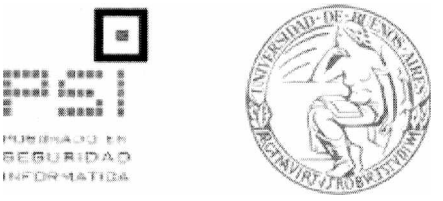


Universidad de Buenos Aires

Facultades de Ciencias Económicas,
Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final

Tema
Single Sign On, Servidores LDAP, Aplicaciones Web y Monitoreo de Eventos

Título
Implementación CAS con Servidores LDAP y Monitorización de Eventos de Seguridad en SSO Web

Autor: Flavio Alejandro Alfano

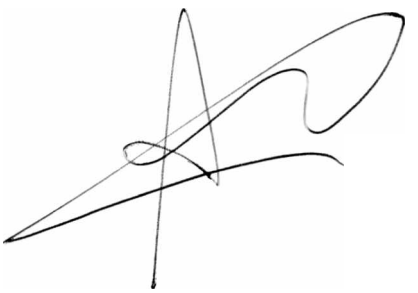
Tutor del Trabajo Final: Juan Alejandro Devincenzi

Año de Presentación: 2018

Cohorte del Cursante: 2015

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual".



Resumen

El presente trabajo enmarca en conjunto una serie de conceptos e implementaciones que brindarán un panorama completo sobre una arquitectura CAS-LDAP y su relación con las aplicaciones Web. En la primera base introductoria se brindarán detalles generales sobre la motivación del tema elegido, así como componentes esenciales de conocimiento que serán desarrollados para permitir al lector entender las interrelaciones de los componentes y sus funcionalidades.

El alcance es claro y acotado a lograr reproducir en un laboratorio una solución funcional. Este no es un trabajo meramente teórico, sino que pretende brindar las herramientas para que luego quien lo desee pueda continuar con su desarrollo y tenga también conocimiento de la variedad de opciones existentes para integrarse a una solución de este tipo. Hacia el final del trabajo se brindarán recomendaciones sobre futuros trabajos posibles relacionados con el desarrollado, entendiendo que las tecnologías utilizadas, así como la solución propuesta, tienen mucho potencial en diversos ámbitos y aplicaciones.

Aquellas personas con conocimiento de infraestructura, seguridad y desarrollo en nuevas tecnologías web encontrarán este informe como una herramienta de utilidad y de sumo interés. Por otra parte, aquellas personas interiorizadas en las herramientas de monitorización y ante interrogantes sobre monitorización de CAS, LDAP, autenticación de servicios Web, también lo encontrarán como un material sumamente útil.

Indice

Declaración Jurada de origen de los contenidos II

Resumen III

1 Introducción 1

1.1 Fundamentación del tema elegido 2

1.2 Objetivos y alcance 3

1.3 Metodología y Estructura del trabajo. 4

2 Desarrollo 5

2.1 Marco Teórico para una arquitectura CAS-LDAP-CENTOS. 5

2.1.1 CAS 5

2.1.2 LDAP 7

2.1.3 Centos 8

2.1.4 Nagios 9

2.2 Diseño de una arquitectura CAS-LDAP-CENTOS 11

2.3 Implementación de una arquitectura CAS-LDAP-CENTOS 11

2.4 Integración del Servidor CAS con un sistema de monitoreo. 25

2.5 Integración con Portales en PHP 28

2.6 Arquitectura Final CENTOS-CAS-LDAP-PHP-MONITOREO. 29

2.7 Implementación Final 31

2.7.1 Incorporación del módulo CAS Management Web App 31

2.8 Desarrollo de la prueba de concepto y casos de uso básicos. 44

2.8.1 Usuario con contraseña incorrecta y servicio habilitado. 44

2.8.2 Usuario con contraseña correcta y servicio no habilitado. 47

2.8.3 Usuario con contraseña correcta y servicio habilitado. 49

2.8.4 Respuesta del sistema de monitoreo del servidor CAS. 51

2.9 Interacciones de red con el servidor CAS (Foco - LDAP) 54

3 Conclusiones 57

4 Glosario 59

5 Apéndices y Anexos 61

6 Bibliografía 63

1 Introducción

En el presente trabajo se exponen todos los aspectos a ser considerados teniendo como objetivo principal brindar un ejemplo de implementación de un servidor CAS (Central Authentication Service) con servidores LDAP (Lightweight Directory Access Protocol) en su backend como soporte a la autenticación para luego lograr la integración a nivel de aplicaciones web en materia de single sign on.

Por otra parte, este trabajo también considera la inclusión de una prueba de concepto con ejemplos prácticos de monitoreo de eventos en la solución brindada.

En las siguientes secciones se brindarán más detalles como ser el fundamento sobre la elección del tema, hasta donde se llegará en materia de análisis, así como los objetivos del presente trabajo final.

Cabe destacar que el trabajo considera la introducción teórica al servicio CAS, al concepto de autenticación SSO (Single sign-on) y a la monitorización de eventos para luego alcanzar un ejemplo práctico con herramientas que faciliten su aplicación. Los componentes principales involucrados en el escenario práctico serán máquinas virtuales con sistema operativo Centos, el Proyecto Jasig como framework de CAS (el cual más adelante mencionaremos de forma breve), Nagios como software de monitorización y finalmente servidores LDAP como backend de CAS.

Palabras Clave: CAS, SSO, JASIG, LDAP.

1.1 Fundamentación del tema elegido

En estos momentos la diversidad de sistemas y la gran variedad de ellos con interfaces web hace necesario considerar los aspectos en materia de seguridad para por un lado facilitar las tareas de auditoría y control y por otro lado facilitar las tareas de mantenimiento y al mismo tiempo brindar facilidad de uso.

En estos momentos todas las organizaciones en mayor o menor medida poseen aplicaciones web como soporte a su negocio, las cuales en muchos casos incluyen un método de autenticación separado por cada sistema o integrado sólo parcialmente.

La gran mayoría de las organizaciones por requerimientos de auditoría suelen tener la necesidad de reportar todos los incidentes detectados según eventos de autenticación. Ante esta situación, brindar un medio para centralizar la autenticación de las aplicaciones WEB, integrar esa autenticación a servidores LDAP existentes y brindar una forma de monitorizar los eventos que surjan de la actividad de los distintos usuarios se presenta como un trabajo de sumo interés.

Por otra parte, el presente trabajo considera la utilización de soportes Open Source con el objetivo de facilitar su acceso a empresas con presupuesto ajustado en materia de Seguridad.

Una parte importante del trabajo de especialización se relaciona de forma estrecha con CAS (Central Authentication Service). CAS es un protocolo de Single Sign-on para aplicaciones Web. Su objetivo es permitir al usuario final acceder a diversas aplicaciones web ingresando sólo una vez su usuario y contraseña. Por otra parte, también permite a las aplicaciones web autenticar usuarios sin necesidad de tener acceso a sus credenciales como ser su contraseña. Más adelante en el cuerpo del trabajo se brindarán mayores detalles para comprender de forma general el servicio CAS.

Para finalizar la fundamentación comentaremos que es Jasig. Jasig es una organización fundada por un grupo de universitarios de TI en 1999. El objetivo de su creación era generar código para ser utilizado en entornos de educación superior. La mayoría del código generado era código Java. CAS en sus comienzos

fue creado en la universidad de YALE, pero en 2004 se volvió un proyecto de la organización JASIG llegando a ser conocido hoy como JASIG-CAS que originalmente era un software para ser utilizado por estudiantes de YALE de forma de tener un único sign-on en sus plataformas Web.

1.2 Objetivos y alcance

Objetivo General: Brindar un conocimiento general sobre una implementación de una arquitectura que utilice un servidor CAS, servidores LDAP como soporte a la autenticación para aplicaciones PHP y a su vez brinde un medio para monitorizar la actividad de seguridad del entorno implementado en materia de autenticación.

Objetivos específicos:

- Identificar y comprender el framework brindado por el proyecto JASIG en materia de servidor CAS considerando Linux como software de base y PHP como lenguaje WEB de las aplicaciones a ser integradas en SSO.
- Definir una arquitectura básica que considere un Servidor CAS, servidores LDAP y aplicaciones PHP.
- Identificar una opción para poder monitorear eventos de seguridad en un Servidor CAS según el proyecto JASIG.
- Implementar un entorno a modo de prueba de concepto que permita ver funcionando SSO para aplicaciones WEB PHP con un servidor CAS, servidores LDAP y también permita monitorear los eventos que se producen ante autenticaciones fallidas.

1.3 Metodología y Estructura del trabajo.

La metodología que se ha utilizado considera en primer lugar realizar una investigación sobre todo el conocimiento que se tiene sobre el tema adquirido en las materias cursadas y trabajos prácticos realizados, profundizando el mismo con herramientas web y soportes bibliográficos.

El objetivo siguiente es poder elaborar un trabajo que permita mostrar una aplicación práctica de una solución inicial que luego cualquier organización pueda extender según sus requerimientos.

Según lo antes mencionado, el trabajo será estructurado de la siguiente forma:

- Marco Teórico para una arquitectura CAS-LDAP-CENTOS.
  - CAS
  - LDAP
  - Centos
  - Nagios
- Diseño de una arquitectura CAS-LDAP-CENTOS
- Implementación de una arquitectura CAS-LDAP-CENTOS
- Análisis de Integraciones con PHP y la monitorización de Eventos.

