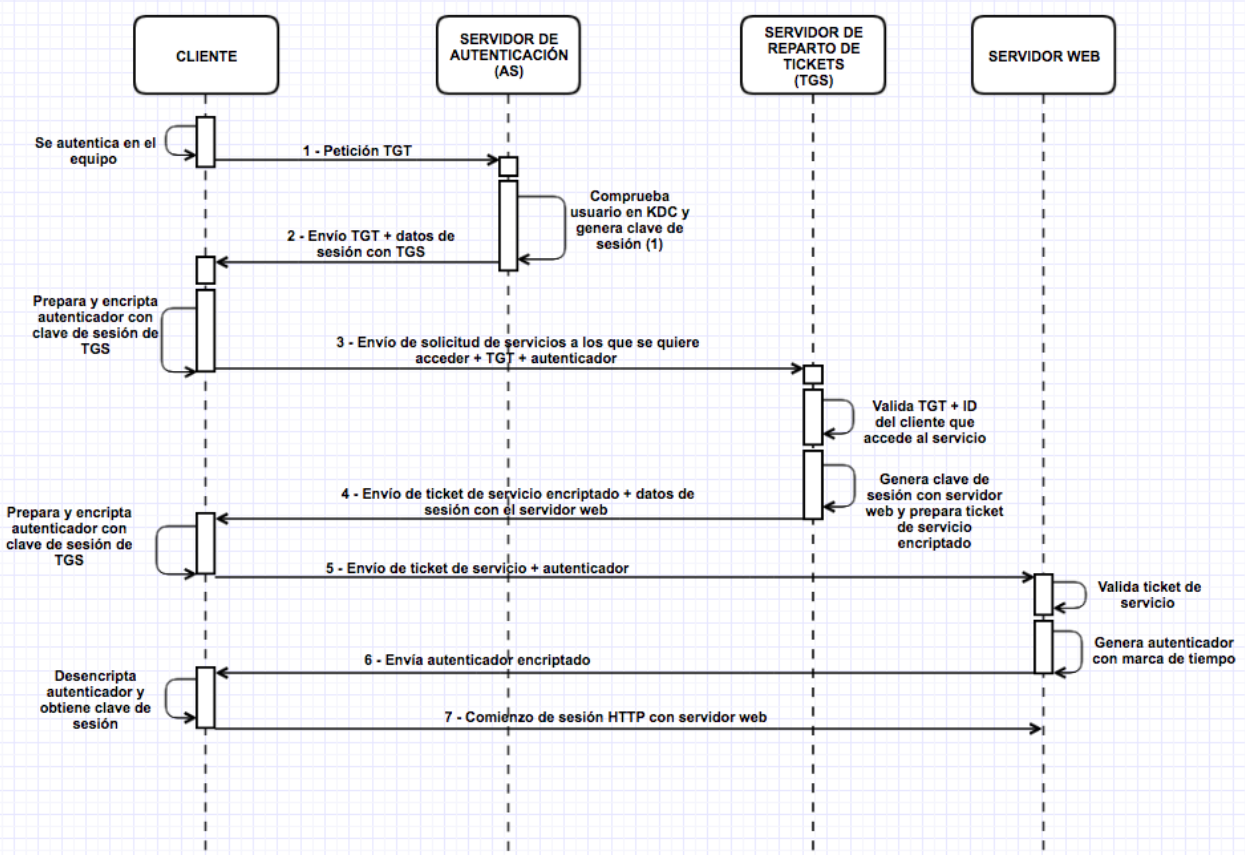


Figura 3-12 Diagrama de paso de mensajes de Kerberos

Petición TGT para acceder al TGS

Cuando el cliente quiere acceder al servicio web, este introduce sus credenciales en el equipo cliente. Es requisito necesario que previamente ese usuario haya sido creado en el sistema (AS) por un administrador.

Una vez el usuario inicia sesión para acceder al servicio, se envía un mensaje al servidor de autenticación (AS) en texto plano pidiendo el *Ticket-Granting Ticket,* o *TGT.* Este mensaje del cliente en texto plano incluye:

- El nombre/ID del cliente
- El nombre/ID del servicio al que solicita acceso (En este caso, el cliente solicita acceso al TGS, que es el que le dará acceso al servicio en el que está interesado)
- La dirección de red del cliente
- El tiempo de vida válido para el *TGT*

Envío de datos *TGT* y datos de sesión con el *TGS*

Una vez el *AS* recibe esta petición, comprueba si el usuario está en la base de datos del *KDC*. Este sólo comprueba si el usuario existe, no comprueba nada relacionado con las credenciales del mismo.

Si durante la comprobación no ha habido ningún error (como que el usuario no se haya encontrado), el *AS* genera una **clave de la sesión** para usarla entre el cliente y el *TGS*.

El AS devolverá dos mensajes al cliente. Un mensaje es el ***TGT***, que contiene:

- El nombre/ID del cliente
- El nombre/ID del *TGS*
- Una marca de tiempo
- La dirección IP del cliente
- El tiempo de vida del *TGT*
- La clave de la sesión entre el cliente y el *TGS*

Este mensaje irá encriptado con la **clave secreta del *TGS***, por lo que el cliente no podrá ver el contenido de este mensaje.

El otro mensaje, que es un subconjunto del anterior, contiene:

- El nombre/ID del *TGS*.
- Una marca de tiempo
- Tiempo de vida
- La clave de la sesión entre el cliente y el *TGS*

Ahora, este mensaje se encripta con la clave simétrica secreta del cliente. Es importante recordar que la clave simétrica secreta del cliente son las credenciales de usuario que el cliente tiene asignado para acceder a ese serivicio web.

Una vez el cliente reciba estos mensajes, guardará en su caché de credenciales el *TGT* encriptado.

Envío de solicitud de servicios a los que se quiere acceder junto con el *TGT* y el autenticador

En este punto, el cliente enviará **dos mensajes y un autenticador al *TGS***.

Primero, el cliente envía un mensaje que se utilizará a modo de autenticador, **el cual se encripta con la clave de la sesión con el *TGS*** y contiene:

- Nombre/ID del cliente
- Marca de tiempo

Una vez el usuario se encuentre autenticado contra el TGS, se envía otro mensaje que contiene:

- El Nombre Del Servicio/ID del servicio web al que el cliente quiere acceder
- El tiempo de vida para el Ticket de Acceso al Servicio Web

El otro mensaje que el cliente envía es el *TGT*, el cual tiene almacenado en su caché de credenciales de la interacción anterior.

Envío de ticket de servicio encriptado y datos de sesión con el servidor web

Una vez el TGS recibe estos mensajes, comprobará que el servicio al cual el cliente quiere acceder existe. Si existe, desencripta el *TGT* con su clave secreta. A partir de ahora el *TGS* es capaz de desencriptar el autenticador que el cliente le envió ya que dispone de la clave de la sesión del cliente y el *TGS*. Una vez lo haya desencriptado, hará lo siguiente:

- Compara el ID del cliente del autenticador con el del *TGT*
- Compara la marca de tiempo del autenticador con la del *TGT* (tiene una tolerancia en la marca de tiempo de 2 minutos por defecto, pero es modificable)
- Comprueba si el *TGT* ha expirado
- Comprueba que el autenticador **no** está en la caché del *TGS*, evitando así un tipo de ataques basado en la repetición del envío de estos mensajes
- Si la dirección de red de la petición original no es nula, compara la dirección de red del cliente con la del *TGT*

Si todo es correcto, el *TGS* **genera aleatoriamente la clave de sesión con el Servicio Web**, y prepara el ticket del Servicio Web, que contiene:

- El Nombre/ID del cliente
- El Nombre/ID del Servicio Web al que el cliente quiere acceder
- La dirección de red del cliente
- Una marca de tiempo
- El tiempo de vida válido para este ticket
- La clave de la sesión con el Servicio Web

Y **encripta el ticket con la clave secreta del Servicio Web**. Por lo tanto, el *TGS* enviará dos mensajes. El ticket para el Servicio Web encriptado y el siguiente mensaje, que contiene:

- El Nombre/ID del servicio Web
- Una marca de tiempo
- El tiempo de vida
- La clave de la sesión con el Servicio Web

Envío de ticket de servicio y autenticador

Para acceder al Servicio Web, la máquina del cliente prepara otro autenticador que contiene:

- El Nombre/ID del cliente
- Un marca de tiempo

Y **va encriptado con la clave de la sesión con el Servicio Web**. Entonces el cliente enviará el autenticador y el ticket del Servicio Web, el cual está encriptado y el cliente no puede ver el contenido.

Una vez recibidos los mensajes por el servidor web, este **desencripta el ticket del Servicio Web con su clave privada**. De ese mensaje obtiene la clave de la sesión que compartirán él y el cliente, por lo tanto, lo primero que hace con esta clave de sesión es desencriptar el autenticador enviado por el cliente.

Despues, comprobará lo siguiente:

- Compara el ID del cliente del autenticador con el del ticket recibido
- Compara las marcas de tiempo del autenticador con la del ticket (con una tolerancia de dos minutos, por defecto)
- Comprueba que el ticket **no** haya expirado
- Comprueba que no tiene el autenticador en la caché del servidor web
- Si la dirección de red no es nula en la petición original, compara la dirección de origen con la del cliente.

Envío de autenticador encriptado

Si la verifcacion anterior tuvo éxito, el servidor web envía un autenticador que contiene el ID del Servicio Web y una marca de tiempo. Este mensaje se envía encriptado con la clave de la sesión que ahora ambas entidades comparten.

Comienzo de sesión HTTP del cliente y el servidor web

Una vez el cliente recibe el autenticador, lo desencripta con la clave de la sesión que tiene almacenada en su caché HTTP. A partir de ahora el flujo de mensajes entre el cliente y el servidor irá encriptado.

4 DESARROLLO DEL PROYECTO

Este capítulo se dedica a la explicación del desarrollo de este caso práctico. Se comienza mostrando los distintos elementos que lo forman y cómo interactuan entre ellos, continuando con la configuración de las distintas maquinas virtuales y la puesta en marcha de los distintos servicios.

4.1. Escenario

El escenario sobre el que se va a trabajar está compuesto por tres entidades, las cuales son:

- **Servidor de identidad (WSO2 Identity Server - IS)**.

  Este producto de WSO2 es el que incluye todas las funciones de Kerberos que participan en la creación y distribución de tickets. En este servidor se incluye:

  - Una base de datos LDAP que almacena los usuarios creados junto a sus roles.
  - El centro de distribución de claves de Kerberos (Key Distribution Center - KDC), encargado de la creación, distribución y validación de los tickets de Kerberos.

- **Bus de servicios de empresas (WSO2 Enterprise Service Bus - ESB).**

  Este producto de WSO2 se utiliza en este escenario como el servidor de eco. En este caso práctico, **el ESB contiene el servicio de eco al que el cliente quiere acceder**.

  En este servicio de eco la autenticación de usuarios será configurada mediante el protocolo de Kerberos, haciendo así uso del IS como KDC de Kerberos.

- **Cliente del servicio eco**.

  El cliente es una aplicación desarrollada en Java la cual utiliza las librerías de WSO2 para gestionar la autentación de un usuario contra el servicio de eco a través del protocolo Kerberos. El detalle de su funcionamiento será explicado en apartados posteriores, pero al final de esta sección se puede ver un esquema donde se detalla la interacción con el IS y el ESB.

Cada uno de estos elementos será desplegado en una máquina virtual diferente, por lo tanto, para el desarrollo de este caso práctico será necesario configurar tres máquinas virtuales.

A continuación se muestra un diagrama con la interacción entre los distintos componentes que forman el escenario:

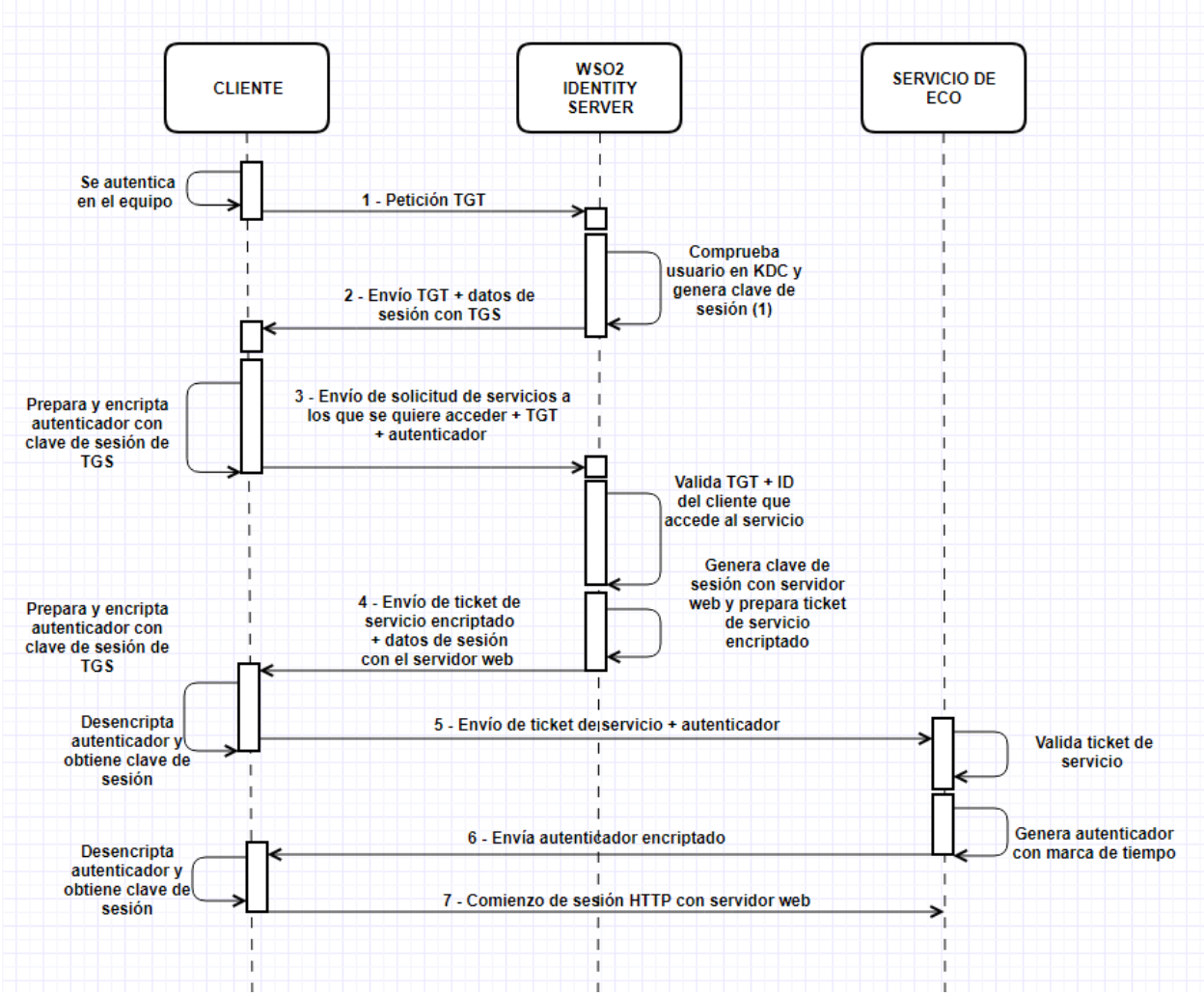


Figura 4-1 Diagrama de paso de mensajes de Kerberos con productos WSO2

Dado que el funcionamiento en detalle ha sido explicado en capítulos anteriores, se proporciona un breve resumen de los puntos más importantes de la comunicación entre los distintas elementos:

1. El cliente introduce nombre de usuario y contraseña en su ordenador para acceder al servicio de eco, que se encuentra acesible a través del ESB (**en este caso, el servicio de eco se encuentra desplegado en la misma máquina virtual que el ESB**). El cliente enviará peticion al IS, el cual contiene el servicio de autenticación de Kerberos (AS). Ahora, el IS actuará como el Centro de Distribucion de Claves (KDC)
2. El IS valida al usuario y le devuelve el TGS y otro mensaje con la clave de la sesión entre el TGS y el cliente .
3. Una vez el usuario recibe los dos mensajes del IS, este genera el autenticador y el ticket solicitando el TGT y se los envía de nuevo al IS. Ahora, el cliente pretende acceder al servicio de TGS.
4. Despues de validar el contenido de ambos mensajes enviados por el cliente, el TGS genera dos mensajes, incluyendo en ellos la clave de la sesión entre el servicio de eco y el cliente.
5. Una vez recibidos por el cliente, este prepara un nuevo autenticador y el último mensaje que le envió el TGS.
6. Cuando el ESB los recibe y desencripta la clave de la sesión con el cliente, le vuelve a enviar otro autenticador cifrado con la clave de la sesión, comenzando asi la comunicación cifrada entre el ESB y el cliente.

7. A partir de aquí comienza la comunicación HTTP encriptada entre los dos extremos.

4.2. Virtualbox

Para el desarrollo de este proyecto se utilizan máquinas virtuales con el sistema operativo Ubuntu, en la version 16.04. La elección de este sistema operativo se ha basado en la disponibilidad de los paquetes de Kerberos, siendo para los alumnos más sencilla la instalación de estos paquetes que en otras distribuciones Unix.

El software de virtualización escogido para el desarrollo de este caso práctico ha sido *Virtualbox*. Su página de descarga es:

https://www.virtualbox.org/wiki/Downloads

Antes de proceder a ejecutar las máquinas virtuales y comenzar a trabajar con ellas, es necesario definir una red privada en *Virtualbox*.

Para ello, es necesario ir a las opciones de configuración de *Virtualbox* y elegir sobre la pestaña de redes:

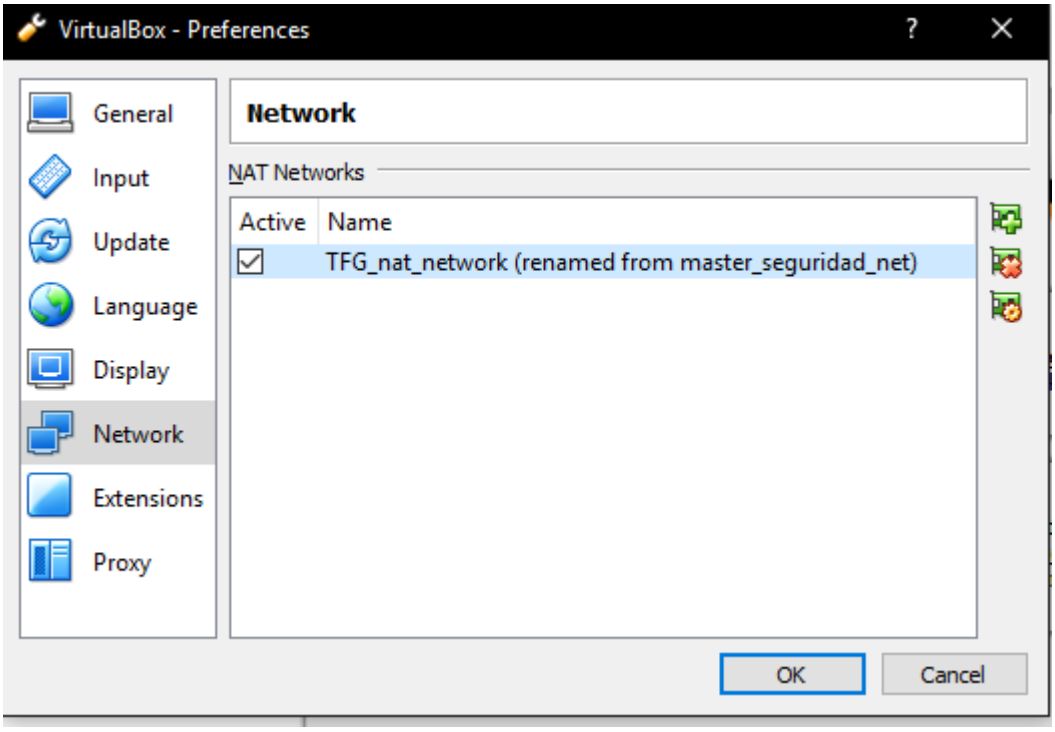


Figura 4-2 Creacion de red interna para máquinas virtuales en VirtualBox (I)

Ahora, es necesario elegir la opción con el símbolo "+" y crear una red *NAT*. Con esto, se está definiendo una subred privada la cual utilizarán las máquinas virtuales para obtener su dirección de red, y así conseguir conectividad entre ellas. La configuración de esta subred privada se detalla en la siguiente imagen:

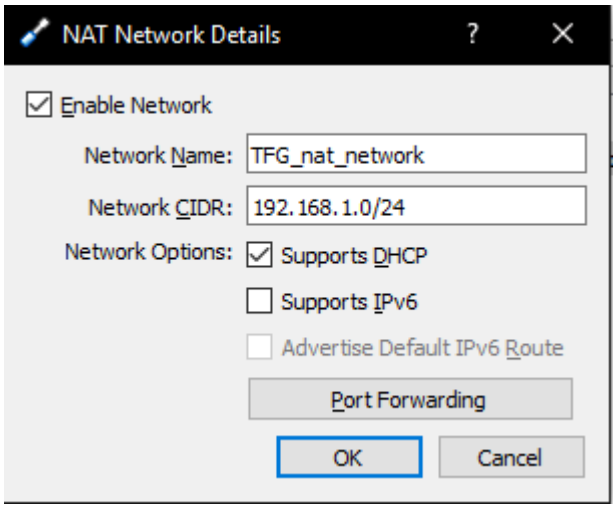


Figura 4-3 Creación de red interna para máquinas virtuales en VirtualBox (II)