- Diseño de arquitectura completa considerando en los componentes a CENTOS-CAS-LDAP-PHP-MONITOREO.
- Implementación de la Arquitectura Propuesta.
- Desarrollo de la prueba de concepto y casos de uso básicos

2 Desarrollo

2.1 Marco Teórico para una arquitectura CAS-LDAP-CENTOS.

2.1.1 CAS

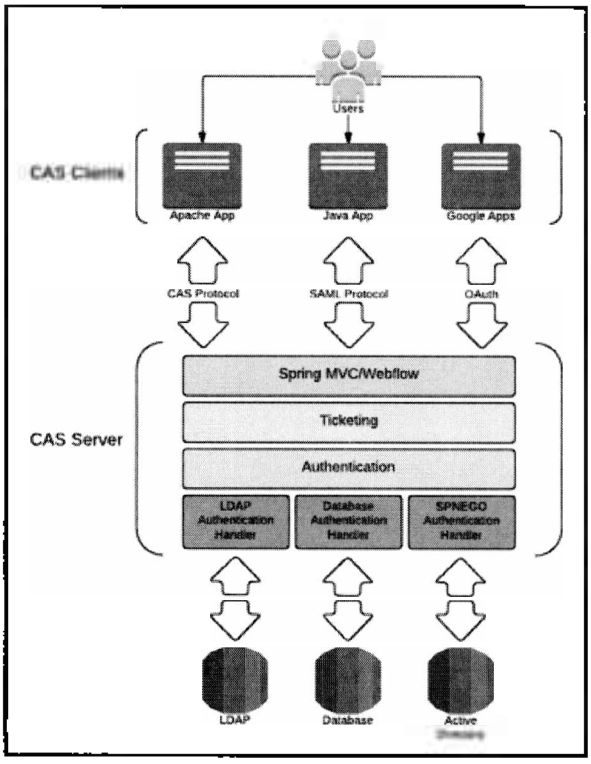
Entre los componentes principales que forman parte de una solución CAS se encuentran *CAS Server, CAS Clients y CAS Protocol.*

CAS Server es el componente principal en la plataforma propuesta en el presente trabajo; este servidor es un servlet java que permite en primer lugar la autenticación al usuario final y luego el acceso a los servicios definidos a través de la utilización tickets de acceso a los servicios. Si bien el CAS Server es ofrecido como una herramienta lista para ser utilizada, se requiere un conocimiento básico de implementaciones con Maven para poder realizar los ajustes a los paquetes ofrecidos según los requerimientos particulares de la instalación.

Luego de comentar el corazón de esta plataforma de SSO, continuaremos con los CAS Clients. Podemos entender a este componente según dos connotaciones: por un lado se denomina *CAS Clients* a las aplicaciones que pueden comunicarse con el componente *CAS Server* vía los protocolos CAS y por otro lado también encontraremos en la documentación a los CAS Clients como un software integrable con las aplicaciones para utilizar algún medio de autenticación como ser SAML u OAuth contra el Servidor CAS. En el conjunto de CAS Clients encontramos soporte para diferentes leguajes como ser: Java con Java CAS Client, .NET con .NET CAS Client, PHP con "phpCAS", Perl con PerlCAS, Python pycas y Apache httpd Server (mod_auth_cas module).

Si bien los backends no forman parte del módulo CAS Server, de los CAS Clients o de los protocolos CAS son una parte esencial del proceso de SSO. Los mismos se incluyen en el siguiente gráfico de arquitectura general de CAS y cabe destacar que a futuro nos interesará de forma particular los backends del tipo LDAP.

Arquitectura General CAS [1]



Con respecto al manejo de servicios a los cuales una arquitectura vigente de CAS brinda control, se pueden gestionar de 2 formas: directamente con los archivos de configuración (lo cual puede ser un poco desgastante para alguien no familiarizado en profundidad con esta solución de SSO) o a través de una herramienta brindada por el proyecto JASIG (que es la ***CAS Management Service Console***). El subcomponente "Registro de Servicios" es el soporte principal de la consola "CAS Management", dado que almacena toda la información relacionada con los servicios incluidos en la solución CAS. Algunas de las operaciones en las cuales interviene esta información son por ejemplo: determinar qué servicios pueden ser utilizados en una sesión de SSO, forzar la autenticación para determinados sitios, facilitar el control ante el uso de Proxy autenticación.

2.1.2 LDAP

Comenzaremos este punto describiendo que entendemos por LDAP de manera general. LDAP representa "Lightweight Directory Access Protocol", siendo LDAP un protocolo liviano utilizado para acceder a servicios de directorio basados en la norma X.500. Este protocolo corre sobre TCP/IP u otro protocolo orientado a conexión. Mayores detalles sobre este protocolo pueden encontrarse en la RFC 4510. Se ha considerado utilizar LDAP dado que al ser un servicio de directorio ofrece la posibilidad de administrar la información de forma centralizada al mismo tiempo que se permite almacenar y acceder a la información mediante métodos estándares ya validados y ampliamente reconocidos.

El concepto de LDAP se basa en un modelo cliente servidor en el cual uno o más servidores LDAP contienen la información de directorio. Los clientes se conectan a los servidores realizando preguntas o comúnmente más conocidos como querys. El servidor responde con una respuesta o un puntero hacia donde se puede consultar información de más utilidad sobre el tema.

En el presente trabajo se estará utilizando OpenLDAP como herramienta de servidor LDAP. OpenLDAP es un proyecto colaborativo de ámbito mundial que ofrece una gran diversidad de herramientas relacionadas con el protocolo LDAP, partiendo desde brindar un servidor LDAP hasta diversas tools de suma utilidad durante la administración, así como también clientes que permiten el acceso a los servicios ofrecidos.

En el caso de la suite OpenLDAP el servicio de servidor LDAP es ofrecido por el servicio SLAPD, el cual ofrece distintas variantes de seguridad para garantizar seguridad robusta y atención a la integridad de los datos. Por otra parte, existe también la posibilidad de utilizar TLS o SSL.

Entre los pasos a seguir para poder contar con el servicio de LDAP funcionando tenemos que tener en cuenta los siguientes:

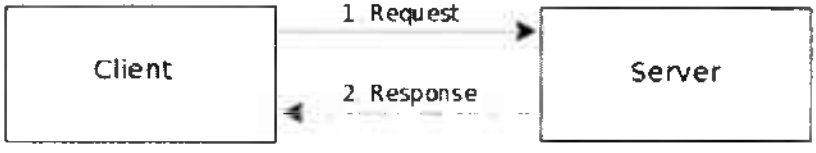
1. Adquirir el software.
2. Descomprimir la distribución.
3. Revisar la documentación (para validar por ejemplo requisitos de software).
4. Ejecutar el proceso de Configure.

5. Construir el paquete a ser instalado.
6. Testear el paquete construido.
7. Instalar el software.
8. Editar los archivos de configuración que sean necesarios.
9. Importar la Configuración de la base, lo cual puede ser de una existente o una nueva.
10. Iniciar el servicio SLAPD.
11. Agregar entradas requeridas en el directorio.
12. Verificar el funcionamiento.

Estos pasos pueden variar dependiendo de la implementación deseada, pero suelen ser casi siempre los mismos.

Si bien existen diversas variantes de configuraciones LDAP, estaremos utilizando servicio de directorio local, en la cual SLAPD brinda servicios para el dominio local únicamente como se muestra en la figura a continuación.

Configuración Local Service [2]



2.1.3 Centos

Centos es un sistema operativo Linux estable, muy versátil para la administración, con numerosas facilidades para el aprendizaje y el debuging. Por otra parte, provee un fácil manejo en lo que respecta a instalación de paquetes, manejo de firewalls, instalación de librerías y actualizaciones. Su código original deriva de RedHat Enterprise Linux, lo que de cierta manera transmite tranquilidad sobre temas de compatibilidad.

En el presente proyecto también se ha considerado la existencia de referencias sobre instalaciones exitosas de OpenLDAP y CAS sobre Centos, por lo cual se presenta como una gran herramienta de soporte.

En el portal de Centos[3] se pueden obtener numerosos manuales y guías para aquellos que comienzan sus primeros pasos en este sistema operativo, así como también son de utilidad para quienes buscan una referencia particular para la instalación o configuración de una determinada funcionalidad.

2.1.4 Nagios

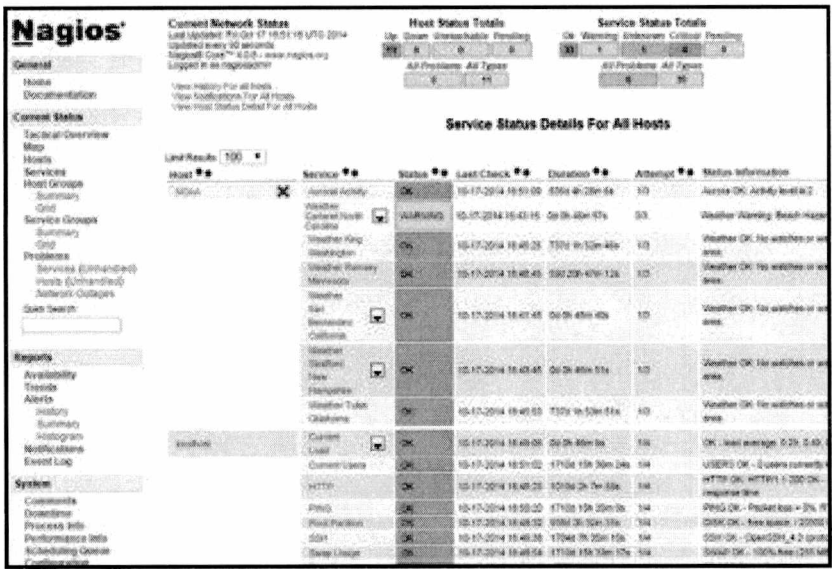
Nagios es una herramienta de monitoreo muy útil en el mercado de TI, suele ser utilizada para monitorizar la infraestructura y ofrece una gran variedad de plug-ins que permiten monitorizar distintos entornos como ser bases de datos, sistemas operativos, servidores web, servidores LDAP, estado de discos-filesystems, etc.