A la hora de instalar las máquinas virtuales en *Virtualbox* solo será necesario seleccionar el archivo con formato `.ova`. Acto seguido, el asistente de *Virtualbox* se ejecutará:

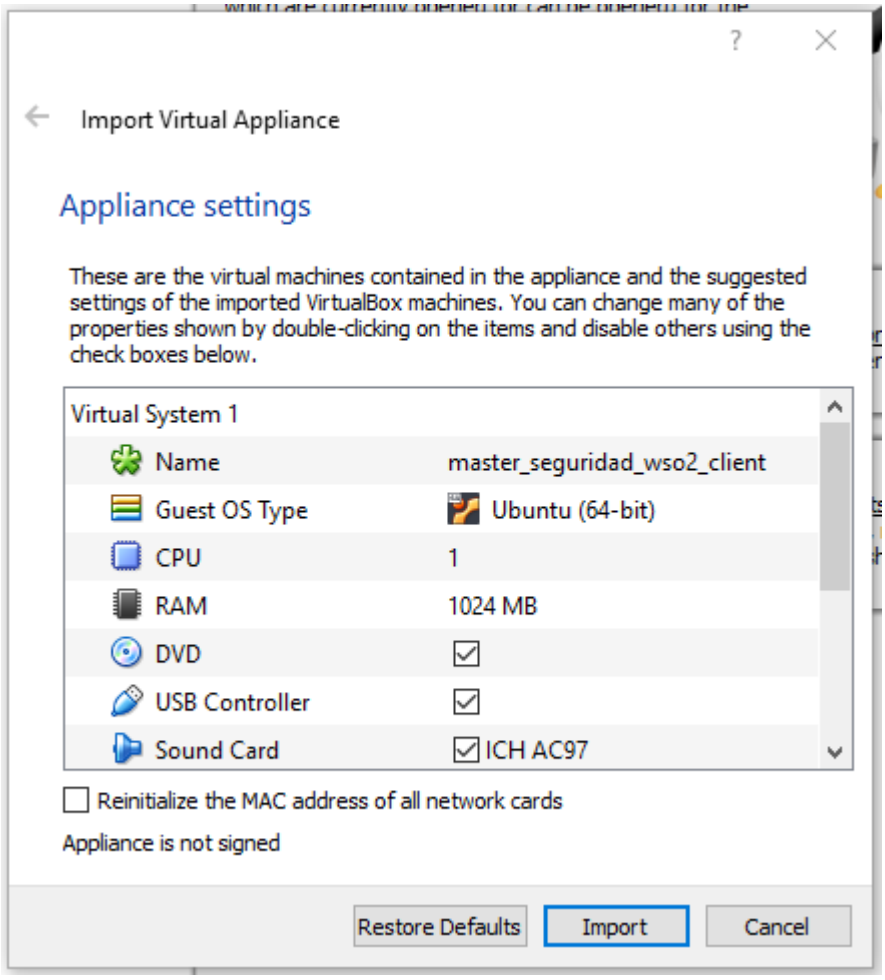


Figura 4-4 Importación de máquinas virtuales en VirtualBox

Seleccionando `Importar`, habrá terminado el proceso de instalación de la máquina virtual. Una vez hecho este paso, es necesario configurar en cada una de las máquinas virtuales la subred que se ha creado en pasos anteriores, por lo que será necesario hacer clic derecho sobre la máquina virtual en la interfaz de *Virtualbox* y

buscar las opciones relacionadas con la red.

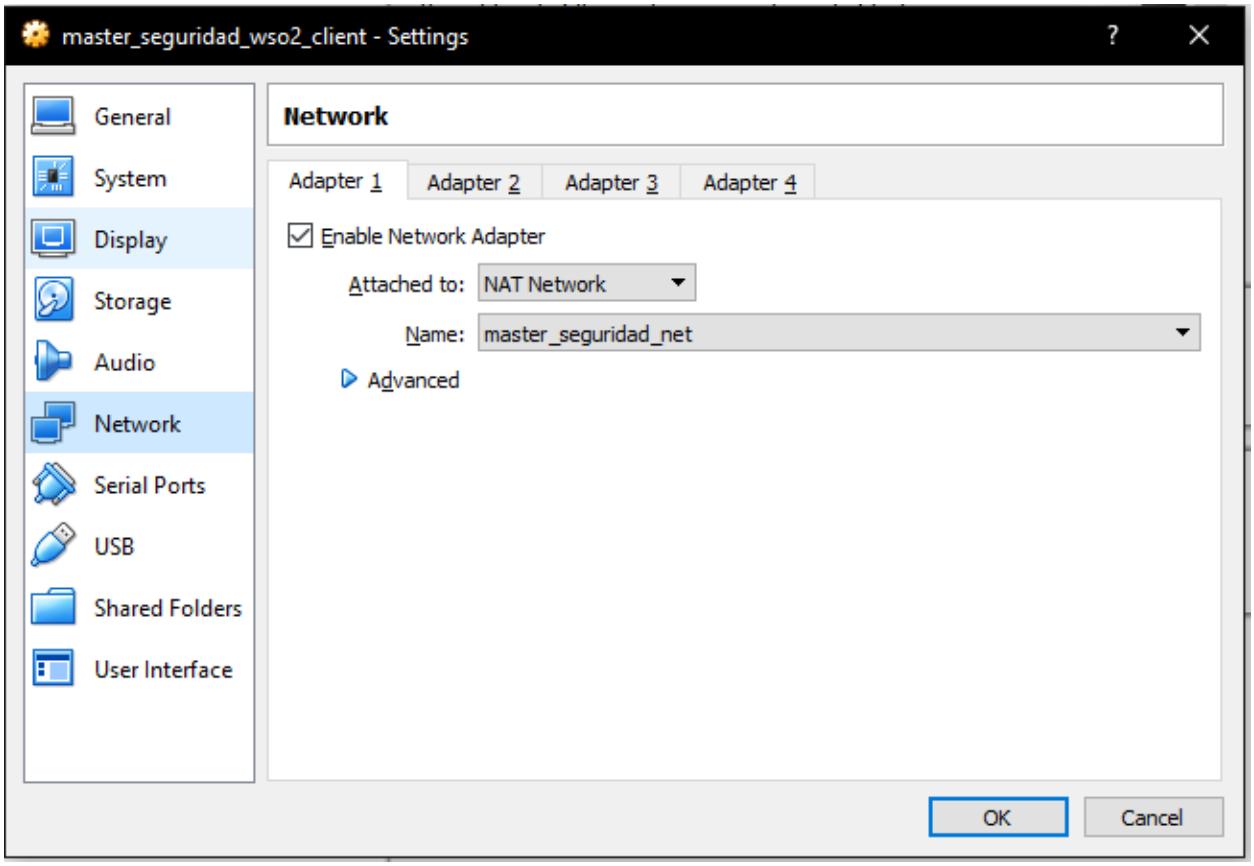


Figura 4-5 Ajuste de red interna para máquinas virtuales en VirtualBox

Como se aprecia en la imagen, es necesario seleccionar una red *NAT* y escoger el nombre creado anteriormente. Una vez modificado esto en las tres máquinas virtuales se puede comenzar a trabajar con ellas.

4.2.1. Información sobre máquinas virtuales

A continuación se adjunta la información sobre los usuarios y contraseñas de los distintos usuarios de las máquinas virtuales:

Máquina virtual	Información	Dirección de red
WSO2 IS	Usuario: **`wso2is`** – Contraseña: **`wso2is`**	192.168.1.4
WSO2 ESB	Usuario: **`wso2esb`** – Contraseña: **`wso2esb`**	192.168.1.6
Cliente	Usuario: **`client`** – Contraseña: **`client`**	192.168.1.5

Tabla 1 – Información sobre máquinas virtuales

Dado que es posible ejecutar este caso práctico dentro de un solo ordenador usando las diferentes máquinas virtuales, se adjunta tambien una lista de reenvío de puertos entre la máquina que ejecuta todas las máquinas virtuales y las propias máquinas virtuales, para que sea mas sencillo acceder a los diferentes servicios que se

lancen dentro de ellas.

Para configurar esta lista de puertos que seran reenviados entre las diferentes máquinas es necesario acceder al menú donde se configuró la red interna de las máquinas virtuales y utilizar la opcion de "*Port Forwarding*":

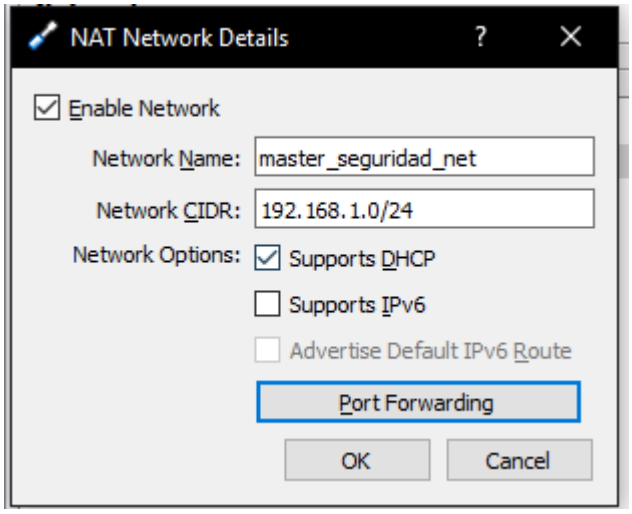
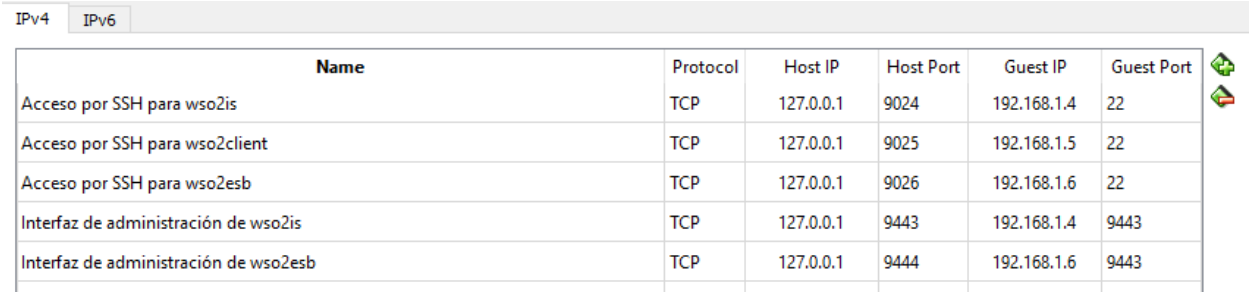


Figura 4-6 Menu de reenvío de puertos



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Acceso por SSH para wso2is	TCP	127.0.0.1	9024	192.168.1.4	22
Acceso por SSH para wso2client	TCP	127.0.0.1	9025	192.168.1.5	22
Acceso por SSH para wso2esb	TCP	127.0.0.1	9026	192.168.1.6	22
Interfaz de administración de wso2is	TCP	127.0.0.1	9443	192.168.1.4	9443
Interfaz de administración de wso2esb	TCP	127.0.0.1	9444	192.168.1.6	9443

Figura 4-7 Lista de reenvío de puertos para máquinas virtuales

4.3. Configuración de direcciones de red de las máquinas virtuales

Ya que para este proyecto no se ha utilizado un servidor DNS interno a la red local por simplicidad del escenario, es necesario modificar el archivo */etc/hosts* de todas las entidades del escenario.

Este archivo es usado por el sistema operativo para guardar la correspondencia entre **dominios de internet** y **direcciones IP.** Este es uno de los diferentes métodos que usa el sistema operativo para resolver nombres de dominios.

Si queremos desplegar una configuración distribuida debemos configurar los archivos de estas máquinas y añadir las siguientes líneas.

- IP del servidor IS y alias
- IP del cliente y alias
- IP del ESB y alias

El resultado final debe ser:

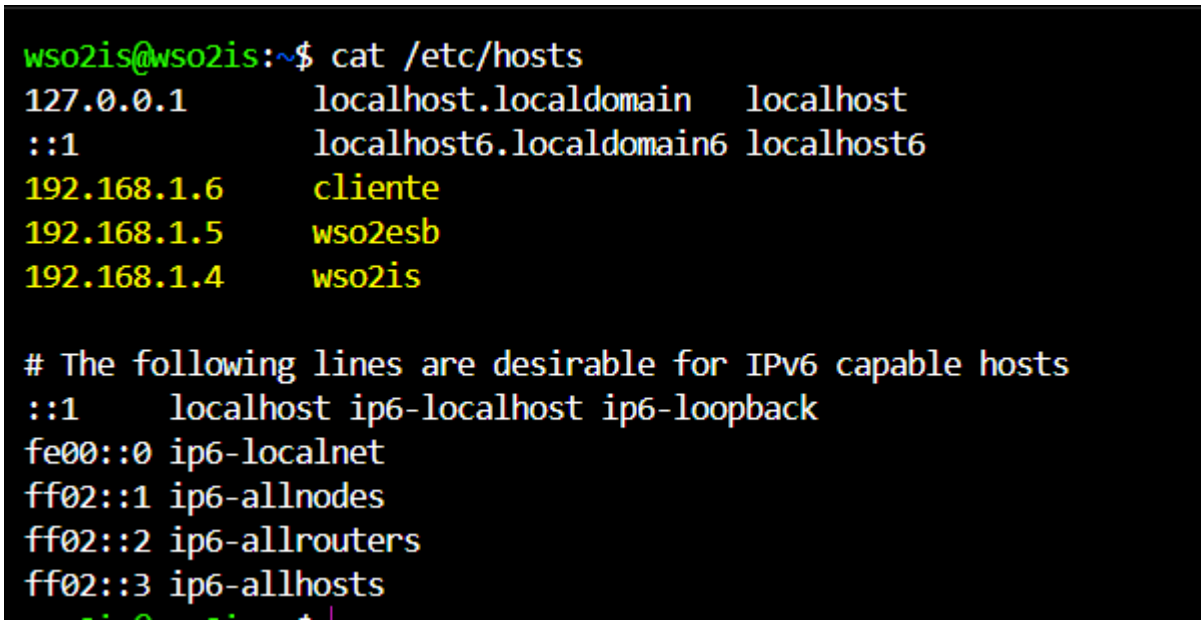


Figura 4-8 Archivo de sistema /etc/hosts

A modo de ejemplo, en la imagen anterior se ha configurado al servidor de identidad con el alias `wso2is`.

4.4. Máquinas virtuales de WSO2

Este apartado está dedicado a la preparación de las máquinas virtuales sobre las que se instalarán los productos de *WSO2,* ya sea el IS o el ESB

4.4.1 Requisitos de *software* a instalar

El *software* necesario para la preparación de esta máquina virtual es el siguiente:

- Java Development Kit 1.6u45
- Carbon - Versión 3.2.3

4.4.1.1. Java Development Kit 1.6

Java Development Kit (JDK) es el kit de desarrollo de *Java*, el cual provee de las herramientas necesarias para para el desarrollo de *software* en Java. El *IS*, el *ESB* y el cliente desarrollado en *Java* hacen uso de él, por lo que será necesario instalarlo en las tres máquinas virtuales con las que vamos a trabajar.

La descarga del **JDK 1.6 u 45** se realiza desde la siguiente página:

http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html#jdk-6u45-oth-JPR

Java SE Development Kit 6u45

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	65.46 MB	jdk-6u45-linux-i586-rpm.bin
Linux x86	68.47 MB	jdk-6u45-linux-i586.bin
Linux x64	65.69 MB	jdk-6u45-linux-x64-rpm.bin
Linux x64	68.75 MB	jdk-6u45-linux-x64.bin
Solaris x86	68.38 MB	jdk-6u45-solaris-i586.sh
Solaris x86 (SVR4 package)	120 MB	jdk-6u45-solaris-i586.tar.Z
Solaris x64	8.5 MB	jdk-6u45-solaris-x64.sh
Solaris x64 (SVR4 package)	12.23 MB	jdk-6u45-solaris-x64.tar.Z
Solaris SPARC	73.41 MB	jdk-6u45-solaris-sparc.sh
Solaris SPARC (SVR4 package)	124.74 MB	jdk-6u45-solaris-sparc.tar.Z
Solaris SPARC 64-bit	12.19 MB	jdk-6u45-solaris-sparcv9.sh
Solaris SPARC 64-bit (SVR4 package)	15.49 MB	jdk-6u45-solaris-sparcv9.tar.Z
Windows x86	69.85 MB	jdk-6u45-windows-i586.exe
Windows x64	59.96 MB	jdk-6u45-windows-x64.exe
Linux Intel Itanium	53.89 MB	jdk-6u45-linux-ia64-rpm.bin
Linux Intel Itanium	56 MB	jdk-6u45-linux-ia64.bin
Windows Intel Itanium	51.72 MB	jdk-6u45-windows-ia64.exe

Back to top

Figura 4-9 Descargas del *jdk 6u45*

Dependiendo de la arquitectura del procesador de la máquina en la que se instale el *Java Development Kit (JDK)* se escogerá entre una arquitectura u otra. Como se va a trabajar con una distribución de ubuntu, es necesario elegir el formato **.bin** para la descarga.

Una vez descargado se dan permisos de ejecución con el comando:

```
chmod +x jdk-6u45-linux-x64.bin
```

Y ejecutamos con la orden:

```
./jdk-6u45-linux-x64.bin
```

El ejecutable extraerá una carpeta en el directorio donde se haya ejecutado el comando con las herramientas que se necesitan.

Esta carpeta debemos copiarla al directorio donde se almacenan las distintas versiones de la máquina virtual de Java. La ruta es: **`/usr/lib/jvm/.`** Para copiar la carpeta al nuevo directorio se debe ejecutar:

```
cp -rf jdk1.6.0_45/ /usr/lib/jvm/
```

Hecho esto, si comprobamos el directorio `/usr/lib/jvm/` se comprueba que está ahí la carpeta:

```
wso2esb@wso2esb:~$ sudo ls /usr/lib/jvm/
jdk1.6.0_45
```

Figura 4-10 Instalación de *jdk 6u45*

Ahora debemos configurar la variable de entorno **`JAVA_HOME`** con la dirección a esa carpeta. Cuando el *IS* y el *ESB* se inicien comprobarán el valor de esa variable. Si apunta a una máquina virtual de java podrán iniciarse correctamente, si no, obtendremos un fallo con un mensaje de información sobre la configuración de esa

variable.

Para configurar la variable debemos introducir:

```
export JAVA_HOME=/usr/lib/jvm/jdk1.6.0_45/
```

Una vez hecho esto, la máquina virtual de Java estará correctamente configurada.

4.4.1.2. Carbon

Al igual que con todos los productos de WSO2, hay dos formas de instalarlos:

- Descargar *Carbon* e instalar todas las características del producto que se desea instalar (en este caso el servidor de identidad).
- Descargando *Carbon* configurado con todas las caracterísiticas del producto que se busca configuradas.

Los binarios de *Carbon* se puede encontrar en el siguiente enlace:

http://wso2.com/more-downloads/carbon/

En este caso se utilizará la versión **3.2.2:**

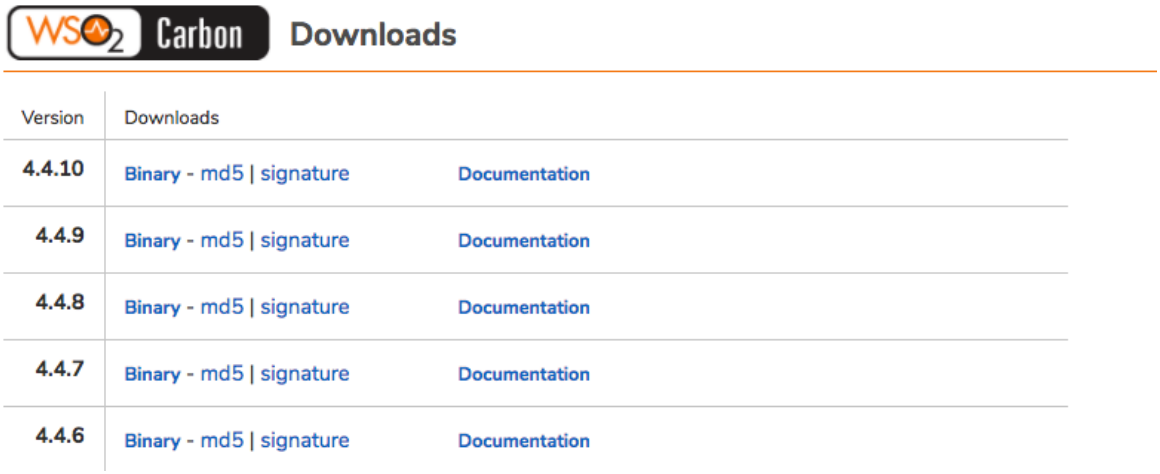


Figura 4-11 Descarga de *Carbon*

Pulsando sobre *Binary* hará que comience la descarga automática de *Carbon*, el cual estará comprimido en formato *.zip.* Una vez descargardo y descomprimido tendremos la carpeta que contiene el servidor de *Carbon.*

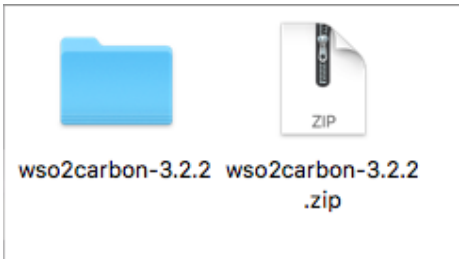


Figura 4-12 Descompresión de *Carbon* en ordenador local

La estructura de carpetas de *Carbon* (y de todos los productos de WSO2) es la siguiente:

```
wso2carbon-3.2.2
├── INSTALL.txt
├── LICENSE.txt
├── README.txt
├── bin
├── dbscripts
├── lib
├── release-notes.html
├── repository
└── tmp
```

Una breve explicación sobre los distintos directorios:

- **`bin/`** : Todos los binarios y archivos de código necesarios para iniciar el servidor de *Carbon.*
- **`dbscripts/`** **:** Archivos de código que crean las distintas bases de datos que *Carbon* trae por defecto.
- **`lib/`** **:** Librerias de sistema.
- **`repository/`** **:** Archivos de configuración de *Carbon*.
- **`tmp/`** **:** Archivos temporales utilizados durante la ejecución de *Carbon.*

Una vez *Carbon* esté instalado y el *JDK* esté correctamente configurado, se procede a instalar las características de *Carbon* para conseguir las funcionalidades del *IS*. Para ello, se debe de iniciar *Carbon* siguiendo estos pasos:

- Se define la variable *CARBON_HOME* para mostrar la ruta con la que estemos trabajando. Esta ruta puede variar dependiendo de donde se hayan descargado los binarios del servidor. En este caso la variable *CARBON_HOME* está configurada con:

```
export CARBON_HOME= /home/$(whoami)/Downloads/wso2carbon-3.2.2/
```

  Para iniciar *Carbon* se ejecuta el siguiente comando:

```
chmod +x $CARBON_HOME/bin/wso2server.sh && sh $CARBON_HOME/bin/wso2server.sh
```

  Y aparecerán en la consola trazas del inicio del servidor de gestión de *Carbon*:

```
[2017-09-09 03:12:41,883]  INFO {org.wso2.carbon.ui.internal.CarbonUIServiceComponent} -
 Mgt Console URL  : https://192.168.1.129:9443/carbon/
[2017-09-09 03:12:41,890]  INFO {org.wso2.carbon.core.internal.StartupFinalizerServiceCom
ponent} -  Started Transport Listener Manager
[2017-09-09 03:12:41,891]  INFO {org.wso2.carbon.core.internal.StartupFinalizerServiceCom
ponent} -  Server           :  WSO2 Carbon-3.2.2
[2017-09-09 03:12:41,891]  INFO {org.wso2.carbon.core.internal.StartupFinalizerServiceCom
ponent} -  WSO2 Carbon started in 8 sec
```

Figura 4-13 Inicio de *Carbon*

Una vez el servidor se haya iniciado correctamente, mostrará la dirección de red en la que el servidor está escuchando para poder acceder a ella y administrarlo.

Si se accede a la dirección y puerto mostrados en la consola aparecerá en el navegador web la interfaz de gestión de *Carbon,* donde se muestra un formulario para ingersar en la consola de gestión.

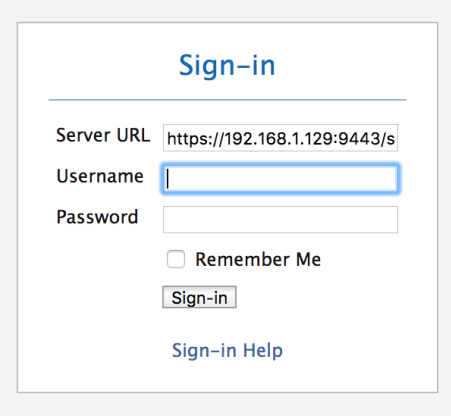


Figura 4-14 Inicio de sesión en *Carbon*

El usuario y contraseña por defecto es ***admin*** y ***admin***.

Una vez dentro de la consola de administración se utilizará el menú lateral para acceder a la configuración de las característica instaladas en el servidor:

Figura 4-15 Menú de características de *Carbon*

Una vez se acceda al menú de *Características (Features)* encontraremos la interfaz para añadir un repositorio de caracterísiticas del servidor. Este repositorio puede ser local o estar albergado en algún servidor público de Internet. En este caso se utiliza el repositorio de WSO2 de la versión 3.2.2. La dirección de este servidor es:

http://product-dist.wso2.com/p2/carbon/releases/3.2.2/

Para configurar el repositorio se accede pulsando el botón de *Añadir Repositorio (Add Repository)* y se rellena la información necesaria sobre este:

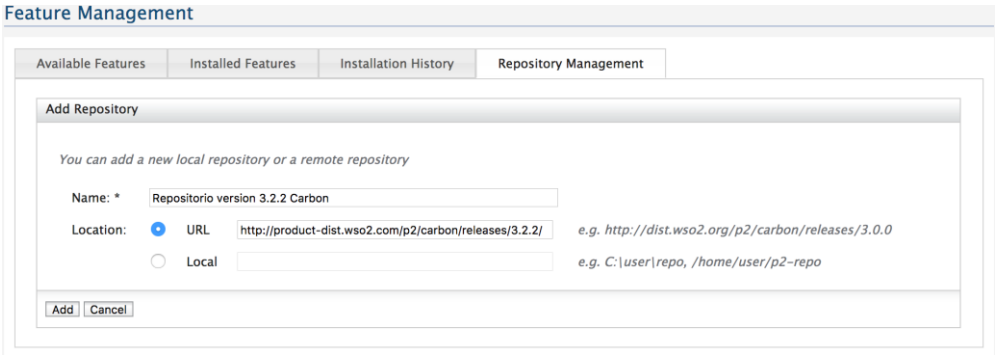


Figura 4-16 Asistente de configuración de repositorios de características de *Carbon*

Se añade el repositorio pulsando sobre el botón *Añadir (Add)*.

Una vez añadido, se puede comenzar a buscar características del servidor pulsando sobre el botón *Encontrar Características (Find features)*. Por defecto, si no se rellena ningún parámetro de búsqueda en el campo del formulario destinado a ello, se mostrarán todas las características disponibles para instalar:

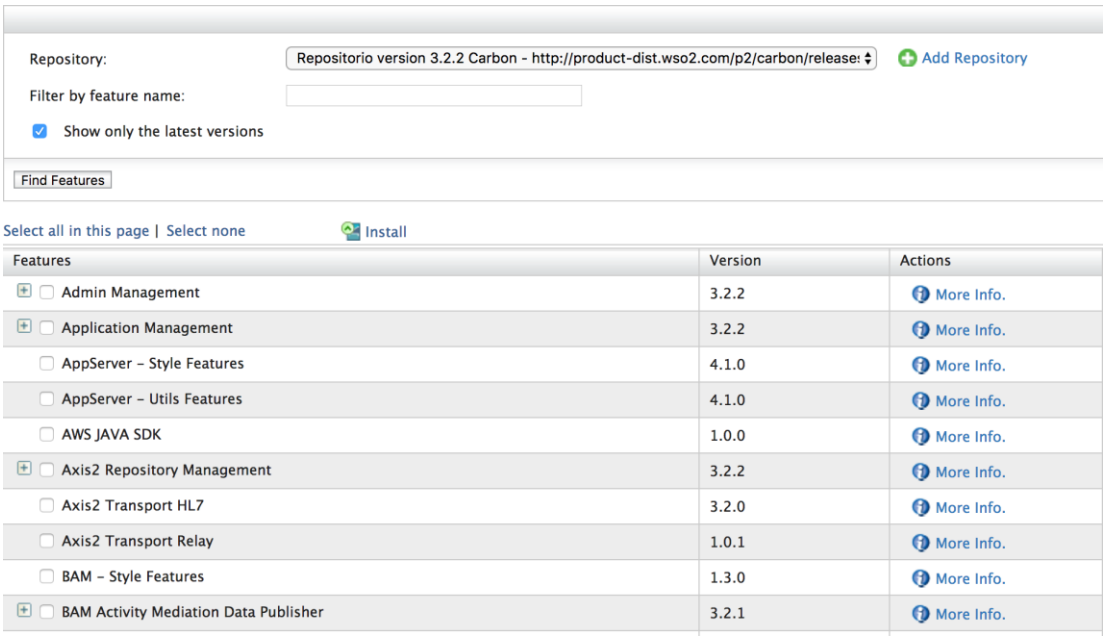


Figura 4-17 Instalación de caraterísticas de *Carbon*

Si por lo contrario no se desea configurar *Carbon* con los paquetes de características del *IS* y el *ESB*, también es posible instalar el *IS* o el *ESB* descargando la versión que lo include todo configurado por defecto. El Enlace es el siguiente:

```
http://wso2.com/more-downloads/identity-server/
http://wso2.com/more-downloads/esb/
```

La descarga contendrá un archivo de formato *.zip* en el que contiene la misma estructura de carpetas (ya que sigue siendo el servidor *Carbon*) pero configurado con todos los paquetes necesarios del IS y el ESB.

4.5. Máquina virtual IS

Basado en el apartado anterior donde se ha realizado una instalación de *Carbon*, se procede a instalar las características específicas del *IS* sobre *Carbon*.

Esta es la lista de caracterísiticas que se deben seleccionar de la lista desplegable de características disponibles de características de *Carbon* para instalar:

COMPONENTE	VERSION
Carbon	3.2.2
Claim Management	3.2.0
Event Common	3.2.0
Event Server	3.2.1
Feature - Qpid – Server	3.2.1
Governance Notifications Configuration	3.2.2
Identity Core	3.2.2
Identity Oauth	3.2.1
Identity Provider	3.2.2
Identity Relying Party	3.2.2
Identity SAML2.0 Single Sign-on	3.2.2
Identity Self Registration	3.2.0
Identity User Profiles	3.2.0
Identity XACML	3.2.2
IS - Style Features	3.2.2

IS - Utils Features	3.2.2
Logging Management	3.2.2
Mex Module	3.2.0
Passive STS	3.2.2
Profile Management Core	3.2.0
Registry Community Features	3.2.2
Registry Core	3.2.2
Registry Profiles	3.2.2
Registry Resource Properties	3.2.2
Registry UI Menu	3.2.0
Remote user management Server	3.2.0
Security Management	3.2.2
Service Principle Management	3.2.0
SOAP Tracer	3.2.0
Statistics	3.2.2
STS	3.2.2
Token based authenticator	3.2.0
WS-Discovery Core	3.2.0
WS-Discovery Management UI	3.2.0
Xfer Module	3.2.0
XKMS Management	3.2.0

Tabla 2 – Características de *Carbon* de WSO2 IS

Como se ve, no es necesario instalar los paquetes de Kerberos ya que *Carbon* los incluye por defecto a partir de la versión 3.2.2.

Una vez marcadas todas estas características se procede a la instalación pulsando el botón *Instalar (Install)*, aceptando los términos y condiciones de la instalación. Cuando el proceso de instalación haya terminado será necesario reiniciar *Carbon.* Para ello en el menú lateral hay una sección específica destinada a ello:

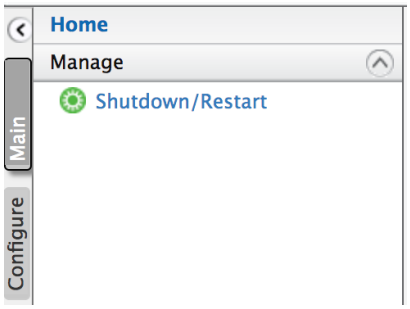


Figura 4-18 Menú de gestión de *Carbon*

Dentro de las opciones de *Shutdown/Restart* tendremos las distintas opciones para reiniciar o apagar el servidor. Para reiniciar el servidor se utilizará la opción de *Graceful Restart*, haciendo que el servidor siga computando las peticiones con las que está trabajando pero deje de aceptar peticiones nuevas. Cuando el servidor haya terminado de atender las peticiones restantes se reiniciará.

Despues del reinicio el servidor de *Carbon* tendrá la apariencia del IS, pues uno de los paquetes instalados era la interfaz gráfica del mismo.

Con esto, queda terminada la instalación del IS. El siguiente capítulo está dedicado a la configuración del mismo.

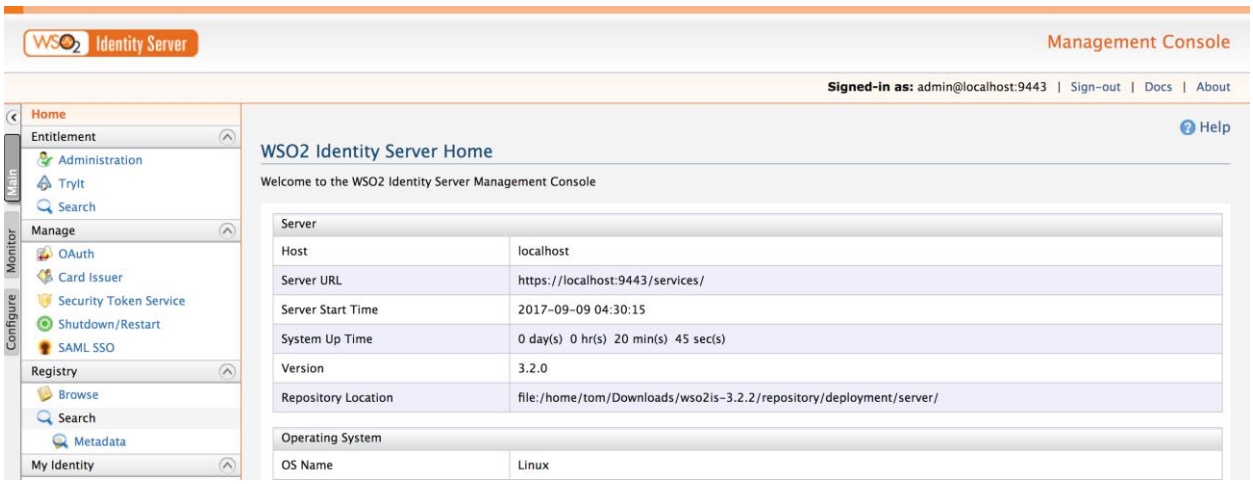


Figura 4-19 Interfaz de inicio de WSO2 IS

4.5.1. Configuración del IS

Esta sección está dedicada a la configuración del servidor de identidad. Antes de comenzar, es recomendable parar la ejecución del servidor.

La ruta que se muestra a continuación es la ruta absoluta al directorio en el que tengamos instalado el IS. Se ha utilizado una variable de entorno para hacer referencia a ella en futuros comandos, haciendo así mas simple para el lector los comandos ejecutados o la localización de los archivos. Dependiendo del nombre del usuario del sistema y de la ruta de instalación esta variable debe ser modificada:

```
export IS_HOME=/home/$(whoami)/wso2carbon-3.2.3/
```

4.5.1.1. Archivo de configuración – embebbed-ldap.xml

Este archivo se encuentra en la ruta:

```
$IS_HOME/repository/conf/embedded-ldap.xml
```

Este archivo está escrito en el lenguaje de etiquetas *XML* y almacena la configuración de la conexión entre el IS y la base de datos LDAP.

Todos los productos de WSO2 incluyen una base de datos LDAP (Lightweight Directory Access Protocol). Esta base de datos almacena los distintos usuarios y roles que serán creados posteriormente en la ejecución del caso práctico.

Para configurar la conexión entre el KDC y la base de datos LDAP es necesario buscar la sección del archivo que albega la configuración del mismo:

```
<KDCServer>
  <Property name="name">defaultKDC</Property>
  <Property name="enabled">true</Property>
  <Property name="protocol">UDP</Property>
  <Property name="host">localhost</Property>
  <Property name="port">${Ports.EmbeddedLDAP.KDCServerPort}</Property>
  <Property name="maximumTicketLifeTime">8640000</Property>
  <Property name="maximumRenewableLifeTime">604800000</Property>
  <Property name="preAuthenticationTimeStampEnabled">true</Property>
</KDCServer>
```

Figura 4-20 Activación del KDC en el IS

Es necesario modificar la propiedad ***enabled***, cambiandola a ***true***.

Esta configuración hace que el IS utilice el servidor de LDAP local (lo trae embebido) en vez de un servidor LDAP externo. Aparte, configura opciones de segurdad como la pre-autenticación con marca de tiempo, que están diseñadas para ofrecer protección contra los ataques de fuerza bruta para contraseñas.

4.5.1.2. Archivo de configuración – user-mgt.xml

Este archivo se encuentra en la ruta:

```
$IS_HOME/repository/conf/user-mgt.xml
```

Al igual que el anterior archivo se encuentra escrito en el lenguaje de etiquetas XML y se encarga de la gestión de la configuración de los usuarios del servidor.

En este caso, se han de modificar varias secciones del archivo. Al abrir el archivo, se puede ver que esta sección de código está descomentada y es la que se ejecutaria en caso de iniciar el servidor:

```
<UserStoreManager class="org.wso2.carbon.user.core.ldap.ApacheDSUserStoreManager">
    <Property name="ReadOnly">false</Property>
    <Property name="ConnectionURL">ldap://localhost:${Ports.EmbeddedLDAP.LDAPServerPort}</Property>
    <Property name="ConnectionName">uid=admin,ou=system</Property>
    <Property name="ConnectionPassword">admin</Property>
    <Property name="passwordHashMethod">SHA</Property>
    <Property name="UserNameListFilter">(objectClass=person)</Property>
    <Property name="UserEntryObjectClass">wso2Person</Property>
    <Property name="UserSearchBase">ou=Users,dc=wso2,dc=org</Property>
    <Property name="UserNameSearchFilter">(&amp;(objectClass=person)(uid=?))</Property>
    <Property name="UserNameAttribute">uid</Property>
    <Property name="PasswordJavaScriptRegEx">[\\S]{5,30}</Property>
    <Property name="UsernameJavaScriptRegEx">[\\S]{3,30}</Property>
    <Property name="UsernameJavaRegEx">^[^~!@#$;%^*+={}\\|\\\\&lt;&gt;]{3,30}$</Property>
    <Property name="RolenameJavaScriptRegEx">[\\S]{3,30}</Property>
    <Property name="RolenameJavaRegEx">^[^~!@#$;%^*+={}\\|\\\\&lt;&gt;]{3,30}$</Property>
    <Property name="ReadLDAPGroups">true</Property>
    <Property name="WriteLDAPGroups">true</Property>
    <Property name="EmptyRolesAllowed">true</Property>
    <Property name="GroupSearchBase">ou=Groups,dc=wso2,dc=org</Property>
    <Property name="GroupNameListFilter">(objectClass=groupOfNames)</Property>
    <Property name="GroupEntryObjectClass">groupOfNames</Property>
    <Property name="GroupNameSearchFilter">(&amp;(objectClass=groupOfNames)(cn=?))</Property>
    <Property name="GroupNameAttribute">cn</Property>
    <Property name="MembershipAttribute">member</Property>
    <Property name="UserRolesCacheEnabled">true</Property>
</UserStoreManager>
```

Figura 4-21 Activación del KDC en el IS (2)

Esta es la configuración por defecto de *Carbon*. Es la configuración base que se incluye con la instalacion de *Carbon*, por lo que no activará el IS ni ninguna de sus características. Ya que en este caso práctico se está utilizando el IS como KDC, es necesario comentar esta sección de código.

Para hacerlo será necesario añadir lo siguiente en las etiquetas que lo engloban:

```
<!--UserStoreManager>
        <Property>…</Property>

        ...
 <UserStoreManager-->
```

Si se continua mirando en el archivo, se observa que hay un comentario indicando lo siguiente:

```
<!-- Following user manager is used by Identity Server (IS) as its default user manager -->
```

Indicando que el trozo de código mostrado a continuación es el de la configuracion de *Carbon* junto con la del IS, por lo tanto , es necesario descomentarla:

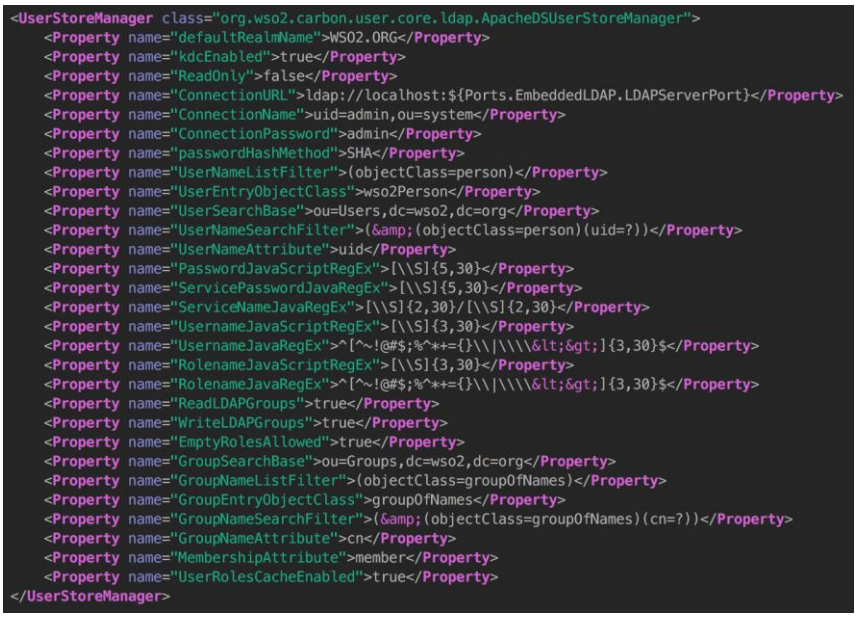


Figura 4-22 Activación del KDC en el IS (3)

Tambien será necesario modificar la propiedad **`kdcEnabled`**, cambiando su valor a **`true`** y activando así el servicio de Kerberos del IS.