Nagios existe en una versión comercial y otra OpenSource. En nuestro caso, dadas las características del presente trabajo, se ha decidido utilizar la opción OpenSource. En esta versión, el componente principal es el modulo llamado Nagios Core. La función básica de Nagios Core es ser un planificador de eventos, procesador de eventos y manejador de alertas para los elementos siendo monitorizados. Gracias a diferentes APIs logra brindar una gran diversidad de funcionalidades. Este módulo Core tiene un demonio que fue escrito en C por razones de performance y está diseñado para correr en Linux o Unix de forma nativa.

Nagios Core posee básicamente un frontend CGI. Esta interface cuenta con varios CGIs que brindan a los usuarios de Nagios funcionalidades básicas con el objetivo de ver y manejar los elementos que son monitorizados por el servidor Nagios Core. Estos CGIs han sido definidos como estándar de Nagios y es por ello que se han utilizado en numerosas extensiones que brindan funcionalidades adicionales.

A continuación, se muestra una interfaz general que será utilizada una vez que Nagios Core quede productivo y funcionando reportando los eventos.

Ejemplo de consola de monitoreo nagios[4]



En lo que respecta al monitoreo del servidor CAS a través de nagios creo oportuno mencionar que el plugin que será utilizado para brindar información sobre el estado del servidor CAS al servidor Nagios fue finalmente desarrollado de forma particular, utilizando las herramientas de status incluidas en la instalación de los binarios del módulo CASServer.

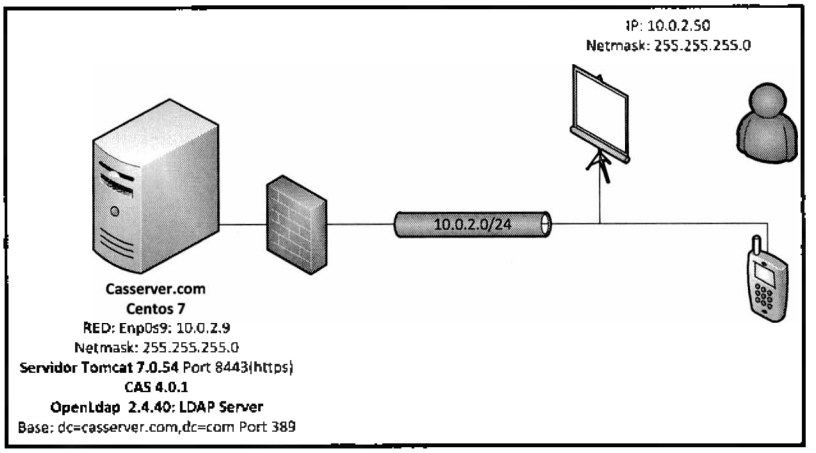
El plugin de Nagios que será utilizado se llama check-cas.pl - Nagios CAS (Central Authentication Service) check plugins. Este plugin realiza dos operaciones básicas: en primer lugar realiza un request para contar con el Login Ticket y luego realiza un request para autenticar con usuario y password. De forma adicional brinda información de performance a Nagios sobre el funcionamiento de CAS.

Se procederá con la inclusión de comandos de monitoreo adicionales para complementar al módulo check-cas.pl dado que en su versión actual aún no soporta varias de las características de las versiones de CAS más recientes.

2.2 Diseño de una arquitectura CAS-LDAP-CENTOS

Habiendo introducido los distintos conceptos, pasaremos a diseñar la arquitectura que estaremos implementando, en primer lugar considerando una integración de CAS y LDAP sobre Centos. Para ello se expone la siguiente imagen a modo de guía para luego pasar a describir los distintos componentes de la misma.

Arquitectura Inicial Propuesta



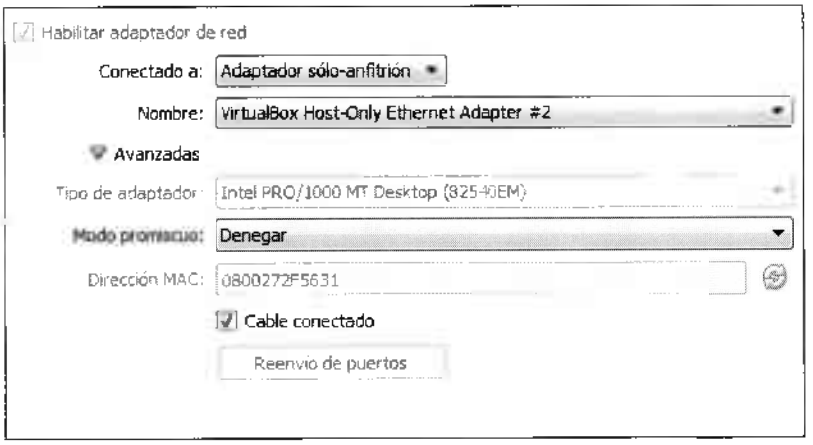
En esta arquitectura podemos observar la elección del puerto 389 y el protocolo LDAP en lugar del protocolo ldaps. La elección de esta metodología es poder visualizar el intercambio a nivel de red entre el servidor CAS y el servidor LDAP. Si bien en este caso se encuentran en el mismo servidor, en otra arquitectura podrían estar separados, por lo cual se recomendaría en un entorno productivo realizar las modificaciones necesarias para que el intercambio LDAP sea sobre un protocolo seguro.

2.3 Implementación de una arquitectura CAS-LDAP-CENTOS

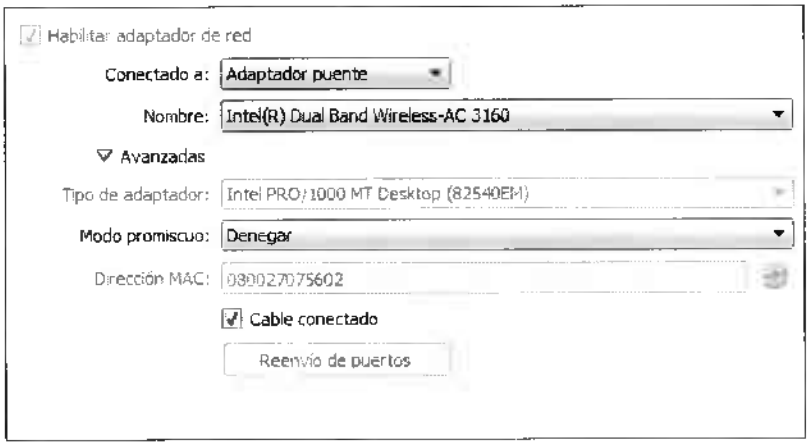
Durante la implementación de esta arquitectura básica se utilizó como herramienta de virtualización a VirtualBox por ser de descarga gratuita. Para parte del proceso de instalación y posterior interconexión de la plataforma, considerando contar con conectividad hacia Internet (comunicación con los repositorios de instalación) y local (acceso al servidor CAS y los futuros servicios de monitoreo), se

procedió a realizar varias configuraciones con el objetivo de dejar preparadas las interfaces según el siguiente detalle: la interfaz de Red 2 será la utilizada para la conectividad entre el servidor CAS y el cliente

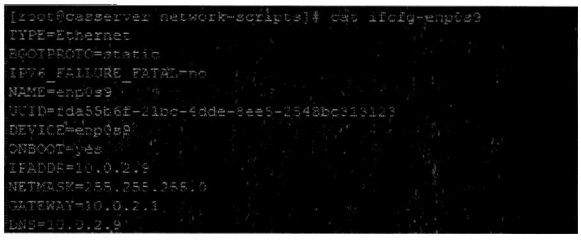
Interfaz de Red 2. Conectividad CAS Server – Cliente



Interfaz de con salida a internet para descarga de componentes de instalación.



A nivel de interfaces de OS se asignó una dirección IP fija según el siguiente detalle:



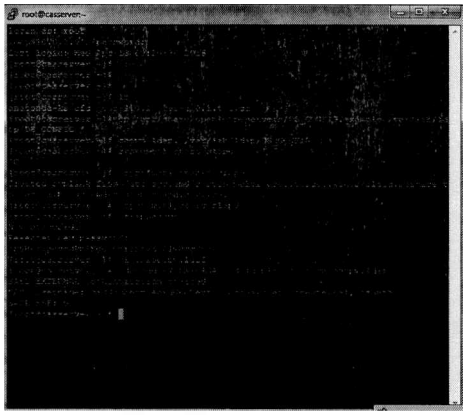
Teniendo ya las interfaces definidas y luego de haber instalado Centos 7 se procedió a instalar el servidor LDAP y el servidor CAS, lo cual fue realizado según el siguiente detalle:

Instalación del Servidor LDAP según la versión OpenLdap 2.4

1. El proceso de instalación comienza con la instalación de los paquetes de servidor y clientes de Open LDAP según el siguiente comando:

```
yum -y install openldap-servers openldap-clients
```

2. Luego siguen pasos relacionados con el inicio del servicio y configuración de la password de administración del servicio LDAP.