4.5.1.3. Inicio del IS

Una vez se ha llegado a este punto el IS ha sido correctamente configurado. Una vez el IS se inicie, el KDC se iniciará y quedará a la espera de peticiones. Para ello, es necesario ir a la carpeta que contiene el ejecutable del IS:

```
$IS_HOME/bin/
```

Dentro de esa carpeta se encontrará el archivo de código de arranque del servidor: ***`wso2server.sh`***. Antes de ejecutarlo es necesario otorgarle permisos de ejecución para que pueda iniciarse, para ello:

```
chmod +x wso2server.sh
```

Hecho esto se puede iniciar el servidor con:

```
./wso2server.sh
```

En la consola comenzará a aparecer la información de inicio del servidor. Entre estas trazas se encuentran las mostradas en la figura 4-23, las cuales informan del inicio del servicio de Kerberos:

```
[2016-07-06 17:58:59,387]  INFO {org.apache.directory.server.kerberos.kdc.KdcServer} -
  Kerberos service started.
Kerberos service started.
```

Figura 4-23 Inicio del servicio de Kerberos en el IS

Como se observa, el servicio de Kerberos (KDC) es iniciado por el IS.

Si este mensaje no aparece, es necesario revisar las secciones anteriores y modificar los archivos necesarios.

5.5.1.4 Configuración de *principal* del servicio eco en el IS

Como se mencionó en apartados teóricos anteriores, Kerberos utiliza *principales* para identificar de manera unívoca a usuarios y servicios dentro de un dominio de Kerberos.

Estos *principales* siguen el formato:

```
{Nombre de usuario/servicio} / {instancia} / {Dominio de Kerberos}
```

Donde se configura el identificador del servicio, máquina donde ese servicio o usuario se encuentra y el dominio de Kerberos donde se encuentra ese servicio o usuario. En este apartado se procede a la creación de un *principal* para el servicio de eco, de manera que pueda ser identificado en el dominio de Kerberos.

Si se accede a la consola de gestión del IS, recordando que por defecto estará en el puerto 9443 de la máquina en la que se haya instalado, aparecerá la interfaz de usuario de la consola. En esta interfaz de gestión es necesario iniciar sesión con usuario y contraseña ***admin***, como ya se mostró en un apartado anterior. El valor por defecto de estas credenciales aparece en el archivo ***user-mgt.xml***, ya modificado.

Una vez se inicia sesión en el sistema se puede apreciar en la interfaz principal de la consola de gestión la información sobre el servidor de identidad. En el menú lateral se dispone de una sección de administración de los diferentes usuarios y distintas opciones que dispone el servidor de identidad.

El siguiente paso es crear el ***Service Principal***. En el menú lateral de la consola de gestión se accede a la opción **"Configure".** Dentro de esa opción accedemos a la de **"Kerberos KDC / Service Principals"** :

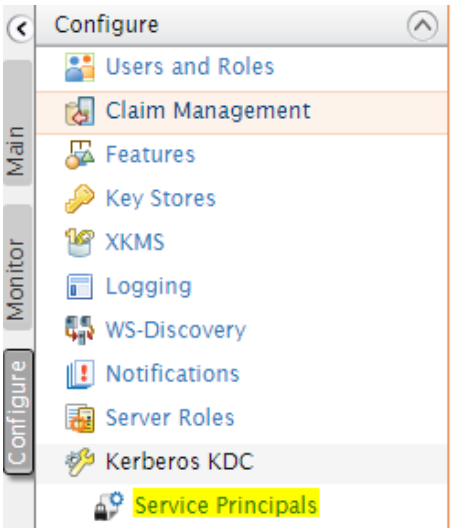


Figura 4-24 Menú lateral de configuración de Kerberos

Y se elige la opcion ***Add a new service principal***:

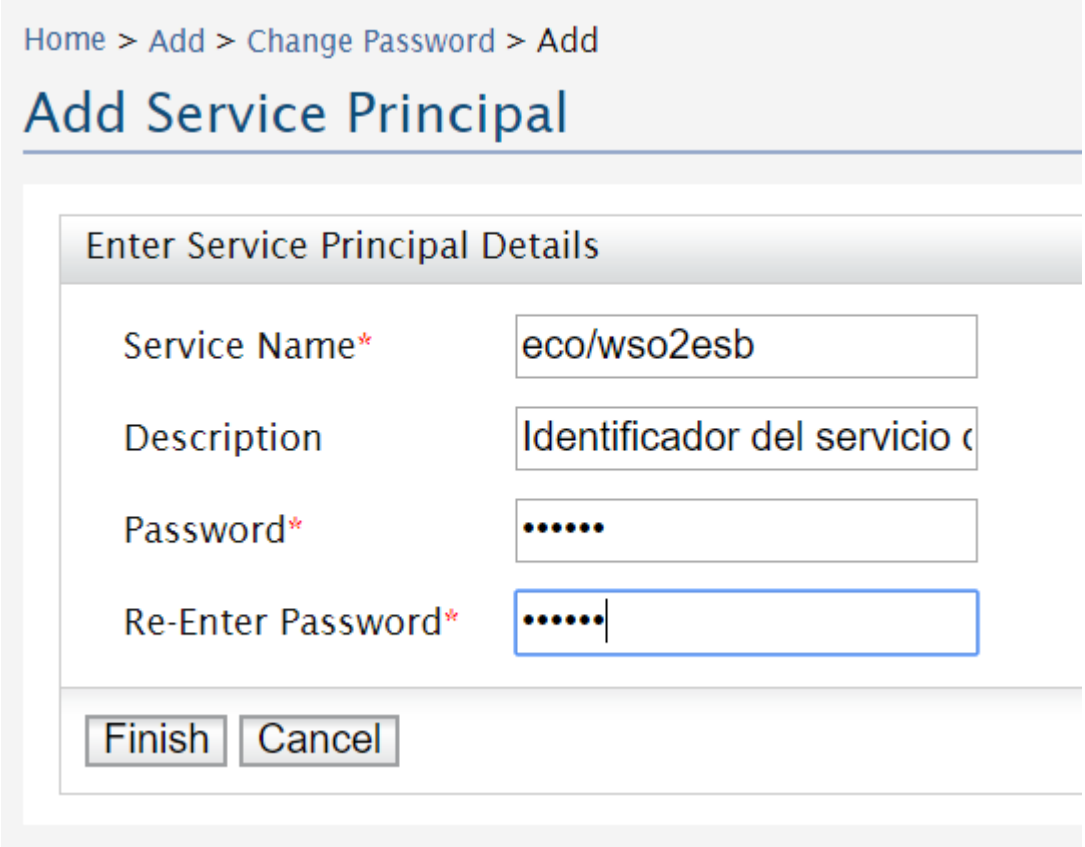


Figura 4-25 Configuración de ID de servicio de eco

WSO2 tiene por defecto configurado el dominio de Kerberos, que por defecto es WSO2.ORG, en sus archivos de configuración, por lo que no es necesario añadir el diminio de Keberos.

Como se mencionó anteriormente, es necesario configurar un id único de servicio y la máquina en la que el servicio está alojado. Dado que el servicio de eco se encuentra alojado en el ESB, esta es la configuración que se va a utilizar:

- `Service Name: eco/wso2esb`
- `Description: Service principal for ESB` (Id del servicio representando al ESB)
- `Contraseña: qazqaz`

Se ha utilizado ***wsoesb*** como alias del ESB. Este alias fue configurado anteriormente en el archivo ***/etc/hosts***.

Cuando se finalice la creación del id del servicio debería aparecer en la interfaz de los id de servicio:

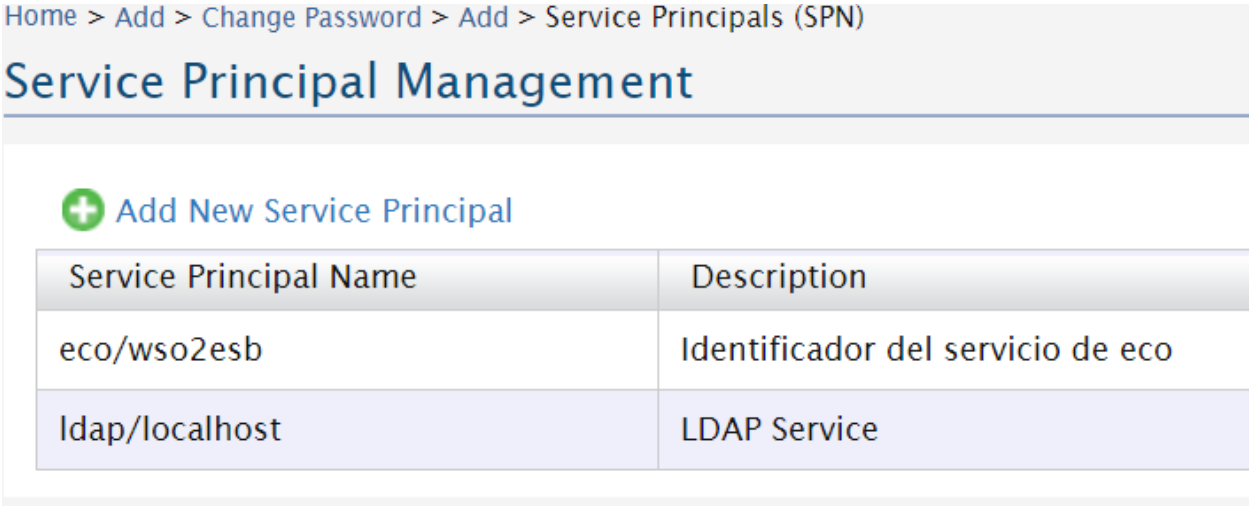


Figura 4-26 Creación de ID de servicio de eco

Una vez creado el identificador del servicio de eco es necesario crear el usuario que va a acceder a este servicio. En el esquema de la sección anterior el cliente envía al servidor de identidad sus credenciales de usuario, las cuales se crearán a continuación. Para crear este usuario se debe pulsar en ***"Users and Roles" →"Users"***:

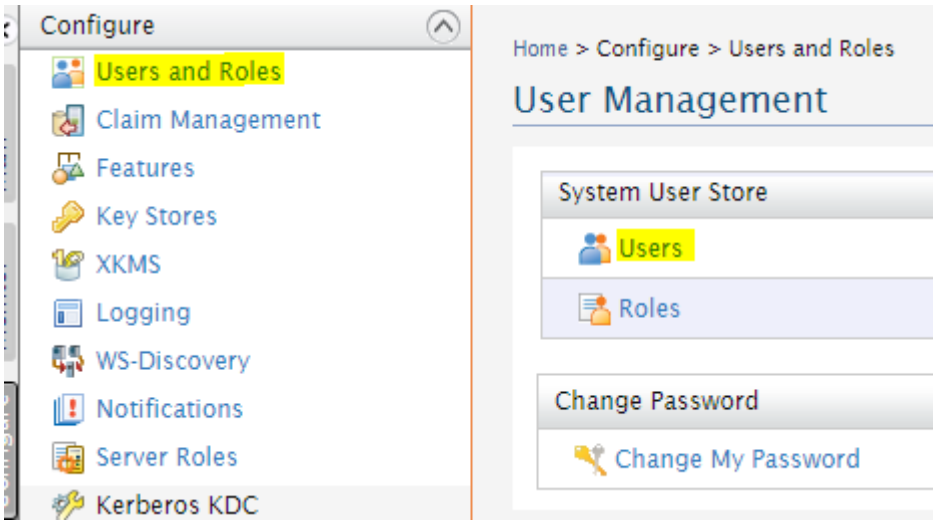


Figura 4-27 Menú de configuración de usuarios y roles

Y utilizamos la opción de crear un nuevo usuario siguiendo la interfaz web:

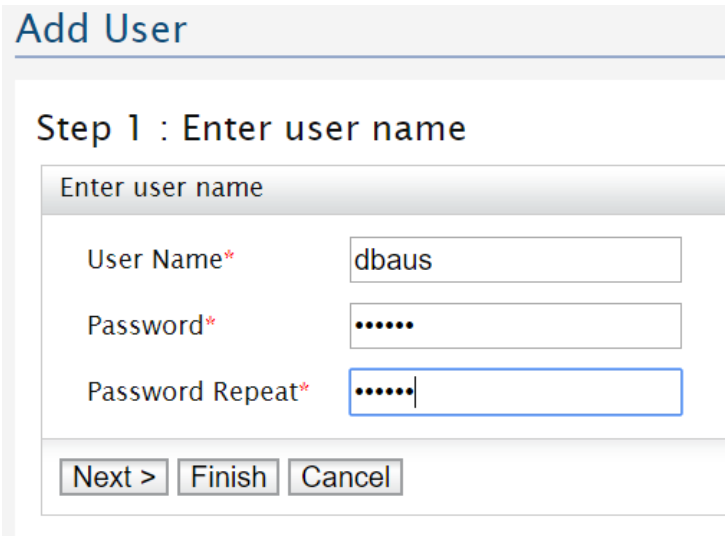


Figura 4-28 Configuración de usuario y contraseña para el servicio de eco

Los datos usados en la realización de este caso práctico son:

```
Usuario: dbaus
Contraseña: wsxwsx
```

Una vez creado este usuario aparecerá en la sección de usuarios:

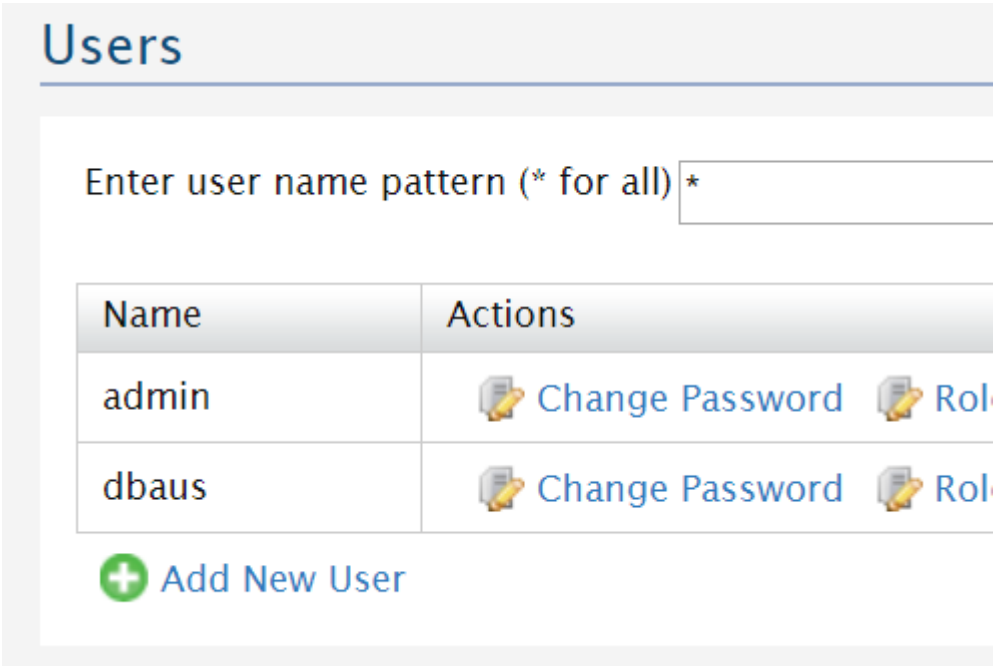


Figura 4-29 Creación de usuario para el servicio de eco

En este punto el IS estaría preparado para ser utilizado como KDC en este caso práctico. Se dispone de un service principal que se utilizará posteriormente para la autenticación con el servicio de eco y un usuario que utilizará el alumno para autenticarse contra el IS.

4.6. Máquina virtual ESB

Es importante mencionar que en esta máquina virtual es donde se encontrará alojado el servicio de eco. Este servicio es desplegado a través del ESB, por lo que es necesario configurar el ESB para después poder configurar el servicio de eco.

Ya que la instalación de *Carbon* ha sido explicada en apartados anteriores, se muestra a continuación la lista de paquetes que necesitan ser instalados en *Carbon* para configurar el ESB:

COMPONENTE	VERSION
Application Management	3.2.2
Axis2 Transport HTTP Pass-through	1.0.2
Carbon	3.2.3
CSG Agent	3.2.3
Datasource Management	3.2.3
Deployment Synchronizer	3.2.2
EC2-Client Module	3.2.0
Endpoint Management	3.2.3
ESB - Samples Features	4.0.3
ESB - Styles Features	4.0.3
Event	3.2.1
FIX Transport Core	3.2.0
JMS Transport Core	3.2.0
Local entry Management	3.2.2
Logging Management	3.2.2
Mail Transport Core	3.2.0
Mediation Initializer	3.2.3
Mediation Statistics	3.2.3
Mediation Tracer	3.2.0
Mediators	3.2.3

Message Flows	3.2.0
Message Processor Management	3.2.0
Message Relay Core	3.2.3
Message Store Management	3.2.3
Mex Module	3.2.0
NHTTP GET Request Processor	3.2.3
OAuth Mediation	3.2.1
Priority Mediation Management	3.2.2
Proxy Services	3.2.3
Registry Associations/Dependencies	3.2.3
Registry Community Features	3.2.3
Registry Core	3.2.3
Registry Resource Properties	3.2.2
Registry UI Menu	3.2.0
Reporting Core	3.2.3
Reporting UI	3.2.2
Rest API	3.2.3
Rule Drools Engine	3.2.0
Rule JSR94 Engine	3.2.0
Rule Mediation	3.2.0
Scheduled Tasks	3.2.0
Security Management	3.2.3
Sequence Tempalates Management	3.2.0
Sequences Management	3.2.2
Service Management	3.2.3

Smooks Mediator Aggregated	3.2.0
SOAP Tracer	3.2.0
Statistics	3.2.2
Stratos - Load Balance Agent UI	1.1.0
STS	3.2.2
SVN Based Deployment Synchronizer	3.2.2
Synapse	2.1.0.wso2v5
Synapse Application Deployer	3.2.0
Synapse Application Management	3.2.0
Synapse Artifact Uploader	3.2.2
Synapse Config Admin	3.2.3
Synapse FIX Transport	2.1.0.wso2v5
Synapse Nhttp Transport	2.1.0.wso2v5
Synapse VFS Transport	2.1.0.wso2v5
Synapse VFS Transport's SMB supports	2.1.0.wso2v2
Transport Management	3.2.2
Transport Statistics	3.2.0
Tryit	3.2.2
VFS Transport Management	3.2.0
WS-Discovery Core	3.2.0
WS-Discovery Management UI	3.2.0
WSDL Tools	3.2.0
XACML Mediation	3.2.2
Xfer Module	3.2.0
XKMS Management	3.2.0

Tabla 3 - Características de *Carbon* de WSO2 ESB

Una vez instaladas todas estas características es necesario seguir el mismo proceso que en el capítulo anterior y ejecutar un *Graceful Restart* para finalizar la instalación de todas las caracterísitcas de *Carbon*. Una vez reiniciado, tendremos la interfaz de ESB:

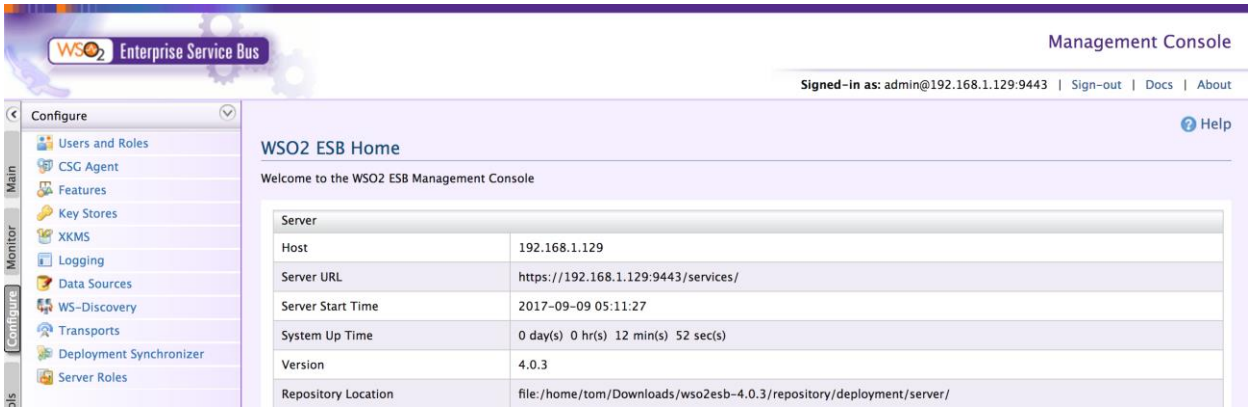


Figura 4-30 Interfaz de inicio de WSO2 ESB

4.6.1. Configuración del ESB y servicio de eco

Esta sección está dedicada a la configuración del ESB y del servicio de eco.

Al igual que en la sección dedicada al IS, para trabajar con los distintos directorios definiremos una variable de entorno llamada $ESB_HOME, la cual contiene la ruta hasta el de la instalación del ESB:

```
$ESB_HOME=/home/$(whoami)/esb/wso2carbon-3.2.3/
```

4.6.1.1. Archivo de configuración – krb5.conf

Como se va a securizar el servicio de eco con Kerberos es necesario crear un archivo de configuración de Kerberos para el ESB, ya que por defecto este servidor no tiene información del KDC. El ESB encontrará en este archivo la información sobre el servidor de tickets de Kerberos (direccion de red, puerto, dominio de Kerberos…).

Este archivo de configuración de Kerberos se ha de llamar ***krb5.conf*** y debe tener el siguiente contenido:

```
[libdefaults]
    default_realm = WSO2.ORG
    default_tkt_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1
    default_tgs_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1
    permitted_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1
    allow_weak_crypto = true
[realms]
WSO2.ORG = {
    kdc = wso2is:8000
```

```
    }
[domain_realm]
  .wso2.org = WSO2.ORG
  wso2.org = WSO2.ORG
```

Dentro de este archivo pueden diferenciarse dos bloques de configuración:

- Opciones por defecto para la librería de Kerberos: Donde se configura el reino de Kerberos por defecto que se va a utilizar, los algoritmos de encriptación para los tickets de Kerberos. Dado que en las versiones más recientes de Kerberos la lista de algoritmos ha sido marcada como insegura, es necesario incluir el parámetro ***allow_weak_crypto*** para que Kerberos la admita.
- Reinos de Kerberos: En este bloque se configura donde está el centro de distribución de claves que el reino de Kerberos ***WSO2.ORG*** va a utilizar. En este caso, contiene el alias de la dirección IP del IS.

Este archivo ha de colocarse en la siguiente ruta:

```
$ESB_HOME/repository/conf/
```

Nota: Como se ve en el código, se sigue utilizando ***is*** como la dirección de red del IS. Es necesario modificar el archivo ***/etc/hosts*** como se hizo anteriormente en la máquina virtual del ESB.

4.6.1.2. Archivo de configuración – Jaas.conf

Este archivo configura el servicio de autenticación y autorización de Java (Java Authentication and Authorization Service - JAAS) [18]. Este servicio es la implementación del módulo de autenticación de Linux, pero en Java. Es utilizado por el ESB para gestionar la autenticación de un cliente contra el servicio de eco haciendo uso del módulo de Kerberos del servicio JAAS.

Para configurar el servicio de autenticación de Java, es necesario crear un archivo llamado **jaas.conf** con el siguiente contenido:

```
Server {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=false
    storeKey=true
    useTicketCache=false
    Initiator=false
    principal="eco/wso2esb";

};
```

Esta configuración hace que el ESB no utilice una caché para guardar los tickets de Kerberos ni los *principales* de los servicios. Es por eso que el *principal* del servicio de eco se guarda en este archivo.

Y colocarlo en la ruta:

```
$ESB_HOME/repository/conf
```

Aquí es importante mencionar que el parámetro *"principal"* de este archivo debe contener el mismo identificador que se utilizó en el IS.

4.6.1.4. Inicio del ESB

Para arrancar el servidor hay que acceder a la carpeta:

```
$ESB_HOME/bin/
```

Y buscar el archivo llamado **wso2server.sh**. Al igual que para el IS es necesario aplicar permisos de ejecución ahora se debe hacer lo mismo:

```
chmod +x wso2server.sh
```

Acto seguido, se ejecuta el archivo de código encargado del inicio del servidor:

```
./wso2server.sh
```

Aparecerá por consola la información relativa al inicio del servidor. Cuando este termine el inicio aparecerá entre las trazas de ejecución el siguiente mensaje con la dirección de red de la consola de gestión:

```
[2016-07-07 12:24:35,726]  INFO - ServiceBusInitializer Starting ESB...
[2016-07-07 12:24:36,782]  INFO - CarbonUIServiceComponent Mgt Console URL  : ht
tps://192.168.1.138:9445/carbon/
[2016-07-07 12:24:36,807]  INFO - ServiceBusInitializer Initializing Apache Syna
pse...
[2016-07-07 12:24:36,816]  INFO - SynapseControllerFactory Using Synapse home :
/home/barranco/wso2esb-4.0.3/.
[2016-07-07 12:24:36,816]  INFO - SynapseControllerFactory Using synapse.xml loc
ation : /home/barranco/wso2esb-4.0.3/./../repository/deployment/server/synapse-co
nfigs/default
```

Figura 4-31 Mensajes de inicio del ESB con dirección de red de administración

4.6.1.5. Consola de administración web

Si se accede a la dirección mostrada en la información de inicio del servidor encontraremos la interfaz de la consola de administración:

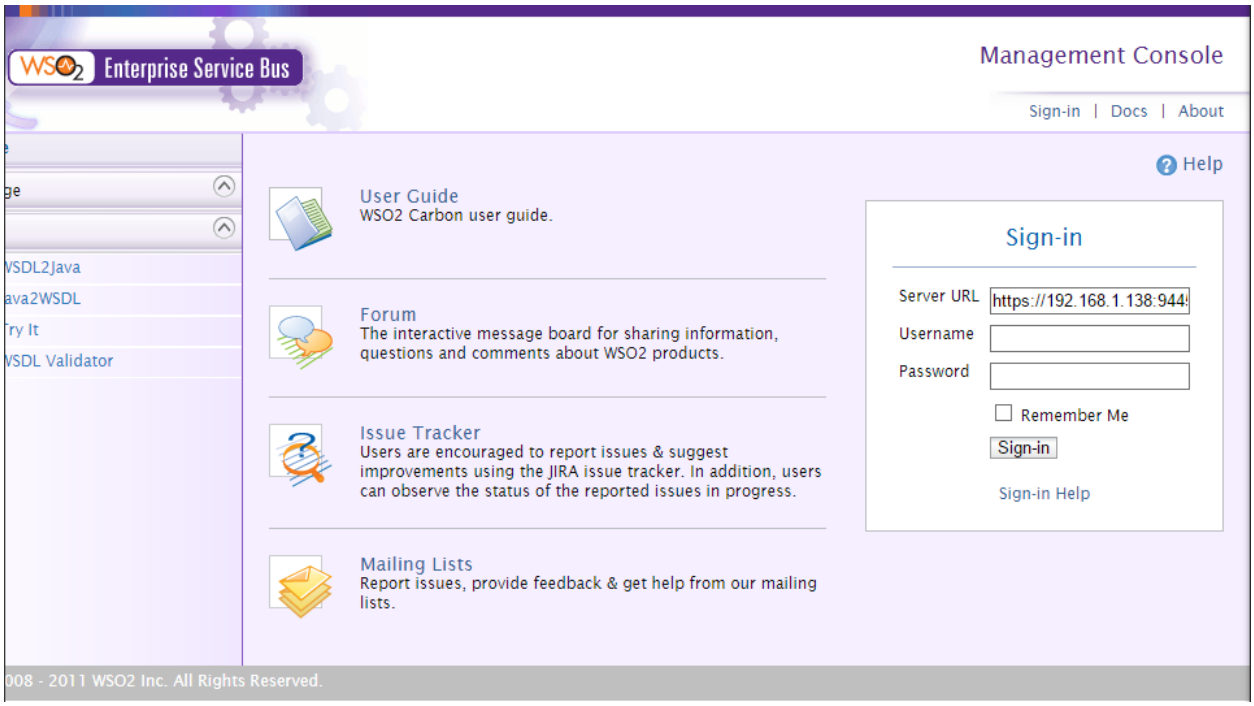


Figura 4-32 Inicio de sesión en la consola de administración web del servicio

Para iniciar sesión en el sistema utilizaremos las mismas credenciales que para el IS por defecto. Usuario y contraseña ***admin***.

La siguiente interfaz que veremos es muy parecida a la del servidor de identidad. Se muestra una tabla con la información principal del servidor y en el menú lateral encontramos:

- Una sección de monitorización donde se almacenan las estadísticas del servidor
- Una sección de configuración
- Una sección principal con las diferentes herramientas que tiene el bus de servicio empresarial
- Una sección de herramientas

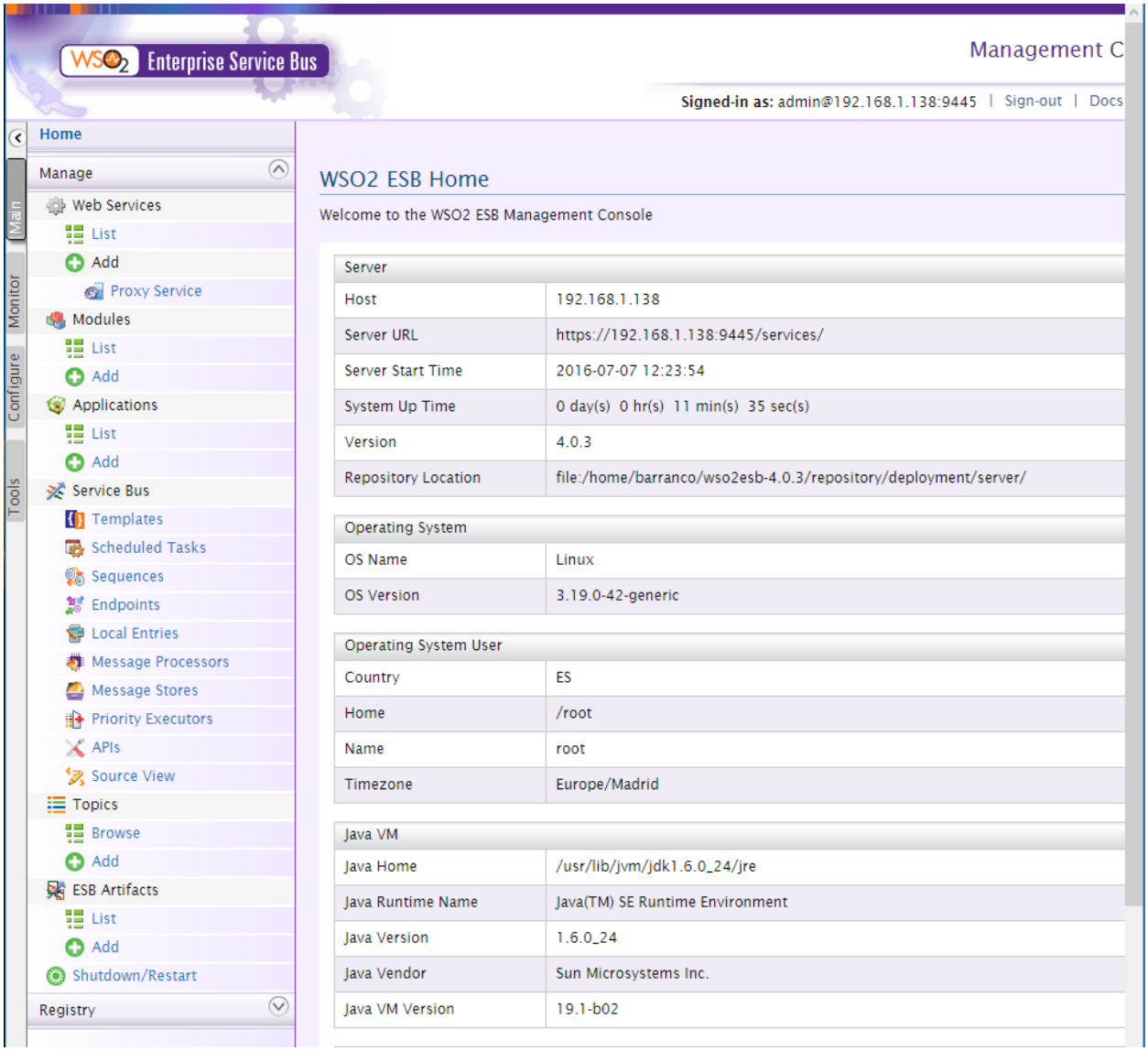


Figura 4-33 Interfaz de inicio del ESB

4.6.1.6. Activación y configuración del servicio de eco

Una vez hemos accedido a la consola de administración, se procede a iniciar el servicio de eco. Este servicio se incluye en los paquetes de características del ESB, por lo tanto, solo será necesario arrancarlo.

Para ello, en el menú lateral accedemos a la pestaña **Main** y a la sección de **Web Services.** Se pulsará sobre **List:**

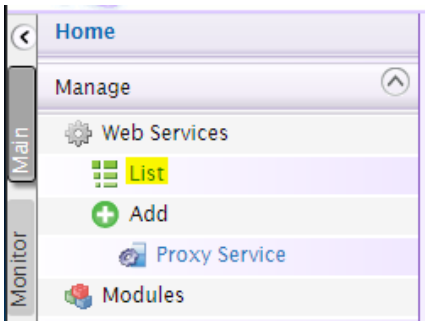


Figura 4-34 Menú lateral de configuración de servicios web

Se mostrará una lista de servicios, en el que se incluye el de eco:

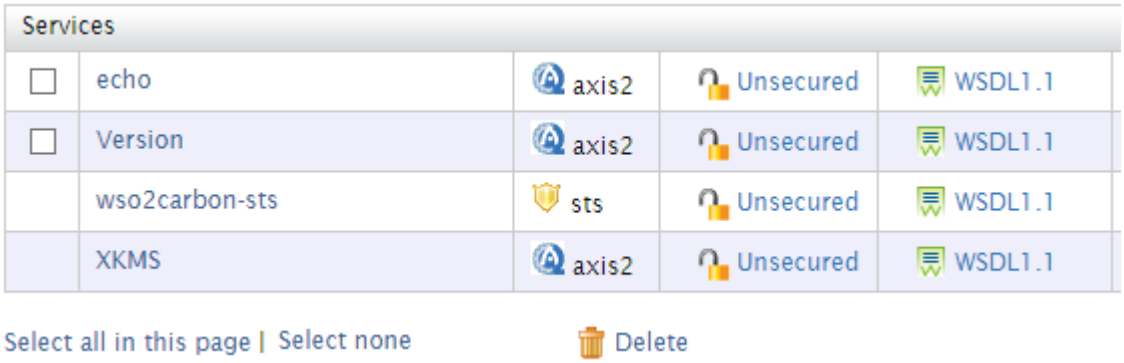


Figura 4-35 Listado de servicios web incluidos en el ESB

Al activar este servicio se expondrá una interfaz accesible desde la dirección de red de el ESB que puede ser utilizada por clientes, los cuales enviarán cadenas de texto. Al recibilas, este servicio reenvía esas cadenas de texto al cliente. Si se pulsa sobre “echo” aparecerá la interfaz gráfica de la consola de administración del servicio:

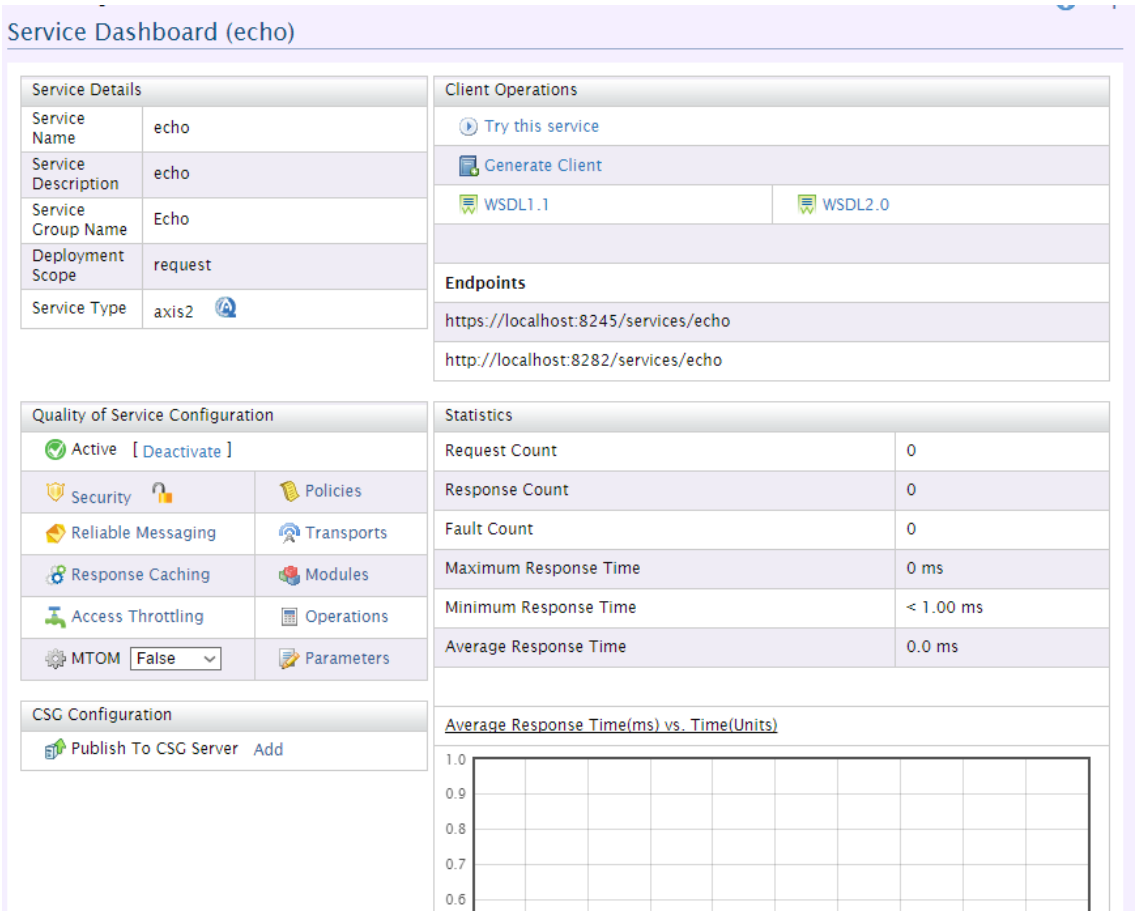


Figura 4-36 Consola de administración del servicio web de eco

En esta consola de administración se muestran estadísticas del uso del servicio, un apartado con los distintos enlaces a la configuración de este servicio y una herramienta con la que realizar pruebas contra este servicio.

Para aplicar la autenticación por Kerberos al servicio se utiliza la opción **"security"** que aparece en la imagen anterior.

Es posible elegir ahora entre una multitud de formas de autenticación para el servicio. Escogeremos la opción número dieciséis, la cual es la autenticación por Kerberos:

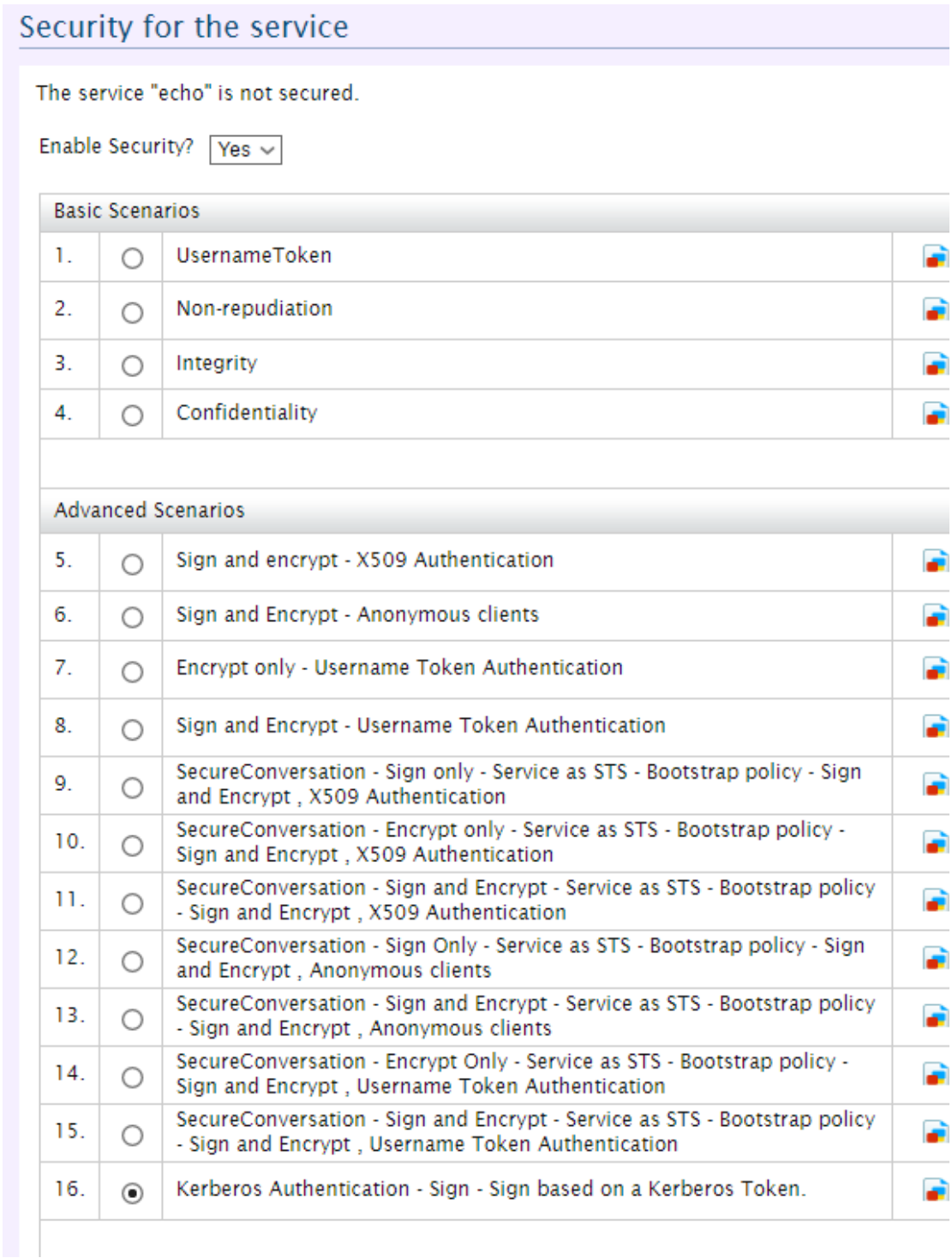


Figura 4-37 Métodos de securización del servicio de eco

Si se pulsa sobre **"Next"** se estará configurando el servicio de eco con las políticas de seguridad de Kerberos. En la siguiente pantalla aparecerá un formulario para introducir el id del servicio, el mismo que se creó en el servidor de identidad.