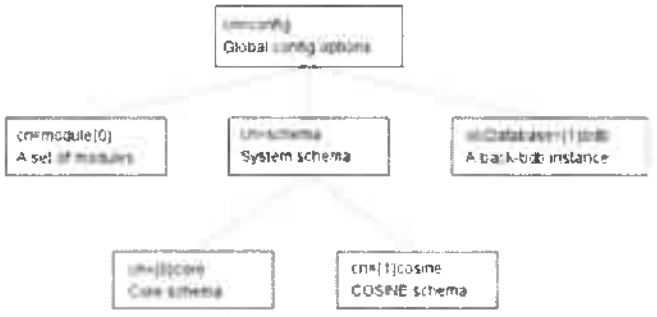
3. A continuación se procedió a importar los esquemas básicos para tener una estructura básica de directorio que luego será poblada y utilizada durante los procesos de autenticación con CAS según los siguientes comandos:

```
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
```

```
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f
/etc/openldap/schema/inetorgperson.ldif
```

Los servicios OpenLDAP almacenan la información de configuración dentro de un esquema y directorio predefinidos. A continuación se expone un formato que puede ser utilizado para almacenar la información de configuración sobre la cual se basa el servicio SLAPD.

Ejemplo de arbol de configuración [2]



En la raíz podemos encontrar donde se almacenarán los valores globales de configuración bajo la representación cn=config. Luego en cada uno de los hijos podemos ver definiciones particulares como ser Backends y Database.

4. A continuación se procedió a generar el nombre de dominio y activar la configuración en el servicio LDAP. Para lograr esto se utilizó basicamente un modelo de archivo *chdomain.ldif*, el cual fue ajustado según el dominio local definido. La estructura básica de este archivo fue consultada en el portal "server-world" en la sección "configuring LDAP"[5], así como también el archivo *basedomain.ldif*.

El archivo basedomain finalmente utilizado contó con la siguiente estructura:

```
[root@casserver classes]# more /root/basedomain.ldif
dn: dc=casserver,dc=tp.com
objectClass: top
objectClass: dcObject
objectclass: organization
```

```
o: casserver tp.com
dc: casserver
dn: cn=Manager,dc=casserver,dc=tp.com
objectClass: organizationalRole
cn: Manager
description: Directory Manager
dn: ou=People,dc=casserver,dc=tp.com
objectClass: organizationalUnit
ou: People
dn: ou=Group,dc=casserver,dc=tp.com
objectClass: organizationalUnit
ou: Group
```

Estos 2 archivos fueron incorporados a la configuración según los siguientes comandos:

```
ldapadd -x -D cn=Manager,dc=srv,dc=world -W -f basedomain.ldif
ldapmodify -Y EXTERNAL -H ldapi:/// -f chdomain.ldif
```

y los siguientes archivos:

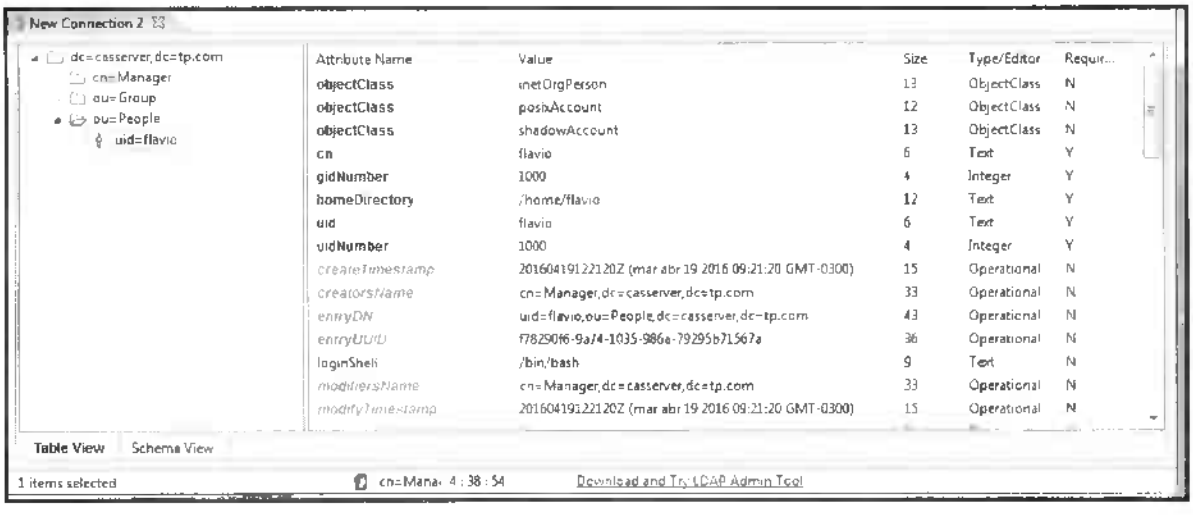
```
-rw-r--r--. 1 root root 1081 abr 15 09:09 chdomain.ldif
-rw-r--r--. 1 root root  409 abr 15 09:09 basedomain.ldif
```

5. Por último teniendo ya la estructura y dominio funcionando se procede a crear los usuarios que luego serían utilizados en la autenticación. Los archivos utilizados para crear los usuarios tuvieron como base el siguiente modelo, el cual fue el primer usuario funcional generado("flavio"):

La creación de usuarios no resulta ser una tarea muy sencilla, sobre todo para alguien que no tenga experiencia en la utilización de scripts de OpenLDAP. Si bien se contaron con formatos de usuario y archivos modelo de entrada, se utilizó una herramienta externa como soporte (LDAPBrowser) para confirmar la creación. Para aquellas personas que no tengan una administración frecuente de este tipo de entornos se recomienda que se apoyen con alguna herramienta externa de gestión de ambientes LDAP con opciones de administración del mismo.

La cuenta principal creada y que luego fue utilizada durante las pruebas fue el usuario "flavio".

Usuario Generado en LDAP Browser.



6. La única actividad faltante para poder ingresar al servidor LDAP sería finalmente habilitar el puerto 389 en el firewall ya que por default todos los puertos vienen deshabilitados.

Instalación del Servidor Tomcat y CAS 4.0.1

Para poder luego instalar CAS, componente que es en realidad un deploy sobre un servidor de objetos, en primer lugar se debe proceder a la instalación de un servidor Tomcat. En nuestro caso buscamos una de las últimas versiones de la linea Tomcat 7 (se ha elegido la versión 7.0.54).

La instalación del servidor Tomcat es muy sencilla dado que se puede proceder simplemente ejecutando la siguiente instrucción:

```
#yum install Tomcat
```

Luego de haberlo instalado Tomcat, restan hacer algunas acciones para que realmente quede utilizable y se pueda posteriormente incorporar el componente CAS, ya que por default este servidor está escuchando en el puerto 80 y mediante protocolo http. En su instalación estandar no se encuentra activo el uso del protocolo https y por otra parte a nivel de firewall de CentOS tampoco se encuentra habilitado el puerto relacionado.

Una vez finalizada la instalación se procedió de la siguiente manera para contar con un respositorio de llaves, parte esencial de la instalación:

- Creación de Keystore:

```
Keytool -genkey -alias Tomcat -keyalg RSA
```

Posteriormente también se debió permitir que la conexión hacia nuestro servidor Tomcat sea segura a través del protocolo https. Es importante destacar que esta característica, además de ser una opción obligatoria para contar con un

entorno seguro y no estar intercambiando información en plano, también es un requisito obligatorio del proceso de instalación del servidor CAS:

- Modificar el archivo server.xml para permitir el uso del puerto 8443.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

- Agregar la información relacionada con el keystore en el archivo server.xml.

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
           maxThreads="150" keystoreFile="/tomcat/.keystore" keystorePass="
```

- Agregar al firewall el acceso por 8443 también para que no sea filtrado a nivel firewall de centos.

```
firewall-cmd --permanent --zone=public --add-port=8443/tcp
firewall-cmd --reload
```

Finalmente se inicia el servicio de Tomcat con el comando:

```
systemctl start Tomcat
```

Y verificamos su estado con el comando:

```
systemctl status Tomcat
```

```
[root@casserver ~]# systemctl status tomcat.service
● tomcat.service - Apache Tomcat Web Application Container
   Loaded: loaded (/usr/lib/systemd/system/tomcat.service; disabled; vendor preset: disabled)
   Active: active (running) since mar 2016-04-19 00:24:   ART; 2h    ago
  Process: 7285 ExecStop=/usr/libexec/tomcat/server stop (code=exited, status=0/SUCCESS)
 Main PID: 7323 (java)
   CGroup: /system.slice/tomcat.service
           └─7323 /usr/lib/jvm/jre/bin/java -classpath /usr/share/tomcat/bin/bootstrap.jar:/usr/share/tomcat/bin/...
```

Luego de Instalar el servidor Tomcat es necesario instalar el servidor CAS para lo cual es importante validar la versión de los binarios a utilizar. Se analizó la web oficial de CAS con el objetivo de determinar cual era la versión indicada teniendo en cuenta el soporte posible ante futuros inconvenientes y la documentación existente para garantizar que el prototipo funcione de forma correcta. Adicionalmente se intento garantizar que luego se contará con información de soporte para poder proceder a la configuración LDAP y que se pueda contar con el soporte en documentación para todas las configuraciones necesarias según la arquitectura definida y los casos de uso a considerar. De este análisis surgio la idea de utilizar CAS 4.0.1. Otro aspecto que a considerar fue la

existencia de casos de exito en materia de configuraciones de backends, upgrades o APIs que utilicen servicios CAS ya que en el pasado, según la experiencia del autor del presente trabajo, una mala elección de la versión puede resultar en muchas horas de trabajo adicionales y la inviabilidad de desplegar un caso de uso.

El proceso de instalación comienza con la instalación de Maven, la cual es una utilidad que nos permitirá facilmente editar las propiedades y codigo original para permitir generar un paquete que luego implementaremos en el servidor Tomcat previamente instalado. Maven se utilizará como principal herramienta de compilación y construcción de la solución CAS. La instalación de Maven se puede realizar de diversas maneras: de forma manual o mediante la utilidad yum. En el entorno de demo se ha procedido de la siguiente forma considerando la versión **3.0.5**:

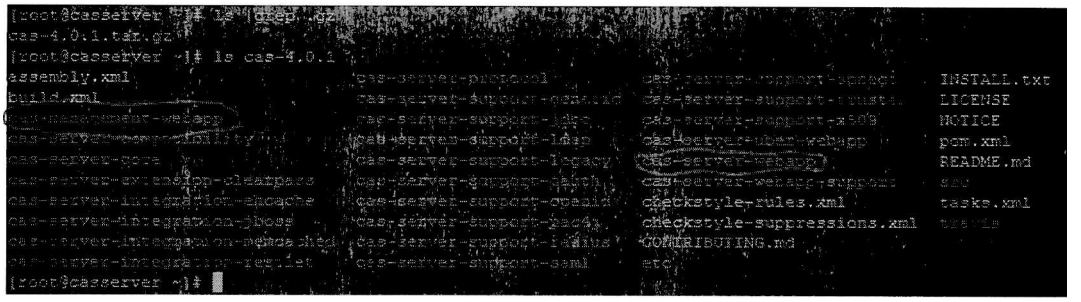
```
# wget http://mirrors.dcarsat.com.ar/apache/maven/maven-
3/3.0.5/binaries/apache-maven-3.0.5-bin.tar.gz
# sudo tar xzf apache-maven-3.0.5-bin.tar.gz -C /usr/local
# cd /usr/local
# sudo ln -s apache-maven-3.0.5 maven
```

Luego confirmamos la versión instalada de la siguiente forma:

```
[root@casserver ~]# mvn -version
Apache Maven 3.0.5 (Red Hat 3.0.5-16)
Maven home: /usr/share/maven
Java version: 1.7.0_99, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.99-2.6.5.0.el7_2.x86_64/jre
Default locale: es_AR, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-229.el7.x86_64", arch: "amd64", family: "unix"
[root@casserver ~]#
```

Una vez instalado el producto Maven, es momento de proceder a instalar el servidor CAS; para ello primero descargamos el paquete cas-4.0.1.tar.gz con el objetivo de tener todos los fuentes que luego construiremos con el comando maven correspondiente.

Una vez descargado este archivo y luego de descomprimirlo podemos observar una estructura de directorios como la siguiente:



En esta estructura nos interesan particularmente 2 directorios, por un lado el "cas-server-webapp" que será nuestro módulo servidor y por otro lado el modulo "cas-management-webapp" que será una herramienta que nos facilitará notablemente la configuración de los servicios cuando queramos configurarlos en un futuro.

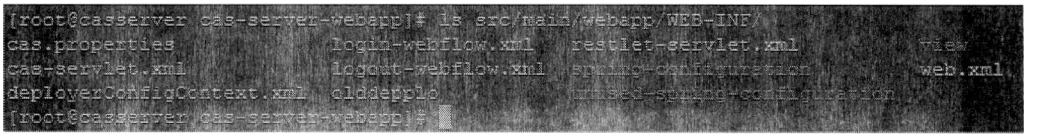
En esta sección estaremos comentando el proceso para implementar el modulo cas-server-webapp considerando por un lado el soporte para LDAP dentro de las dependencias y por otro lado todas las configuraciones necesarias para lograr que realmente funcione nuestra implementación.

En primer lugar habilitamos la opción de soporte a LDAP en el modulo cas-server-webapp; para ello tuvimos que editar el archivo pom.xml (El Project Object Model o POM). Este archivo es una parte fundamental en Maven y es el que brinda al proceso de construcción todo el detalle de dependencias necesarias. Si bien es un archivo XML al cual podríamos dedicar varias páginas, lo importante es que sepamos que contiene información sobre el Proyecto y las particularidades requeridas y utilizadas por Maven para construcción del mismo como se puede observar en el siguiente detalle al habilitar LDAP:

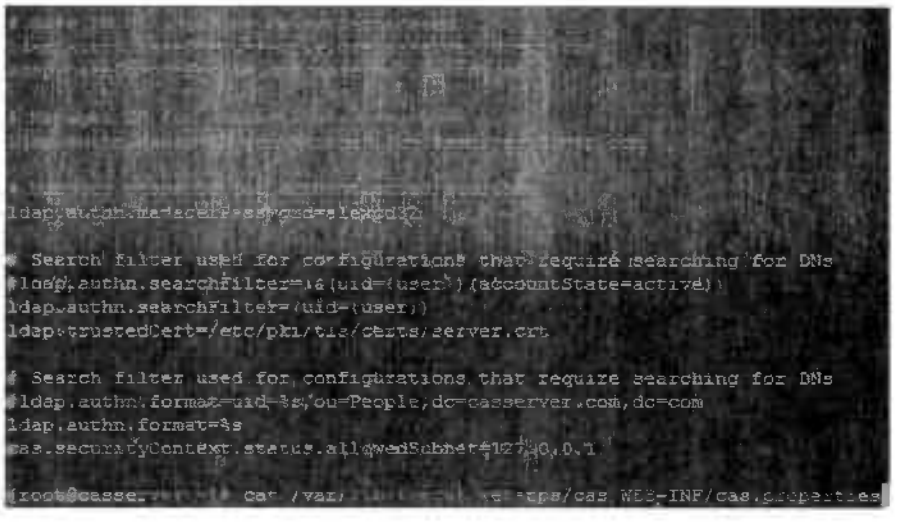
```
[root@casserver cas-server-webapp]# pwd
/root/cas-4.0.1/cas-server-webapp
[root@casserver cas-server-webapp]# ls
cas.log  NOTICE  perfStats.log  pom.xml  src  target
[root@casserver cas-server-webapp]#
```

```
        <dependency>
          <groupId>org.jasig.cas</groupId>
          <artifactId>cas-server-support-ldap</artifactId>
          <version>4.0.1</version>
    </dependency>
```

A continuación, siguiendo con el proceso de configuración, fue necesario editar dos archivos fundamentales en cualquier instalación CAS que soporte LDAP: *deployerConfigContext.xml y cas.properties*



En primer lugar se procedió a cargar en el archivo cas.properties las definiciones básicas para permitir la autenticación LDAP como ser la ruta del servidor, credenciales de manager, basename y formato a utilizar para consultar los distintos usuarios.

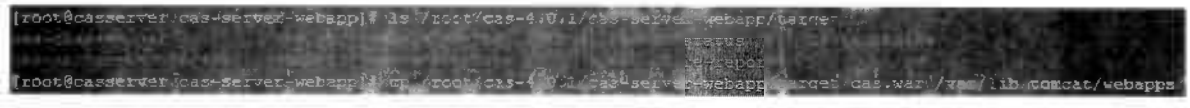


Contando con la configuración base del archivo cas.properties, se procedió con la sección más complicada de la configuración que es el archivo deployerConfigContext.xml. En este archivo de deben configurar básicamente los componentes "LdapAuthenticationHandler" (este componente identifica un usuario y contraseña contra un servidor LDAP el cual puede ser OpenLDAP, como es nuestro caso, o también active directory), "authenticationManager" (para que se establezca el nuevo manejador de autenticación como LDAP), dnResolver llamado por el componente "authenticator" (en este componente se define como serán resueltas las consultas y como serán manejadas las autenticaciones), el cual a su vez es requerido por componente "LdapAuthenticationHandler".

Esta relación de componentes resulta un poco confusa de entender y la documentación encontrada en internet no resulta del todo clara y requiere troubleshooting y análisis adicional para poder contar con una comprensión completa de su funcionamiento, así como para poder implementarlos. Si bien por temas de facilidad de lectura no se incluyen en este texto los archivos, se adjuntaran los archivos generados durante la instalación para que todo aquel que en un futuro necesite revisarlos pueda contar con ellos. También se incluirán ejemplos comprobados de su funcionamiento, de forma de ahorrar tiempo valioso en la configuración.

Una vez realizadas todas las configuraciones antes mencionadas, se procede a realizar la construcción del paquete cas-server-webapp mediante el siguiente comando:

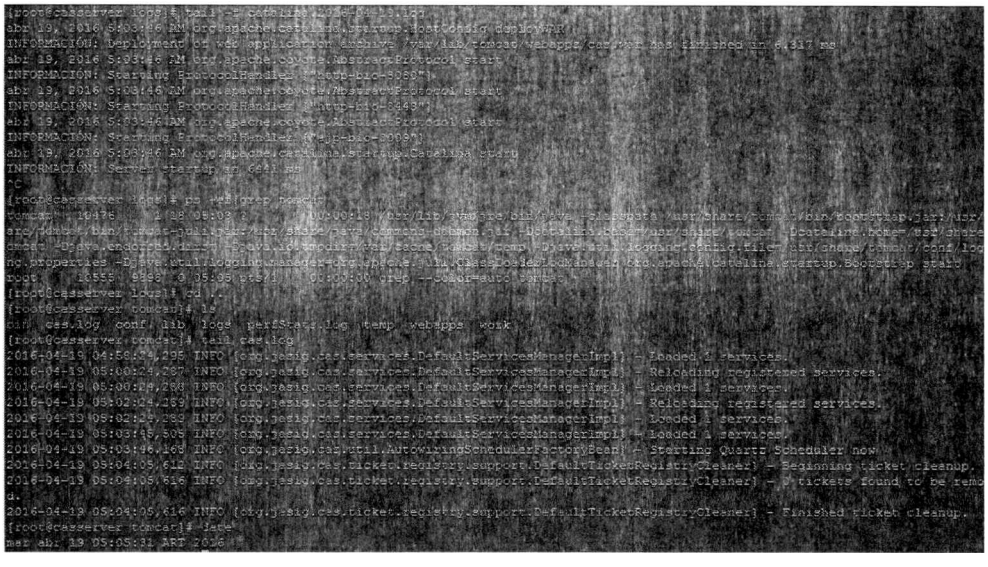
Si bien observamos la leyenda "**BUILD SUCCESS",** como resultado exitoso, es necesario revisar todo el proceso y en lo posible redirigirlo a un archivo para no perder la salida y evaluarlo posteriormente con tranquilidad. Un warning previo puede representar algún inconveniente futuro al momento de la utilización de la plataforma. El autor tuvo que resolver varios problemas de comunicaciones para permitir alcanzar esa construcción exitosa, así como también inconsistencias en los archivos de configuración. Teniendo la construcción exitosa, tendremos un war generado el cual debemos localizar y luego mover a nuestro servidor Tomcat de la siguiente forma:



Finalmente reiniciamos el servidor Tomcat con el comando:

```
#systemctl restart tomcat
```

Y validamos que el servidor Tomcat inicie sin problemas revisando el archivo catalina.out así como también validamos el log cas.log para verificar que se hayan cargado los servicios:



A continuación mostramos el funcionamiento del portal de autenticación de CAS al cual se accederá utilizando el usuario de LDAP "flavio".

Para testear el funcionamiento de CAS, procedemos en primer lugar a acceder mediante un navegador a la url: https://10.0.2.9:8443/cas/. Seguido a esto nos aparecerá la siguiente pantalla en la cual haremos login con el usuario "***flavio***".

Pantalla de Login frente al servicio CAS

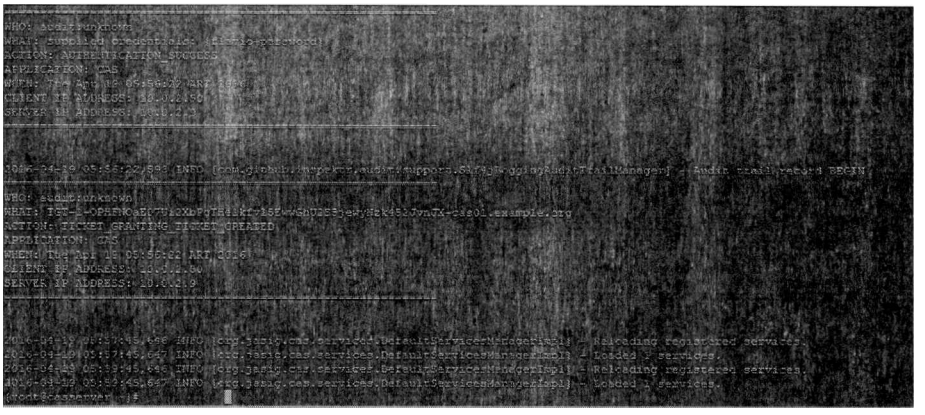
En esta pantalla intentamos un primer ingreso para validar el funcionamiento de la integración CAS/LDAP. Mediante la siguiente pantalla la aplicación nos menciona que el Log In fue exitoso.

Confirmación de autenticación exitosa

Ahora es momento de verificar los logs para confirmar un comportamiento correcto.

También podría ser oportuna la ejecución del comando tcpdump y el análisis de la salida con la tool wireshark para verificar el intercambio de paquetes.

El funcionamiento lo podemos observar al consultar el archivo cas.log



En ese archivo obtendremos la información sobre todo el proceso de autenticación y detalles como ser el manejador utilizado en el proceso.

En todo el proceso de debug del servidor CAS existen los ya mencionados archivos de logs, pero existe un archivo llamado log4j.xml que es de suma utilidad y que define el nivel de log que observaremos en los distintos archivos de logs. Por otra parte, también permite definir que nos interesa loguear, de forma de tener un archivo claro para identificar el inconveniente. Este archivo contiene hacia el final una sección que nos permite cambiar de un modo info a un modo debug:

<root>

<level value="DEBUG" />

<appender-ref ref="console" />

</root>

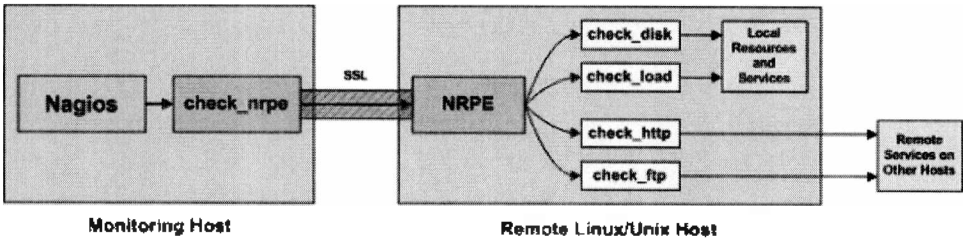
2.4 Integración del Servidor CAS con un sistema de monitoreo.

En lo que respecta a la ***monitorización de eventos,*** estaremos utilizando el cliente de Nagios NRPE en el servidor CAS. El mismo nos permitirá por un lado tener visibilidad de la salud del servidor a nivel general de sistema operativo y por otro lado permitirá saber en todo momento el estado y disponibilidad de las funcionalidades brindadas por el deploy en el servidor de objetos Tomcat, así como también su estado de consumo de memoria en término de procesos java y los tickets de servicio que está entregando.

En primer lugar, pasaremos a describir que es el plugin NRPE de nagios. Este componente brinda la funcionalidad de ejecutar remotamente plugins de Nagios en otros equipos Linux / Unix. Esto nos permitirá monitorear las métricas del servidor remoto (espacio utilizado en disco, CPU, Procesos y métricas de CAS).

A continuación, se expone de forma gráfica su funcionamiento considerando el servidor de monitoreo (Nagios Core Server) y el servidor monitorizado (CAS Server).

Funcionamiento del componente de Nagios NRPE [6]



La funcionalidad del servidor CAS que permite ver su salud es la subrutina /cas/status. Para poder monitorizar la salud del servidor CAS y acceder a esta característica, lo primero que hay que hacer es habilitar la dirección IP desde la cual se consumirá el estado modificando el archivo cas.properties, de forma se permita la consulta a la url ya que por default se encuentra filtrado el acceso. A continuación se indica la línea a editar:

cas.securityContext.status.allowedSubnet=127.0.0.1

Luego debemos crear un script que sea capaz de obtener estas métricas, el cual será utilizado por el proceso NRPE para informar al servidor de nagios la salud del servidor CAS. Finalmente hay que incorporar esta rutina al monitor nrpe para que el servidor de nagios pueda invocar su ejecución de forma remota.

```
root@casserver nrpe.d]# pwd
/etc/nrpe.d
root@casserver nrpe.d]# ls
pp5_commands.cfg
```