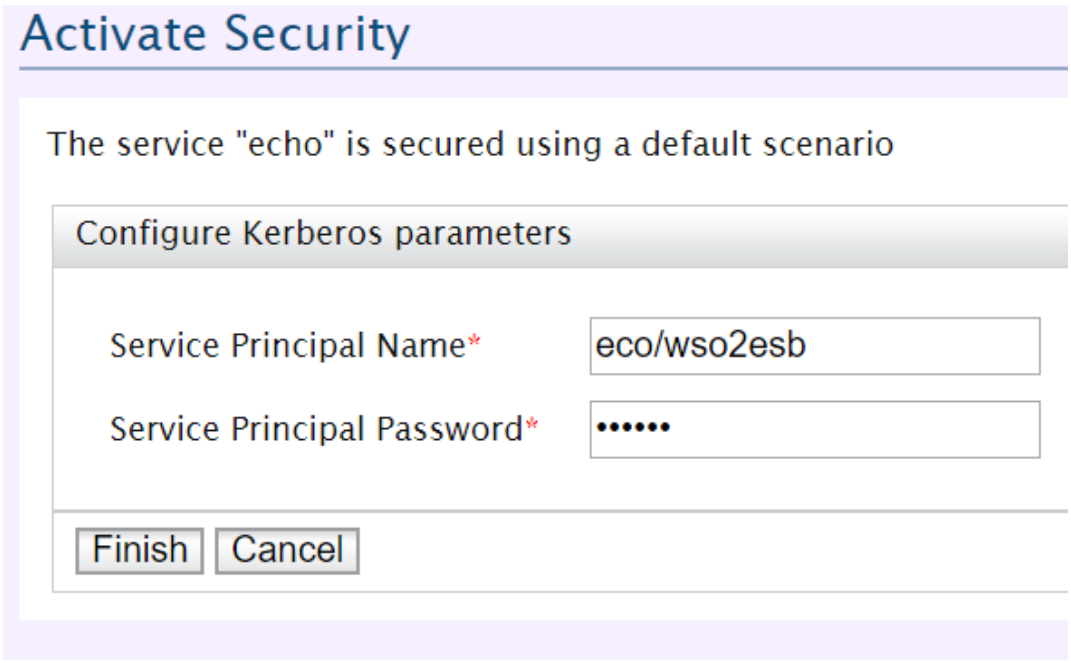


Figura 4-38 Configuración del ID del servicio de eco

Es importante recalcar que el identificador del servicio de eco debe coincidir con el creado en el IS en secciones anteriores.

4.7. Máquina virtual del cliente

Previa a la instalación del *software* del cliente, es necesario instalar la siguiente herramienta en la máquina virtual:

- **Apache Ant**:

  Ant es una herramienta de automatización de procesos de compilación, similar a ***Make*** pero diseñada para Java. Su configuración está basada en archivos de formato **XML**.

  El principal uso de esta herramienta es la compilación y ejecución del cliente que accede al servicio de eco.

  Ant se puede encontrar en los repositorios de ubuntu, por lo que es posible instalarla con el siguiente comando:

```
apt-get install ant
```

  Una vez el comando finalice su ejecución dispondremos de la herramienta Ant. Su uso se detalla en los siguientes apartados.

El cliente que accede al servicio no necesita instalación, la configuración se detalla en los siguientes apartados. Lo único que se debe hacer es decargar el cliente y descomprimir el cliente en el directorio raiz

Este cliente se incluye en los archivos entregados junto al trabajo y una copia del código fuente puede encontrarse en el anexo A. Para descomprimirlo:

```
unzip cliente_kerberos.zip
```

4.7.1. Cliente del servicio eco

Esta sección está dedicada a detallar el funcionamiento del cliente programado para este caso práctico, el cual autentica un usuario mediante el protocolo de Keberos haciendo uso del IS como centro de distribución de claves de Kerberos (KDC) y el ESB como servidor donde se aloja el servicio de eco.

Antes de comenzar a comentar cada uno de los archivos que forman la aplicación del cliente, se muestra la estructura de directorios:

```
clienteKerberos
├── build.xml
├── repo
│   └── conf
│       ├── client.axis2.xml
│       ├── jaas.conf
│       ├── krb.conf
│       └── policy.xml
├── src
│   └── org
│       └── wso2
│           └── identity
│               └── esb
│                   └── kerberos
│                       └── clienteKerberos.java
└── wso2lib
```

A continuación se muestra una breve descripción de cada archivo, ya que en las siguientes secciones se incluye una descripción más detallada de los mismos:

- ***Build.xml***: Este archivo contiene la configuración de Ant para que el cliente pueda ser compilado automáticamente.
- ***Client.axis2.xml****:* Este archivo de configuración define las clases y los protocolos que se utilizan para la comunicación del cliente contra el ESB y el IS. Este archivo es provisto por WSO2.
- ***Jaas.conf*** : Al igual que el ESB, el cliente tambien utiliza el módulo de Kerberos del sistema de autenticación y autorización de Java (JAAS). En este archivo se realiza una configuración básica del módulo de Kerberos.
- ***Krb.conf*** : Este archivo contiene la configuración de Kerberos para el cliente.
- ***Policy.xml***: Este archivo contiene la configuración del cliente que enviará al ESB cuando intente conectarse al servicio de eco. Esta configuración incluye detalles como el *principal* que el cliente quiere utilizar, datos de inicio de sesión del cliente…
- ***clienteKerberos.java***: Este archivo es el que contiene la lógica del cliente que autentica a un usuario contra el servicio de eco utilizando el protocolo de Kerberos y envía una cadena.
- **Carpeta *wso2lib***: Esta carpeta contiene las librerías de WSO2 necesarias para la compilación del cliente.

4.7.1.1. Archivo de configuración de cliente – build.xml

Este archivo es utilizado por la herramienta Ant. En él se pueden configurar los aspectos de compilación del cliente, por lo que la compilación y ejecución resulta mucho más sencilla:

```
<path id="classpath">
    <fileset dir="${carbon.home}/lib" includes="**/*.jar"/>
</path>

<property name="output.jar" value="KerberosClient.jar"/>

<target name="init">
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
</target>

<presetdef name="javac">
    <javac includeantruntime="false" />
</presetdef>

<target name="compile" depends="init"
        description="compila los archivos fuente">
    <javac srcdir="${src}" destdir="${build}" classpathref="classpath" debug="on">
    </javac>
</target>
```

Figura 4-39 – Contenido del archivo de configuración de Ant

Este archivo ya viene configurado por defecto, por lo que no es necesario realizar ninguna configuración sobre el.

4.7.1.2. Archivo de configuración de cliente – krb5.conf

Será necesario de nuevo realizar la configuración de Kerberos. Se creará otro archivo de configuración en la siguiente ruta:

```
    /home/$(whoami)/clienteKerberos/org.wso2.identity.esb.kerberos/repo/conf
```

El contenido de este archivo debe ser el siguiente:

```
[libdefaults]
    default_realm = WSO2.ORG
    default_tkt_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1
    default_tgs_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1
    permitted_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1
    allow_weak_crypto = true
[realms]
WSO2.ORG = {
    kdc = wso2is:8000
}
```

Dentro de este archivo pueden diferenciarse dos bloques de configuración:

- Opciones por defecto para la librería de Kerberos: Donde se configura el reino de Kerberos por defecto que se va a utilizar, los algoritmos de encriptación para los tickets de Kerberos. Dado que en las versiones más recientes de Kerberos la lista de algoritmos ha sido marcada como insegura, es necesario incluir el parámetro ***allow_weak_crypto*** para que Kerberos la admita.
- Reinos de Kerberos: En este bloque se configura donde está el centro de distribución de claves que el reino de Kerberos ***WSO2.ORG*** va a utilizar. En este caso, contiene el alias de la dirección IP del IS.

4.7.1.3. Archivo de configuración de cliente – Jaas.conf

Este archivo configura el servicio de autenticación y autorización de Java (Java Authentication and Authorization Service - JAAS) [18]. Este servicio es la implementación del módulo de autenticación de Linux, pero en Java.

Este archivo es utilizado por el cliente para la configuración de algunos parámetros del módulo de Kerberos. Para configurar el servicio de autenticación de Java, es necesario crear un archivo llamado **jaas.conf** con el siguiente contenido:

```
Client { com.sun.security.auth.module.Krb5LoginModule required
    useTicketCache=false;
};
```

Esta configuración se añade para que el cliente no utilice ninguna caché para almacenar los tickets de Kerberos, de tal forma de que en cada ejecución del caso práctico se realice la autenticación del usuario que accede al servicio mediante Kerberos.

La ruta en la que debe colocarse este archivo es:

```
clienteKerberos/org.wso2.identity.esb.kerberos/repo/conf/Jaas.conf
```

4.7.1.4. Archivo de configuración de cliente – policy.xml

Este archivo contiene las políticas que configuran el cliente que accede al servicio de eco. Si se observa el archivo se puede ver que este contiene los esquemas de configuración que utiliza el servicio de eco sobre políticas de seguridad:

```
</sp:SymmetricBinding>
<sp:SignedParts
        xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <sp:Body/>
</sp:SignedParts>
<sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <sp:Policy>
        <sp:MustSupportRefKeyIdentifier/>
        <sp:MustSupportRefIssuerSerial/>
        <sp:MustSupportRefThumbprint/>
        <sp:RequireSignatureConfirmation/>
    </sp:Policy>
</sp:Wss11>
<sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <wsp:Policy>
        <sp:RequireClientEntropy/>
        <sp:RequireServerEntropy/>
        <sp:MustSupportIssuedTokens/>
    </wsp:Policy>
</sp:Trust10>
```

Figura 4-40 – Configuración de políticas de seguridad para el cliente del servicio eco

Y la configuración de Kerberos:

```
            <sp:MustSupportIssuedTokens/>
        </wsp:Policy>
    </sp:Trust10>
    <rampart:RampartConfig xmlns:rampart="http://ws.apache.org/rampart/policy">
        <rampart:timestampPrecisionInMilliseconds>true
        </rampart:timestampPrecisionInMilliseconds>
        <rampart:timestampTTL>300</rampart:timestampTTL>
        <rampart:timestampMaxSkew>300</rampart:timestampMaxSkew>
        <rampart:kerberosConfig>
            <!-- Authenticating user -->
            <rampart:property name="client.principal.name">dbaus</rampart:property>
            <!-- Authenticating user password -->
            <rampart:property name="client.principal.password">wsxwsx</rampart:property>
            <!-- To which service client needs to talk to -->
            <rampart:property name="service.principal.name">eco/wso2esb@WSO2.ORG</rampart:property>
            <!-- Kerberos jaas.conf -->
            <rampart:property name="java.security.auth.login.config">repo/conf/jaas.conf</rampart:property>
            <rampart:property name="javax.security.auth.useSubjectCredsOnly">true</rampart:property>
            <!-- Kerberos configurations -->
            <rampart:property name="java.security.krb5.conf">
                repo/conf/krb.conf
            </rampart:property>
        </rampart:kerberosConfig>
    </rampart:RampartConfig>
```

Figura 4-41- Configuración de Kerberos para el cliente del servicio de eco

Donde se incluye un enlace al archivo de configuración de Kerberos, el *principal* al que quiere acceder el usuario junto a los datos de inicio de sesión de usuario.

Este archivo es necesario modificarlo para incluir los datos de inicio de sesión y el *principal* que se utilizan en el caso práctico.

4.7.1.5. Código Java del cliente

Este es el archivo principal que contiene el código Java que gestiona el acceso al servicio de eco. Este hace uso de las librerías de WSO2 de forma que la autenticación de Kerberos es invisible para el desarrollador.

Este archivo de código se puede dividir en 2 partes:

- Configuración del contexto, donde se carga el archivo de configuración de WSO2 que define que versiones de protocolos se van a utilizar para la comuniación con el servicio de eco y la carga de las políticas del cliente, las cuales se enviarán junto con el contenido de la petición al servicio.

```
// Crea el contexto de configuracion utilizando el archivo client.axis2.xml
confContext = ConfigurationContextFactory.createConfigurationContextFromFileSystem("repo",
        "repo/conf/client.axis2.xml");

servicePolicy = loadServicePolicy("repo/conf/policy.xml");


run();

System.exit(0);
```

Figura 4-42 – Configuración del contexto del cliente del servicio de eco

- El bucle de código principal, donde se utiliza el contexto que se ha cargado anteriormente y se le pregunta al usuario por dos parámetros:
    - La URL donde el servicio de eco ha sido desplegado
    - El mensaje que se desea enviar al servicio de eco

```
Scanner reader = new Scanner(System.in);
System.out.println("Introduce la URL del servicio de eco: ");
System.out.println("(Ejemplo: http://wso2esb:8280/services/echo)");
String URL_SERVICIO_ECO = reader.nextLine();


client = new ServiceClient(confContext, null);

Options options = new Options();

// Configura el servicio al que se quiere acceder
options.setAction("urn:echoString");

// Configura el endpoint
options.setTo(new EndpointReference(URL_SERVICIO_ECO));
// Configura las politicas del servicio
options.setProperty(RampartMessageData.KEY_RAMPART_POLICY, servicePolicy);

client.setOptions(options);

// Configura el modulo Rampart de Apache
client.engageModule("rampart");

// Obten la cadena que se le va a enviar al servicio de eco por parametros
System.out.println("Introduce el texto a enviar al servicio de eco: ");
String cadena_servicio_eco = reader.nextLine();
reader.close();

OMElement response = client.sendReceive(getPayload(cadena_servicio_eco));
```

Figura 4-43 Bucle principal de código del cliente del servicio de eco

Como se puede ver, en ningún momento se configuran los detalles de la autenticación del cliente contra Kerberos, eso es tarea de las librerías de la aplicación, las cuales al leer el archivo de políticas del cliente comprobarán que se autentica mediante Kerberos.

4.7.2. Ejecución

Una vez configurado el archivo de políticas con la información requirida y teniendo en cuenta de que el IS y el ESB necesitan estar ejecutándose, se puede ejecutar el cliente. Para ello, hay que navegar hacia el directorio donde se haya descomprimido el cliente (en el caso que se muestra a continuación ha sido sobre el directorio raiz):

```
cd /home/$(whoami)/clienteKerberos/org.wso2.identity.esb.kerberos/
```

Y ejecutar el siguiente comando:

```
ant run
```

La compilación y ejecución del cliente está automatizada, por lo que:

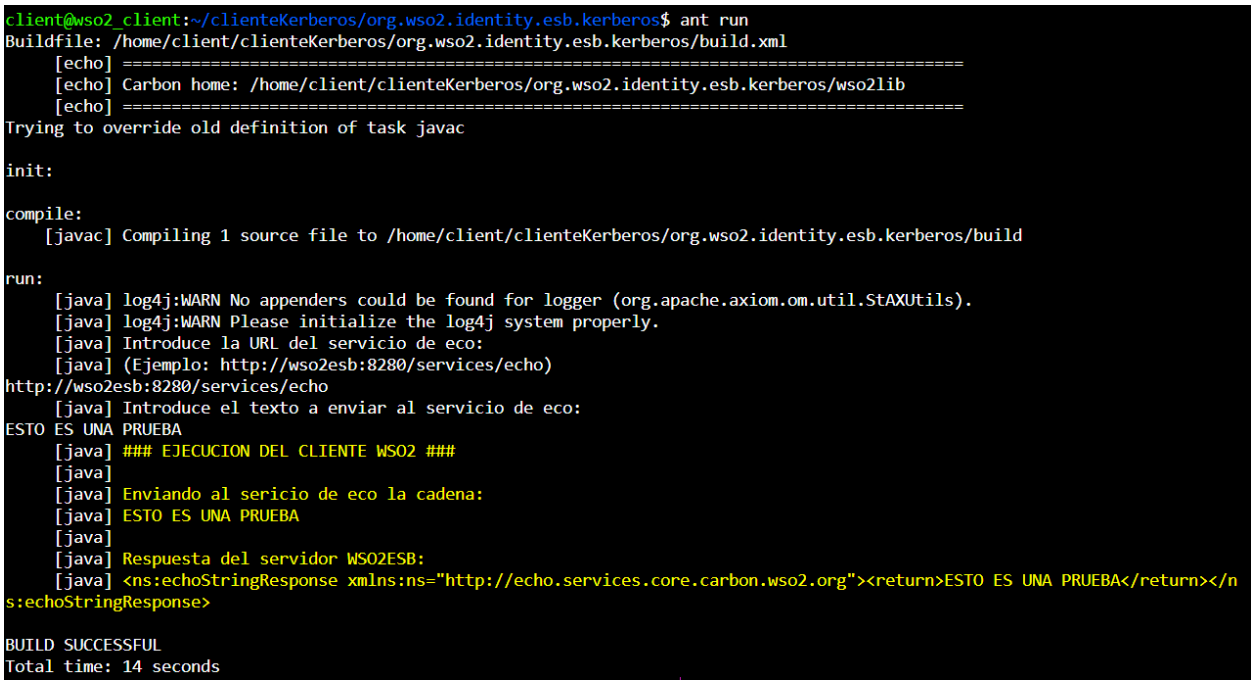


Figura 4-44 Traza de ejecución del cliente

Como se aprecia en la captura de pantalla, durante la ejecución se le solicita al usuario la URL del servicio de eco, la cual puede ser obtenida observando el panel de administración del servicio en el ESB, y el texto que se quiere enviar al servicio de eco.

Si durante la ejecución de este caso práctico alguno de los parámetros no ha sido configurado correctamente, como por ejemplo, diferencias en el identificador del servicio de eco en los distintos archivos, se obtendrá una traza de ejecución del cliente similar a:

```
[java] GSSException: No valid credentials provided (Mechanism level: Server not found in Kerberos database (7) - Server not found in Kerberos database)
     [java]     at java.security.jgss/sun.security.jgss.krb5.Krb5Context.initSecContext(Krb5Context.java:771)
     [java]     at java.security.jgss/sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:265)
     [java]     at java.security.jgss/sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:196)
     [java]     at org.apache.ws.security.message.WSSecKerberosToken$1.run(WSSecKerberosToken.java:131)
     [java]     at org.apache.ws.security.message.WSSecKerberosToken$1.run(WSSecKerberosToken.java:119)
     [java]     at java.base/java.security.AccessController.doPrivileged(Native Method)
     [java]     at java.base/javax.security.auth.Subject.doAs(Subject.java:361)
     [java]                                                                                                 at org.apache.ws.security.message.WSSecKerberosToken.getServiceTicketData(WSSecKerberosToken.java:119)
     [java]     at org.apache.ws.security.message.WSSecKerberosToken.prepare(WSSecKerberosToken.java:193)
     [java]     at org.apache.ws.security.message.WSSecKerberosToken.build(WSSecKerberosToken.java:91)
```

Una vez el caso práctico ha sido ejecutado, se pueden ver estadísticos básicos del servicio accediendo al pandel de adminstración del servicio de eco en el ESB:

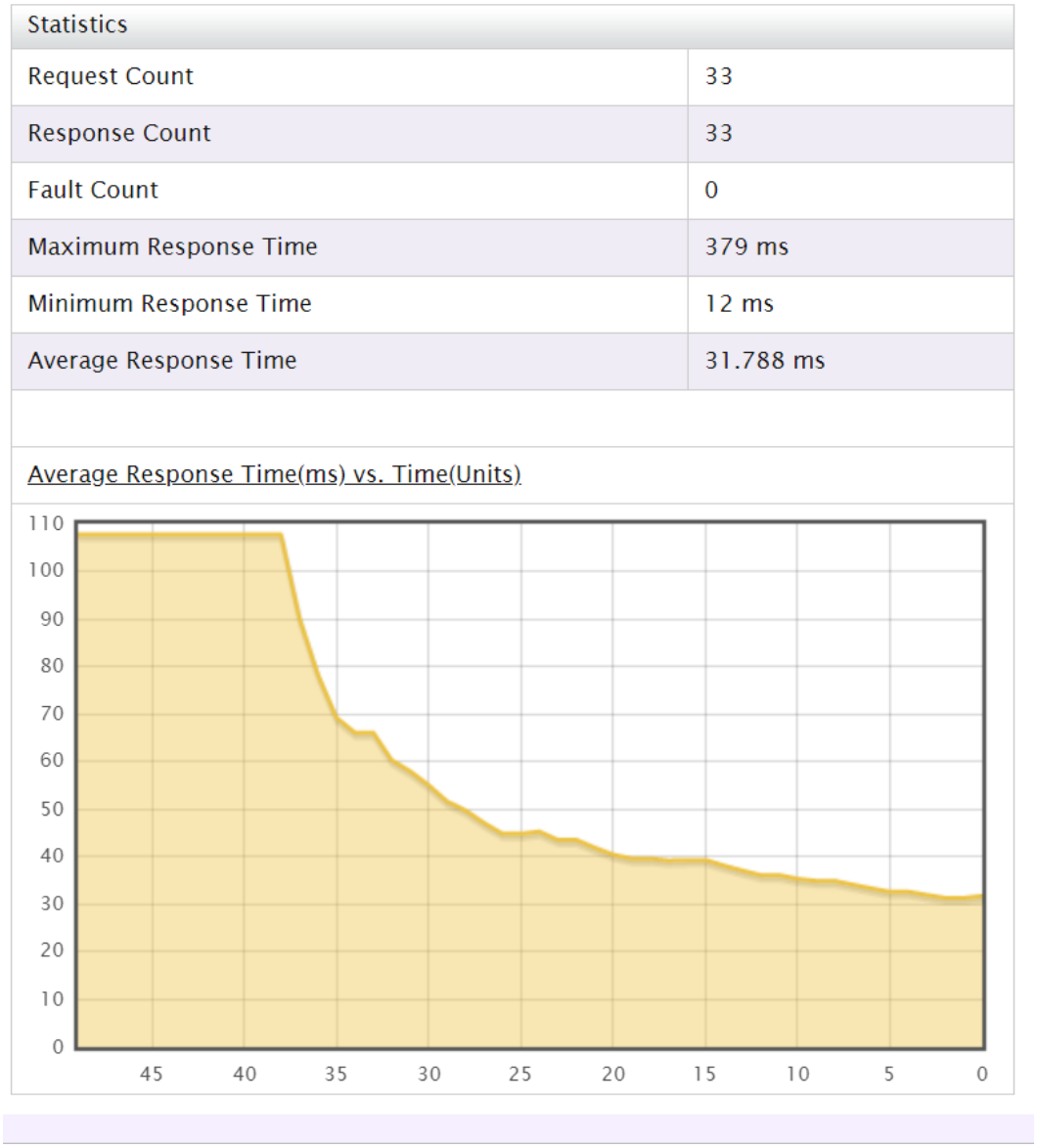


Figura 4-45 Métricas del tiempo de respuesta del servicio de eco

En la imagen se aprecia como se ha hecho uso del servicio varias veces, los tiempos de respuesta máximos y mínimos y una gráfica de los tiempos de respuesta. Se puede ver como tras cierto tiempo de uso del servicio el tiempo medio de respuesta está sobre los 32 milisegundos.

4.8. Análisis trazas de ejecución del cliente

Es posible obtener más trazas de ejecución del cliente en el lado del IS y del ESB, donde se puede obtener información sobre los tickets de Kerberos solicitados y las trazas de la conexión con el servidor de eco. Por defecto en los servidores IS y ESB el nivel de verbosidad de las trazas del sistema está configurado a *INFO*, el cual no muestra información sobre los tickets de Kerberos utilizados.

Para cambiar el nivel de verbosidad, hay que:

- Conectarse a la consola de administración del sistema y acceder a la sección de *logging* dentro de la sección de configuración:

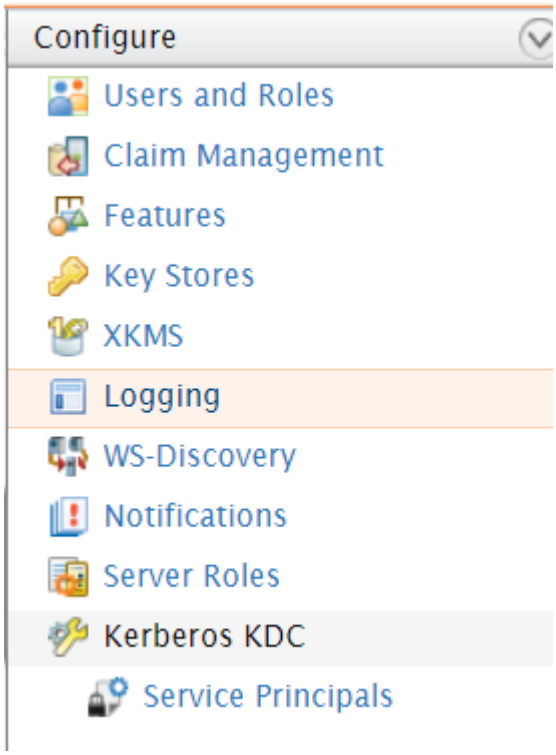


Figura 4-46 – Acceso a la configuración de trazas de sistema del IS

- Dentro de esta sección, se cambia el parámetro de configuración de *INFO* a *DEBUG*:

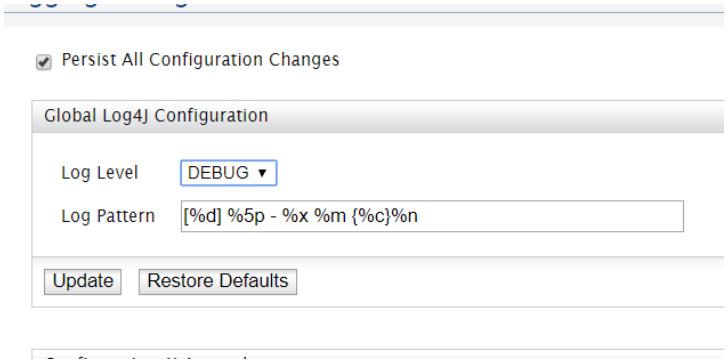


Figura 4-47 – Cambio de configuración de trazas de sistema en el IS

Una vez hecho, se utiliza la opción *update* para guardar este cambio.

Para observar los tickets de Kerberos en las trazas del sistema del IS o el ESB, hay que acceder a la sección de trazas del sistema, dentro de la sección de monitorización:

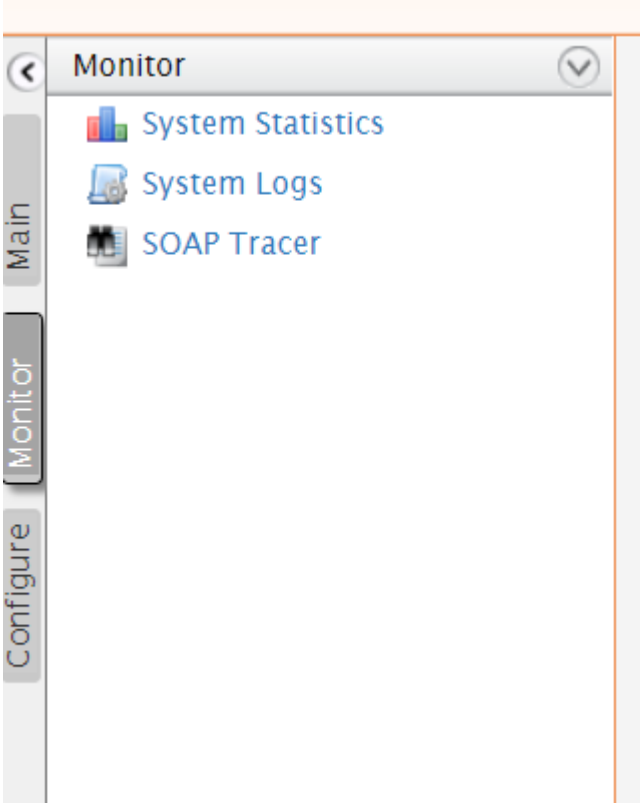


Figura 4-48 – Acceso a la sección de monitorización del IS

Dentro de esta sección es posible descargar el archivo de trazas llamado *wso2carbon.log*, el cual contiene las trazas de ejecución del cliente.

Si una vez descargado este archivo se examina el contenido del mismo, se encuentra las trazas del servidor de autenticación (es una parte del KDC de Kerberos):

```
[2019-06-30 16:24:41,962] DEBUG -  Received Authentication Service (AS) request:
    messageType:           AS_REQ
    protocolVersionNumber: 5
    clientAddress:         192.168.1.6
    nonce:                 1932923517
    kdcOptions:
    clientPrincipal:       dbaus@WSO2.ORG
    serverPrincipal:       krbtgt/WSO2.ORG@WSO2.ORG
    encryptionType:        des3-cbc-sha1-kd (16), des-cbc-crc (1), des-cbc-md5 (3)
    realm:                 WSO2.ORG
    from time:             null
    till time:             19700101000000Z
    renew-till time:       null
    hostAddresses:         null {org.apache.directory.server.kerberos.kdc.authentication.AuthenticationService}
[2019-06-30 16:24:41,962] DEBUG -  Received Authentication Service (AS) request:
```

Petición de autenticación del cliente que accede al servicio de eco. Aquí se incluye la información que ha sido configurada en los archivos de configuración de Kerberos, como algoritmos, versión de protocolo de Kerberos…

Si se sigue examinando el archivo se puede ver las trazas de la búsqueda del IS en la base de datos LDAP de información del usuario que accede al servicio de eco:

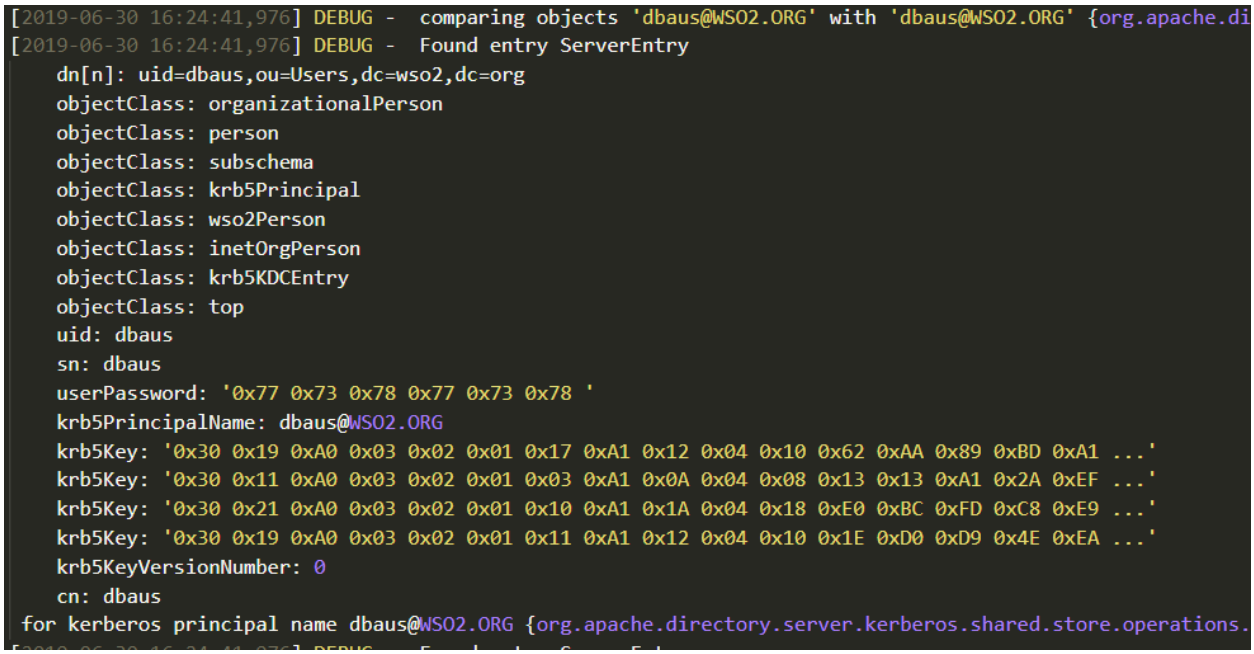


Figura 4-49 – Búsqueda de usuario en base de datos LDAP

Así como la petición de un TGS (ticket de servicio) para el servicio de eco:

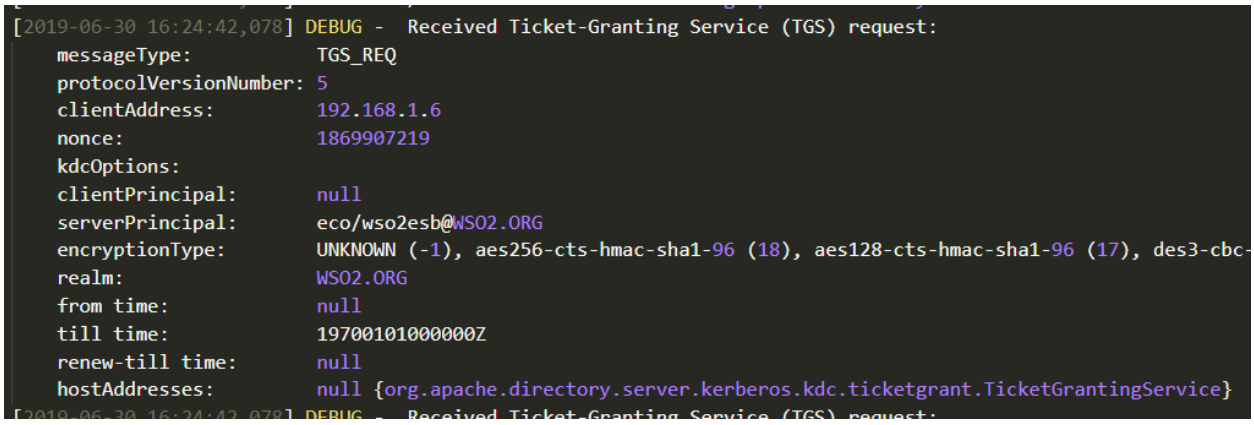


Figura 4-50 – Petición de ticket de servicio para el servicio de eco

Y su respuesta:

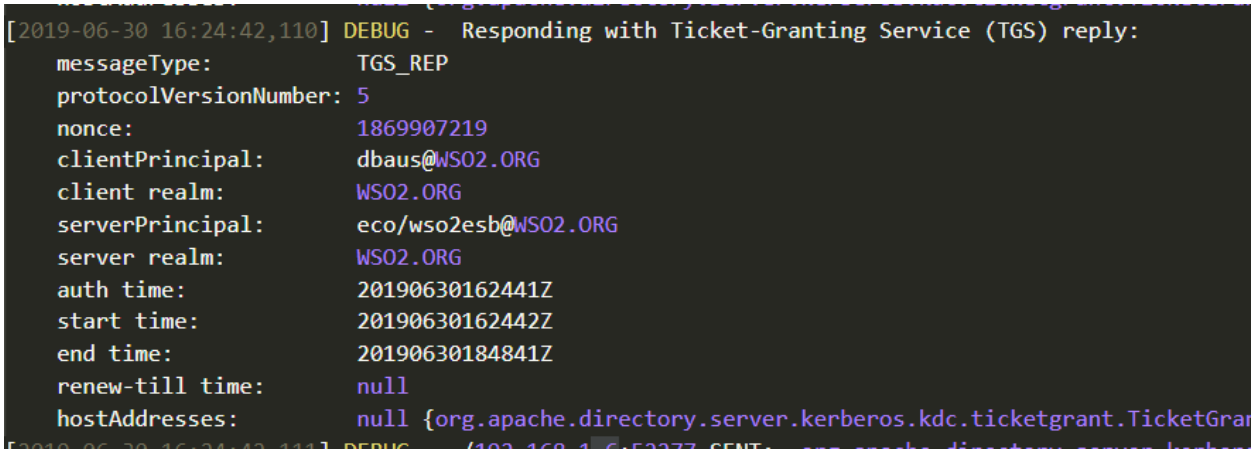


Figura 4-51 – Respuesta de creación de TGS para el servicio de eco

Si se realiza la misma operación en el ESB, es posible encontrar las trazas de ejecución del cliente cuando

accede al servicio de eco:

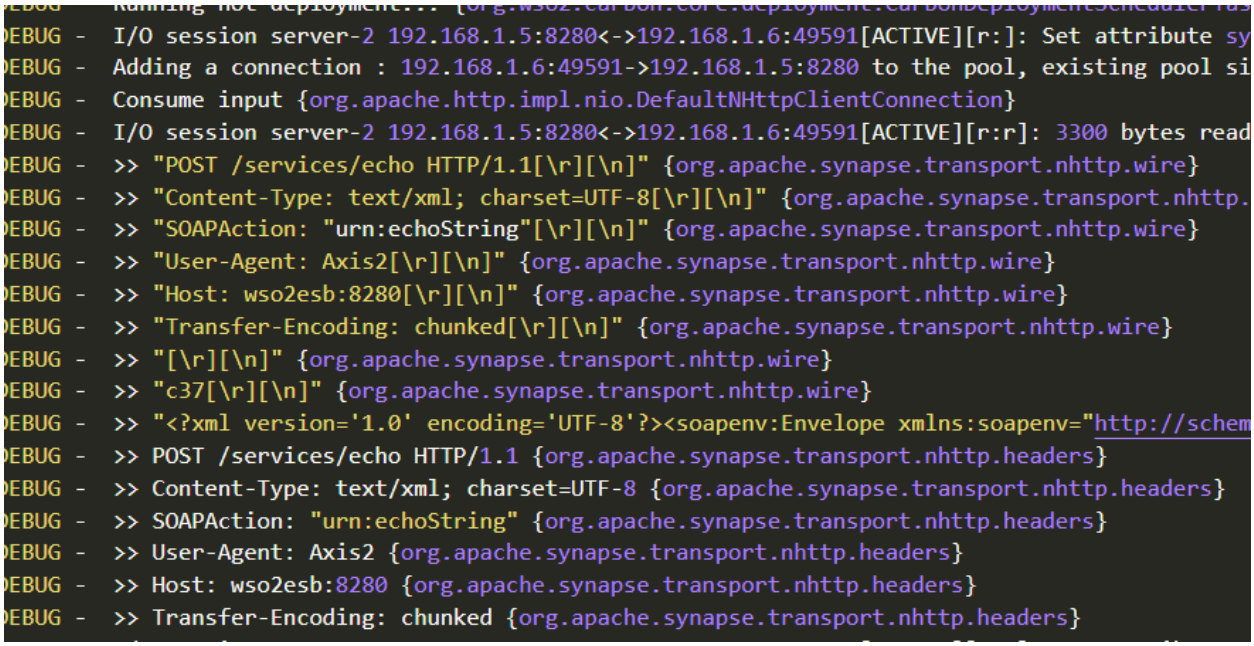


Figura 4-52 – Inicio de sesión del cliente contra el servicio de eco

En la captura de pantalla superior se puede comprobar la petición POST que realiza el cliente contra el servicio de eco.

Una vez se hace esta petición, el ESB valida los tickets de Kerberos del cliente y crea una sesión encriptada:

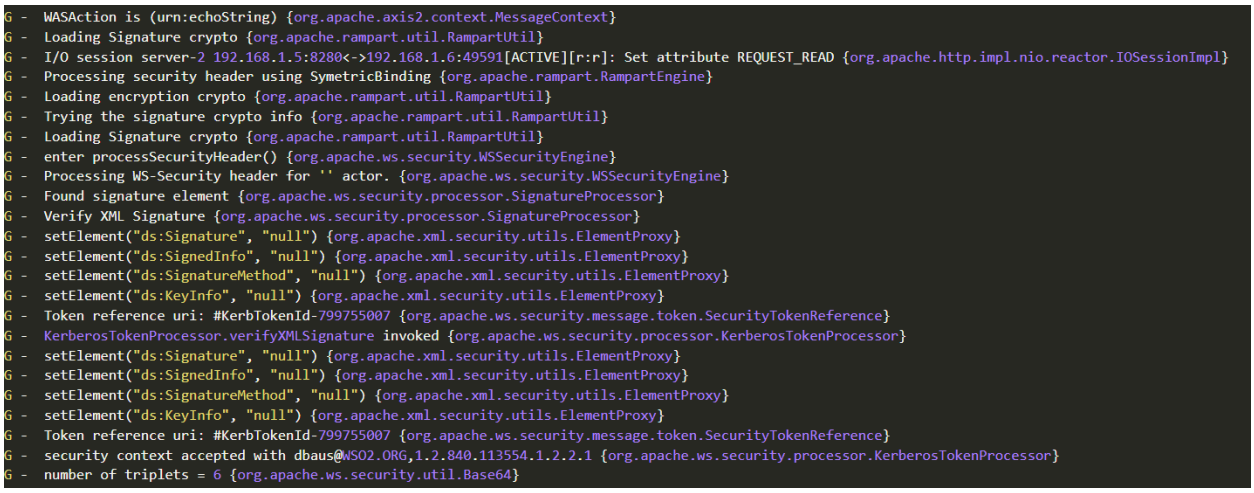


Figura 4-53 – Creación de sesión encriptada con el protocolo Kerberos entre cliente y servicio de eco

Figura 4-54 – Comprobación de marcas de tiempo de los tickets de Kerberos

Y una vez que ha validado al usuario, crea marcas de tiempo para los paquetes que va a enviar al cliente, los encripta y los envía:

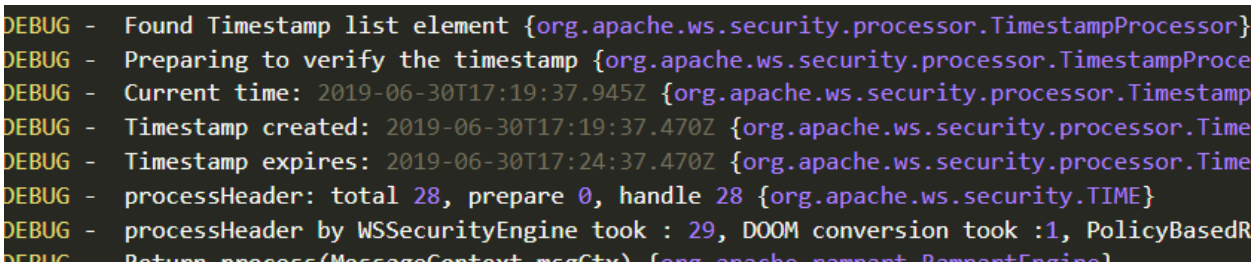


Figura 4-55 – Creación marcas de tiempo para respuesta del servicio de eco al cliente

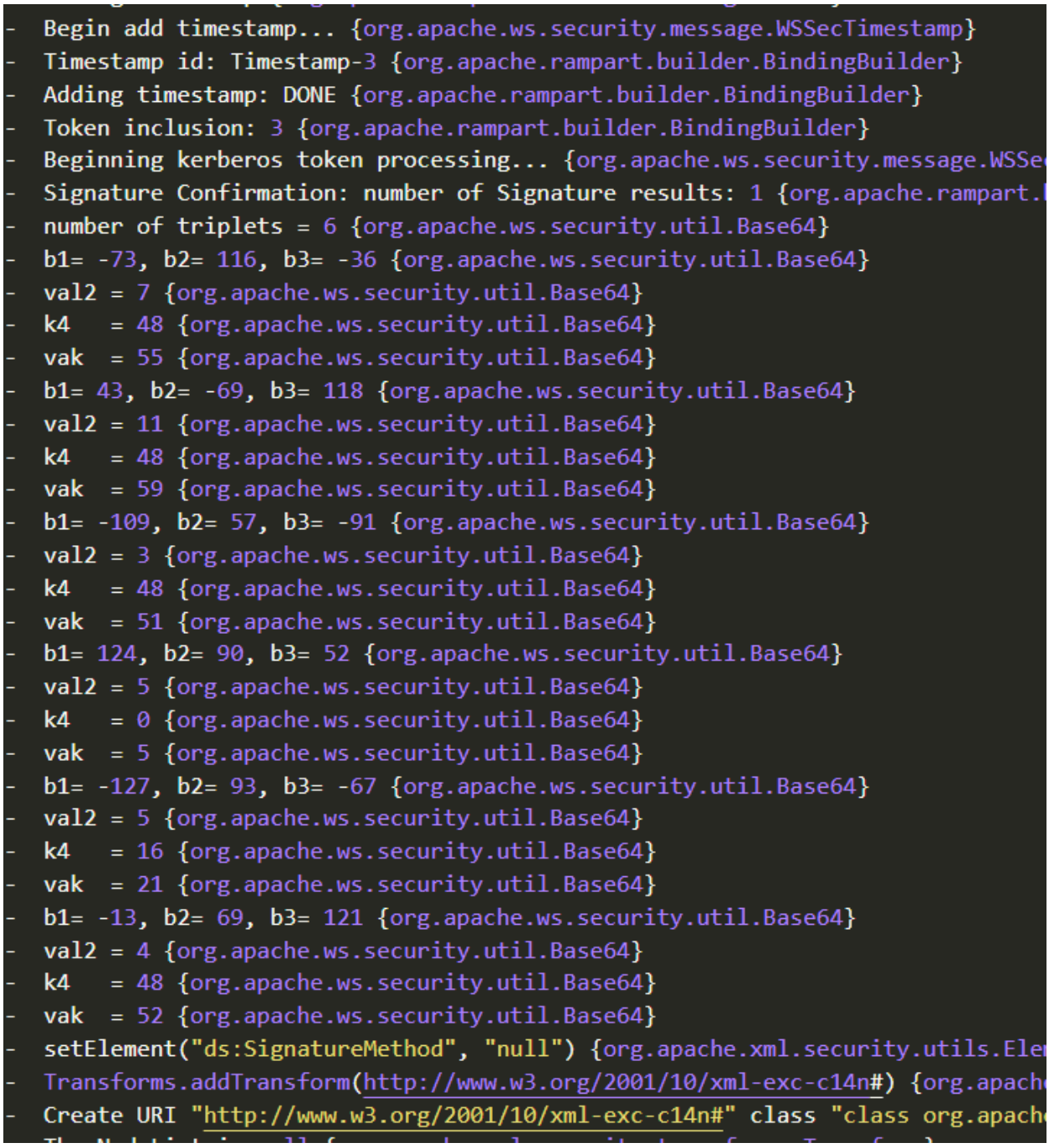


Figura 4-56 – Añadido de marcas de tiempo a trafico de servicio de eco a cliente

```
-  Produce output {org.apache.http.impl.nio.DefaultNHttpClientConnection}
-  I/O session server-2 192.168.1.5:8280<->192.168.1.6:49591[ACTIVE][w:w]: 159 bytes written
-  << "HTTP/1.1 200 OK[\r][\n]" {org.apache.synapse.transport.nhttp.wire}
-  << "Content-Type: text/xml; charset=UTF-8[\r][\n]" {org.apache.synapse.transport.nhttp.wi
-  << "Date: Sun, 30 Jun 2019 17:19:37 GMT[\r][\n]" {org.apache.synapse.transport.nhttp.wire
-  << "Server: Synapse-HttpComponents-NIO[\r][\n]" {org.apache.synapse.transport.nhttp.wire}
-  << "Transfer-Encoding: chunked[\r][\n]" {org.apache.synapse.transport.nhttp.wire}
-  << "[\r][\n]" {org.apache.synapse.transport.nhttp.wire}
-  I/O session server-2 192.168.1.5:8280<->192.168.1.6:49591[ACTIVE][w:w]: Clear event [w] {
-  start writeTo() {org.apache.axis2.transport.http.SOAPMessageFormatter}
-    preserve=false {org.apache.axis2.transport.http.SOAPMessageFormatter}
-    isOptimized=false {org.apache.axis2.transport.http.SOAPMessageFormatter}
```

Figura 4-57 – Envío de codigo de estado OK del servicio de eco al cliente

Una vez enviado esto, el servicio de eco marcará esa sesión como cerrada.

Estos mensajes han sido mostrados siguiendo el orden de llegada en cada servidor, basado en la información y marca de tiempo de cada mensaje recibido.

Se puede comprobar que el flujo del tráfico entre los distintos elementos en el escenario es el mismo que se describió al inicio del capítulo 4:

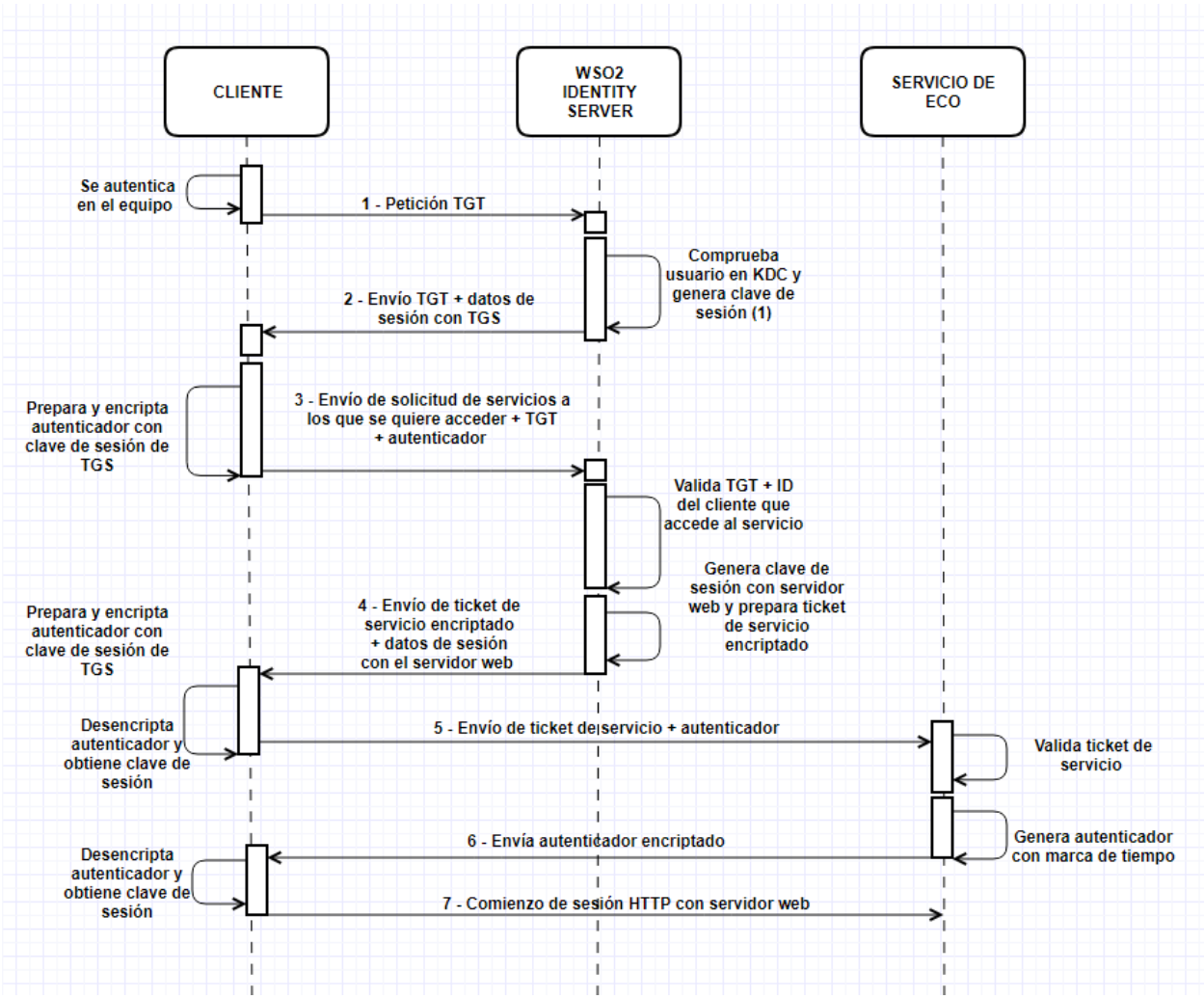


Figura 4-58 – Diagrama de paso de mensajes entre entidades del caso práctico

5 CONCLUSIONES Y TRABAJO FUTURO

Este proyecto aporta un caso práctico que puede utilizado en las sesiones prácticas de autenticación y control de acceso del Máster de Ciberseguridad que oferta la Universidad de Sevilla, actualizando así las actuales sesiones prácticas dedicadas a esta materia.

Gracias al caso práctico realizado en este trabajo se actualizan las sesiones prácticas dedicadas al control de la autenticación de usuarios. Dado que en este trabajo se han utilizado las mismas tecnologías que se utilizan en las sesiones prácticas del control de acceso de usuarios, es posible reutilizar este escenario para añadir la parte del control de acceso de usuarios.

5.1. Trabajo futuro

Este proyecto ha estado centrado en la parte del control de la autenticación de usuarios mediante el protocolo de Kerberos. Dado que en el Máster de Ciberseguridad se incluye una sesión dedicada al control de acceso de usuarios utilizando el servidor de identidad de WSO2, sería interesante ampliar el escenario propuesto en este proyecto con la parte del control de acceso.

Para ello, sería necesario modificar el servicio de eco y el agente del cliente utilizado en este proyecto para que soporte la validación de políticas de usuario mediamente XACML.

Al hacer esto, el beneficio que se obtiene es la conexión de las dos sesiones prácticas del Máster de Ciberseguridad sobre un mismo escenario, siendo mucho más sencillo para los alumnos ver las diferencias entre estos dos conceptos y como están relacionados entre sí.

6 BIBLIOGRAFÍA

- [1] WSO2, Information of the Company. Disponible en: http://wso2.com/about/. Último acceso el 6 de Septiembre de 2018
- [2] WSO2, Platform. Disponible en: http://wso2.com/platform. Último acceso el 6 de Septiembre de 2018
- [3] WSO2 API management. Disponible en: https://wso2.com/api-management/. Último acceso el 6 de Septiembre de 2018
- [4] WSO2 Identity Server. Disponible en: https://docs.wso2.com/display/IS530/WSO2+Identity+Server+Documentation. Último acceso el 6 de Septiembre de 2018
- [5] WSO2 Identity and access management. Disponible en: https://wso2.com/identity-and-access-management/. Último acceso el 6 de Septiembre de 2018
- [6] WSO2 IOT. Disponible en: https://wso2.com/iot . Último acceso el 8 de Septiembre de 2018
- [7] Middleware Architecture with Patterns and Frameworks (Capítulo 2). Disponible en: https://paginas.fe.up.pt/~apm/TDIN/docs/architecture_patterns_book.pdf. Último acceso el 8 de Septiembre de 2018
- [8] WSO2 Carbon. Disponible en: https://wso2.com/products/carbon/ . Último acceso el 8 de Septiembre de 2018
- [9] Service-Oriented architecture. Disponible en: https://en.wikipedia.org/wiki/Service-oriented_architecture. Último acceso el 8 de Septiembre de 2018
- [10] SOA reference architecture. Disponible en: http://www.opengroup.org/soa/source-book/soa_refarch/. Último acceso el 8 de Septiembre de 2018
- [11] Qué se entiende por SOA y cuales son sus beneficios. Disponible en: http://www.i2btech.com/blog-i2b/tech-deployment/que-se-entiende-por-soa-y-cuales-son-sus-beneficios/. Último acceso el 12 de Septiembre de 2018
- [12] Service-Oriented Architecture definition. Disponible en: https://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html. Último acceso el 12 de Septiembre de 2018
- [13] WSO2 Enterprise Service Bus documentation. Disponible en: https://docs.wso2.com/display/ESB500/WSO2+Enterprise+Service+Bus+Documentation. Último acceso el 12 de Septiembre de 2018
- [14] The Kerberos Network Authentication Service. Disponible en: https://tools.ietf.org/html/rfc1510. Último acceso el 12 de Septiembre de 2018
- [15] How the Kerberos V5 Authentication Protocol Works. Disponible en: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc772815(v=ws.10). Último acceso el 12 de Septiembre de 2018
- [16] Kerberos Authentication using WSO2 products. Disponible en: https://wso2.com/library/articles/2012/07/kerberos-authentication-using-wso2-products/. Último acceso el 22 de Septiembre de 2018
- [17] Project Athena. Disponibel en: https://en.wikipedia.org/wiki/Project_Athena. Último acceso el 22 de Septiembre de 2018

- [18] Java Authentication and Authorization Service. Disponible en: https://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html. Último acceso el 22 de Septiembre de 2018

- [19] Distributed Systems: Concepts and Design, 4/e - Couloris. Último acceso el 24 de Septiembre de 2018

- [20] Tech Journey, Secure WSO2 ESB REST APIs using Kerberos. Disponible en: http://hasanthipurnima.blogspot.com.es/2016/07/secure-wso2-esb-rest-apis-using-kerberos.html. Último acceso el 24 de Septiembre de 2018

7 GLOSARIO

SOAP: Simple Object Access Protocol

SOA: Service Oriented Architecture

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

XML: Extensible Markup Language

REST: Representational Transfer Protocol

KDC: Key Distribution Center

TGT: Ticket-Granting Ticket

TGS: Ticket Granting Server

AS: Authentication Server

IS: Identity Server

ESB: Enterprise Bus Service

PDP: Punto de decisión de políticas

PIP: Punto de información de políticas

API: Application Programming Interface

URL: Uniform Resource Locator
SMTP: Simple Mail Transfer Protocol
WSDL: Web Services Description Language
Apache: Web Server
Tomcat: Web Server focused on Java Applications
Java: Programming language
POP: Post Office Protocol
IMAP: Internet Message Access Protocol

Anexo A: Código del cliente del servicio eco

Este anexo incluye el código del cliente que accede al servicio de eco:

```
/**
 * El programa cliente que accede al servicio de eco bajo la aunteticacion de Kerberos.
 */
public class clienteKerberos {

    private static ConfigurationContext confContext = null;
    private static Policy servicePolicy = null;
    public static final String ANSI_YELLOW = "\u001B[33m";
    public static final String ANSI_RESET = "\u001B[0m";

    public static void main(String[] args) throws Exception {

        // Crea el contexto de configuracion utilizando el archivo client.axis2.xml
        confContext = ConfigurationContextFactory.createConfigurationContextFromFileSystem("repo",
                "repo/conf/client.axis2.xml");

        servicePolicy = loadServicePolicy("repo/conf/policy.xml");


        run();

        System.exit(0);
    }

    /**
     * Carga las politicas del cliente.
     * @param xmlPath ruta hacia el archivo de politicas.
     * @return un objeto de politicas
     * @throws Exception si el objeto que contiene las politicas no puede ser creado.
     */
```

```
    private static Policy loadServicePolicy(String xmlPath) throws Exception {
        StAXOMBuilder builder;
        Policy policy;

        builder = new StAXOMBuilder(xmlPath);
        policy = PolicyEngine.getPolicy(builder.getDocumentElement());

        return policy;
    }

    /**
     * Construye la carga que se enviara en la peticion del cliente al ESB.
     * @param value la cadena que se enviara al servicio de eco.
     * @return un objeto que contiene la carga que se envia al servicio de eco.
     */
    private static OMElement getPayload(String value) {

        OMFactory factory = OMAbstractFactory.getOMFactory();
        OMNamespace  ns  =  factory.createOMNamespace("http://echo.services.core.carbon.wso2.org",
"ns1");
        OMElement elem = factory.createOMElement("echoString", ns);
        OMElement childElem = factory.createOMElement("in", null);
        childElem.setText(value);
        elem.addChild(childElem);

        return elem;
    }

    /**
     * Este metodo invoca al servicio.
     */
    public static void run() {

        ServiceClient client = null;
        String s = null;

                                                                                    try {
            // Obten la URL del servicio de eco por parametros
            Scanner reader = new Scanner(System.in);
            System.out.println("Introduce la URL del servicio de eco: ");
            System.out.println("(Ejemplo: http://wso2esb:8280/services/echo)");
            String URL_SERVICIO_ECO = reader.nextLine();
```

```
            client = new ServiceClient(confContext, null);

            Options options = new Options();

            // Configura el servicio al que se quiere acceder
            options.setAction("urn:echoString");

            // Configura el endpoint
            options.setTo(new EndpointReference(URL_SERVICIO_ECO));
            // Configura las politicas del servicio
            options.setProperty(RampartMessageData.KEY_RAMPART_POLICY, servicePolicy);

            client.setOptions(options);

            // Configura el modulo Rampart de Apache
            client.engageModule("rampart");

            // Obten la cadena que se le va a enviar al servicio de eco por parametros
            System.out.println("Introduce el texto a enviar al servicio de eco: ");
            String cadena_servicio_eco = reader.nextLine();
            reader.close();

            OMElement response = client.sendReceive(getPayload(cadena_servicio_eco));

            System.out.println(ANSI_YELLOW + "### EJECUCION DEL CLIENTE WSO2 ###" + ANSI_RESET);
            System.out.println();
            System.out.println(ANSI_YELLOW + "Enviando al sericio de eco la cadena:" + ANSI_RESET
+ "\n" + ANSI_YELLOW + cadena_servicio_eco + ANSI_RESET);

            System.out.println();
            System.out.println(ANSI_YELLOW + "Respuesta del servidor WSO2ESB:" + ANSI_RESET + "\n"
+ ANSI_YELLOW + response + ANSI_RESET);

        } catch (AxisFault axisFault) {
            axisFault.printStackTrace();
            System.err.println("No he podido configurar correctamente el cliente que accede al
servicio");
        } finally {
            if (client != null) {
                try {
                    client.cleanup();
                } catch (AxisFault axisFault) {
                    System.err.println("Error limpiando el cliente que accede al servicio");
```

```
                }
            }
        }
    }
}
```