```
[root@casserver nrpe.d]#
command[check_cas]=/usr/lib64/nagios/plugins/check_cas
```

Por otra parte, dentro de la carpeta /usr/lib64/nagios/plugins procedemos a crear un script capaz de invocar a la url "/cas/status" y brindar un exit 0 o 1 dependiendo si hay o no que reportar una alarma:

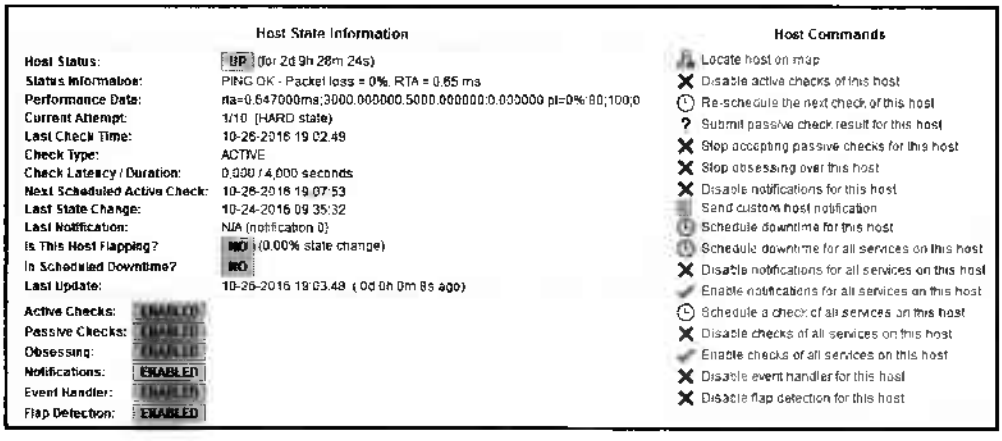
```
[root@casserver plugins]# ./check_nrpe -H 127.0.0.1 -c check_cas
OK -    1.MemoryMonitor: OK - 11,56MB free, 55,02MB total. -    2.SessionMonitor: OK - 1 sessions. 0 service t
ckets.
[root@casserver plugins]#
```

De esta forma, se pueden visualizar los eventos en el servidor de monitoreo eventos según el siguiente detalle:

Consola de Nagios. Vista del servidor CAS general.

Host	Service	Status	Last Check	Duration	Attempt	Status Information
casserver	Check Load	OK	08-31-2016 06:45:28	0d 0h 19m 5s	1/3	OK - load average: 0.00, 0.01, 0.05
	Current Users	OK	08-31-2016 06:44:05	0d 1h 2m 28s	1/3	USERS OK - 3 users currently logged in
	cas health	OK	08-31-2016 06:44:17	0d 0h 52m 16s	1/3	OK - 1 MemoryMonitor: OK - 15,09MB free, 55,02MB total. - 2.SessionMonitor: OK - 1 sessions. 0 service tickets.

Consola de Nagios. Vista del servidor CAS detallada.



El monitoreo del CAS Server puede ser expandido incluyendo también los componentes internos y también monitorizar a nivel de OpenLDAP.

Mediante las configuraciones propuestas y monitores activados se brinda una solución que permite en todo momento saber el estado del servidor CAS y de que manera se encuentra prestando servicio. Más adelante en el presente trabajo se explotarán distintos casos de uso y detalles adicionales que permitirán completar la imagen de la solución propuesta de monitorización.

2.5 Integración con Portales en PHP

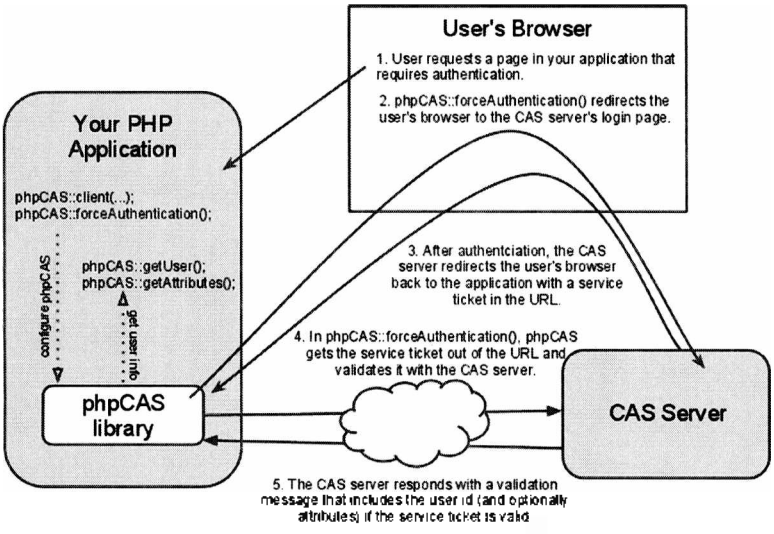
En lo que respecta a la integración con aplicaciones PHP, en primer lugar, debemos contar con un servidor con soporte a PHP. En nuestro caso elegiremos apache, para lo cual se ha optado por instalar el paquete XAMPP el cual brinda un entorno de gran utilidad ya sea para pruebas de integración como para desarrollo en PHP.

Este paquete incluye los submodulos MariaDB, PHP y Perl los cuales son muy útiles para la arquitectura y solución propuesta. Su utilidad radica en que por un lado necesitamos un Servidor Web para contar con un servicio remoto capaz de presentar los front-end de las aplicaciones PHP, el cual será quién aloje los servicios de la solución según los casos de uso a ser presentados, y por otro lado en que con este paquete también resolvemos la necesidad de una base MySQL para poder persistir los datos de los servicios en caso de que podamos y decidamos aplicar persistencia de datos. De no persistir los servicios la configuración de CAS siempre mantendría la información en memoria y esta sería leída de unos archivos XML que costarían mucho editar y mantener a diferencia de una GUI Web persistida en la base MySQL.

En el presente trabajo se propone correr este producto en un equipo diferente para brindar esa sensación de integración de una forma más tangible para un usuario normal.

Un componente importante de la integración PHP propuesta es el paquete "PHPCAS Client". Este paquete es una API que permite la autenticación contra el servidor CAS mediante una serie de métodos y funciones de PHP.

Flujo entre CAS Server, API PHP Cas Client y el usuario [7]



Para las distintas pruebas así como para las configuraciones de la interfaz contra el servidor CAS vamos a utilizar una serie de ejemplos que vienen con la API de PHP con el objetivo de poder brindar respuestas a interrogantes que pueden surgir en una integración de este tipo como ser: ¿Cómo es la configuración inicial de la misma considerando los ficheros cas.php y config.php?, ¿Cuáles son los elementos principales para poder arribar al servidor CAS para realizar un intercambio de autenticación teniendo una aplicación PHP?

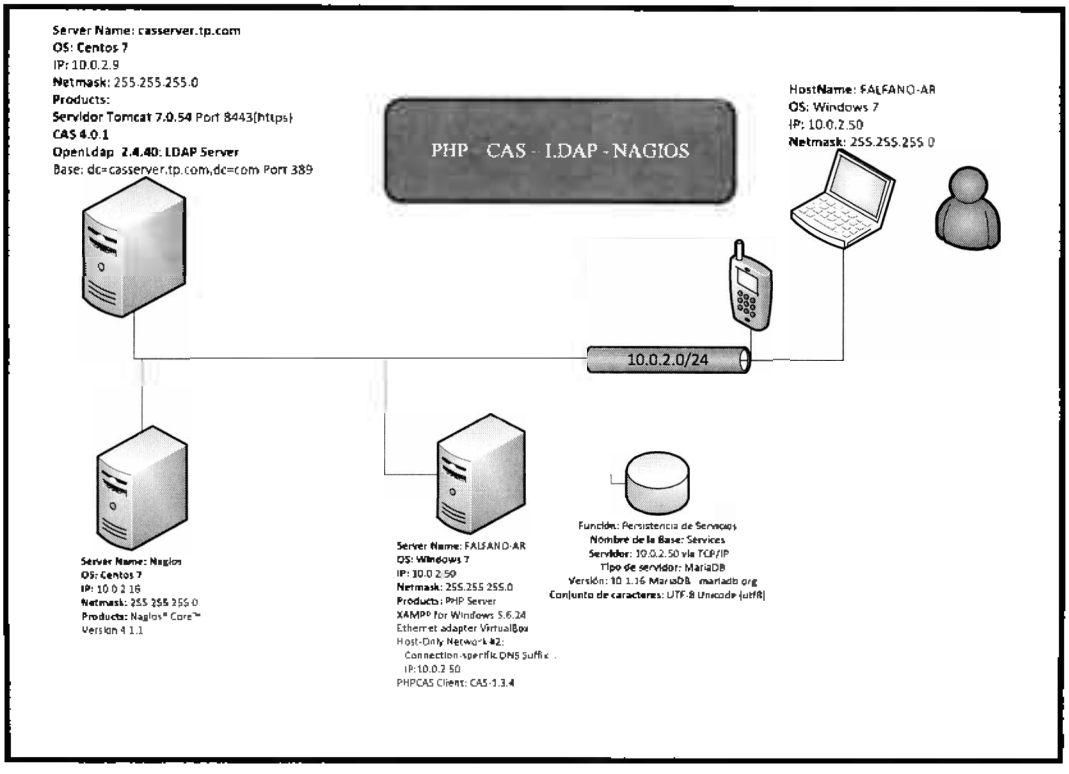
2.6 Arquitectura Final CENTOS-CAS-LDAP-PHP-MONITOREO.

Habiendo analizado todos los componentes de la arquitectura principal: considerando el servidor CAS, los servicios de LDAP, luego habiendo analizado las integraciones con sistemas de monitoreo y finalmente verificado las integraciones con PHP; estamos en condiciones de presentar la arquitectura final propuesta del presente trabajo la cual incluirá todos los componentes mencionados a continuación:

- Servidor CAS con soporte para LDAP.
- Servidor Nagios integrado con el servidor CAS.

- Servidor Apache brindando servicios de PHP integrado con el servidor CAS.
- Entorno de Usuario Final corriendo sistema operativo Windows 7 y ejecutando aplicaciones PHP que soporte autenticación CAS

Arquitectura Final del Modelo propuesto extendido.



Dado que los componentes fueron descriptos ya sea en la etapa de la arquitectura inicial o en las secciones de integraciones, no volverán a ser descriptos en esta sección; en caso de dudas sobre los mismos se puede proceder a consultar a las siguientes secciones 2.2, 2.3, 2.4, 2.5.

2.7 Implementación Final

2.7.1 Incorporación del módulo CAS Management Web App

El interés en este módulo radica en ser una utilidad que facilita la gestión de los servicios bajo un entorno dependiente del servidor CAS. Básicamente ofrece facilidades a los administradores del servidor CAS para definir cómo pueden utilizar los servicios de los distintos CAS Clients. Esto se logra a través de un registro de servicios el cual puede almacenarse de distintas formas y en diversos soportes. Estos servicios y sus distintos parámetros definen los comportamientos de interés para este trabajo particularmente en lo que respecta a los servicios autorizados y autenticación forzada.

La instalación de este módulo en sí no resulta compleja, pero hay que modificar varios archivos de configuración para permitir al módulo CAS Management conectarse al servidor CAS y definir los esquemas de servicios acordes.

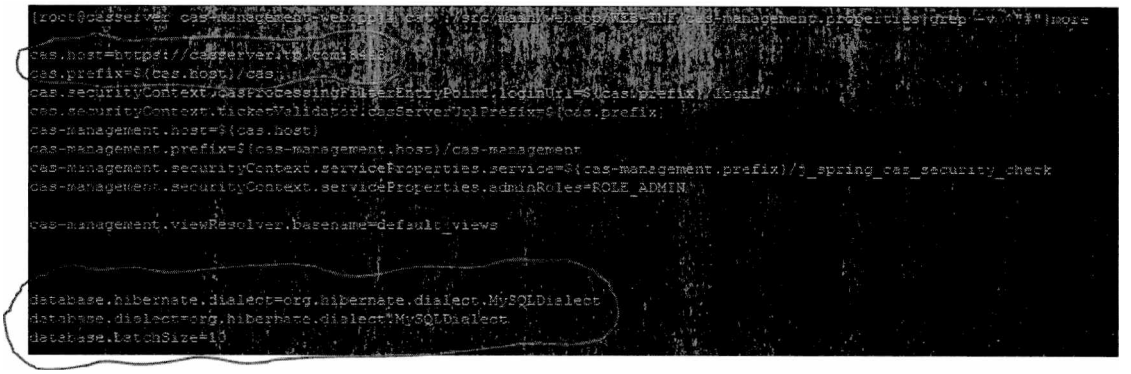
En primer lugar, pasaremos a comentar la ubicación de la carpeta de trabajo en la cual se editan los distintos archivos de configuración que permitirán la integración:

Para brindar mayor claridad sobre el proceso de instalación se propone dividir la instalación en dos fases: una llamada contexto básico de CAS Management y otra Persistencia de Datos la cual como se verá más adelante es bastante compleja.

La aplicación CAS Management finalmente se utilizará como un acceso web desde el cual se configuraran todos los servicios y el acceso a los mismos, entendiendo por servicio una front-end de una aplicación web por ejemplo.

Contexto Básico CAS-Management:

En esta fase, la idea es describir los archivos básicos que se modificaron de la imagen original tendiendo como objetivo brindar claridad para aquellos que quieran entender el proceso de instalación, así como facilitar el mismo en caso de que se desee aplicar. Estaremos resaltando las secciones modificadas en cada uno de los archivos comenzando con el archivo /cas-management.properties:



En este archivo se procede a configurar la dirección de red (Nombre o IP) en la cual se encuentra atendiendo el servidor CAS, así como el puerto de manera que el servicio del módulo CAS-Management pueda interactuar con él sin inconvenientes. Por otra parte, también se procede a configurar unas líneas relacionadas con la persistencia de datos que luego serán explicadas en el siguiente apartado.

Como parte de la definición del contexto también se debe actualizar el archivo **./src/main/webapp/WEB-INF/managementConfigContext.xml** para permitir que el módulo de gestión de servicios se integre con el servidor CAS utilizando las mismas definiciones de servicios.

En aquellos casos en los cuales la persistencia de información de servicios sea en memoria, no hace falta actualizar este archivo, pero para el caso de servicios persistentes si será necesario como lo veremos más adelante.

En lo que respecta al acceso a la consola del módulo CAS-Management, el mismo es a través de la propia autenticación con el servidor CAS, pero la

autorización de los usuarios se establece editando el archivo **user-details.properties** según el siguiente detalle.

Luego de realizar estas actualizaciones en los archivos básicos de contexto se procede a construir el modulo CAS-Management mediante el comando Maven siguiente:

```
[root@casserver cas-management-webapp]# pwd
/root/cas-4.0.1/cas-management-webapp
[root@casserver cas-management-webapp]# mvn clean package
```

Como resultado del proceso y luego de la construcción exitosa, recibiendo la leyenda BUILD SUCCESS, obtenemos el archivo de salida **cas-management.war.**

Este archivo es un deploy el cual debe ser movido a la ruta desde donde el servidor Tomcat realiza la carga de componentes, de la misma forma que lo hemos realizado con el servidor CAS. En nuestro caso quedará localizado en la siguiente ruta:

```
[root@casserver target]# locate cas-management.war
/var/lib/tomcat/webapps/cas-management.war
[root@casserver target]#
```

Como última acción para que el módulo sea finalmente cargado, se requiere realizar un reinicio del servidor Tomcat, como lo hemos realizado en varias ocasiones en el presente trabajo.

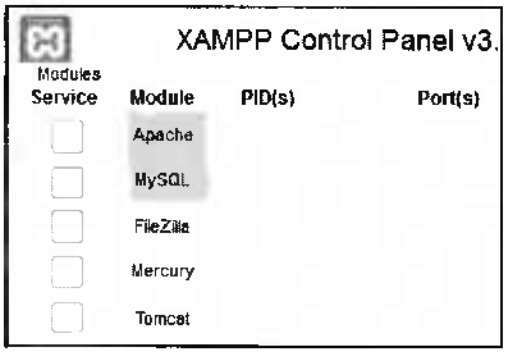
Persistencia de Datos con CAS-Management:

La configuración de la persistencia de Datos para los servicios accedidos ha resultado una tarea muy compleja derivado de la falta de documentación relacionada con un Servidor CAS; si bien existe documentación, la misma es incompleta, tiene errores y no describe claramente una situación de integración con versiones de cada componente de la solución.

Hay distintos aspectos de configuración a validar, en primer lugar, hay que agregar un componente a la solución que será el repositorio de los datos de los servicios. En nuestro caso, aprovechando que el módulo con soporte Apache

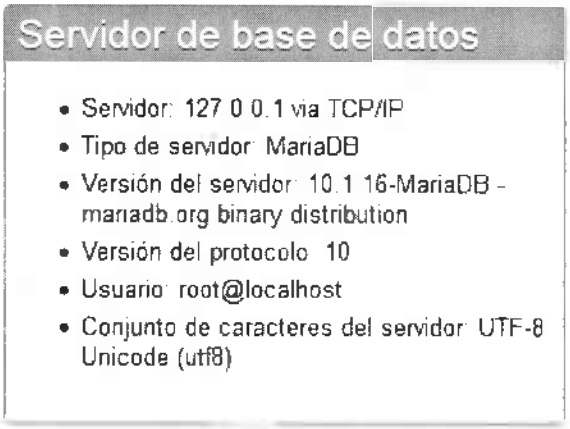
instalado como parte de la solución de pruebas incluye una versión de MySQL, activamos esta característica para iniciar los trabajos en nuestra base que actuará como repositorio. La instalación de MySQL se realizó de forma conjunta con el servidor Apache en la PC que actuará como servidor de aplicaciones y a su vez como interface final de usuario; en la arquitectura podrá ser visualizada como localhost o bajo la dirección IP de red 10.0.2.50. Para esta instalación simplemente se descarga el paquete XAMPP y mediante una simple interfaz como cualquier instalador Windows se continúa por las distintas etapas teniendo como precaución seleccionar las opciones de:

Opciones de Instalación en el módulo XAMPP



Para mayores detalles del paquete XAMPP se puede consultar la sección Integración con PHP. A continuación se detallará una imagen sobre la versión instalada de base de datos MySQL incluida en este paquete:

Versión de MySQL instalada en el presente trabajo

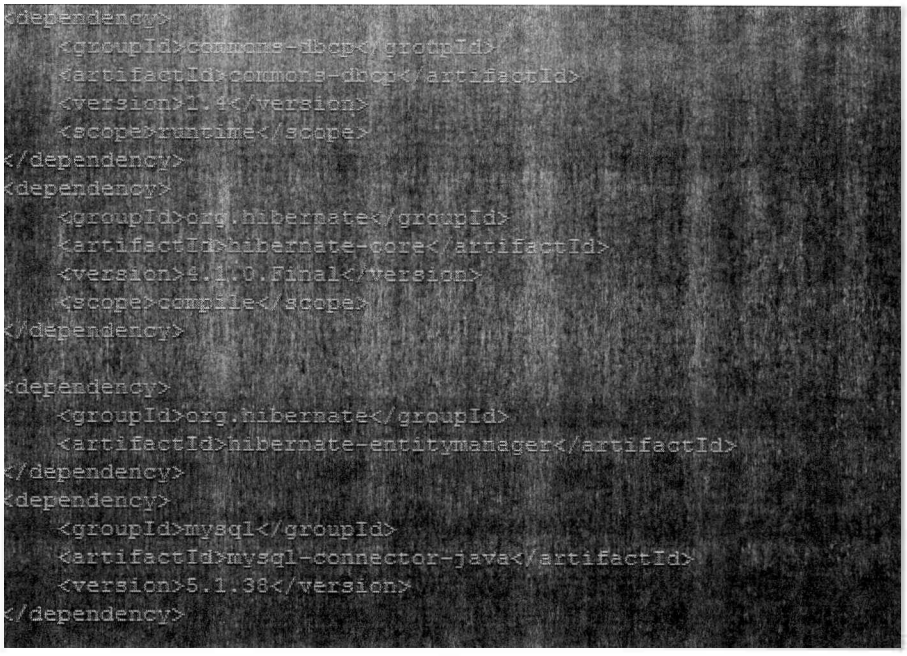


Si bien este servidor es el motor de base de datos, restan unos pasos iniciales para dejarlo utilizable. En primer lugar, se procederá a crear la base de datos en el motor MySQL recientemente instalado. En esta base se almacenarán por un lado información de configuración definida durante la construcción de la aplicación CAS-Management (responsable de la gestión de servicios CAS), y por otra parte, cuando la aplicación CAS-Management esté operativa, será el repositorio central en el cual se soportarán las consultas de servicios y sus modificaciones.

En nuestro caso hemos creado una base llamada ***services*** y un usuario llamado ***services*** también.

Contando con la base de datos y un usuario accesible, pasaremos a mencionar todos los archivos de configuración que fue necesario modificar para permitir la persistencia de datos.

En primer lugar, se debió modificar el archivo ***pom.xml*** (este archivo como se indicó anteriormente es el que guía la construcción del módulo cas-management cuando se utiliza la herramienta Maven) localizado en: /root/cas-4.0.1/cas-management-webapp y /root/cas-4.0.1/cas-server-webapp. La modificación consistió en incorporar las siguientes líneas:



```
<dependency>
    <groupId>commons-dbcp</groupId>
    <artifactId>commons-dbcp</artifactId>
    <version>1.4</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>3.3.0.Final</version>
    <scope>compile</scope>
</dependency>

<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.38</version>
</dependency>
```

Luego se procedió a actualizar los archivos ***cas-management.properties*** y ***cas.properties*** que se encuentran localizados en las rutas ***cas-management-/webappsrc/main/webapp/WEB-INF/ y cas-server-webapp/src/main/webapp/WEB-INF.***

En estos archivos se incorporaron las siguientes líneas:

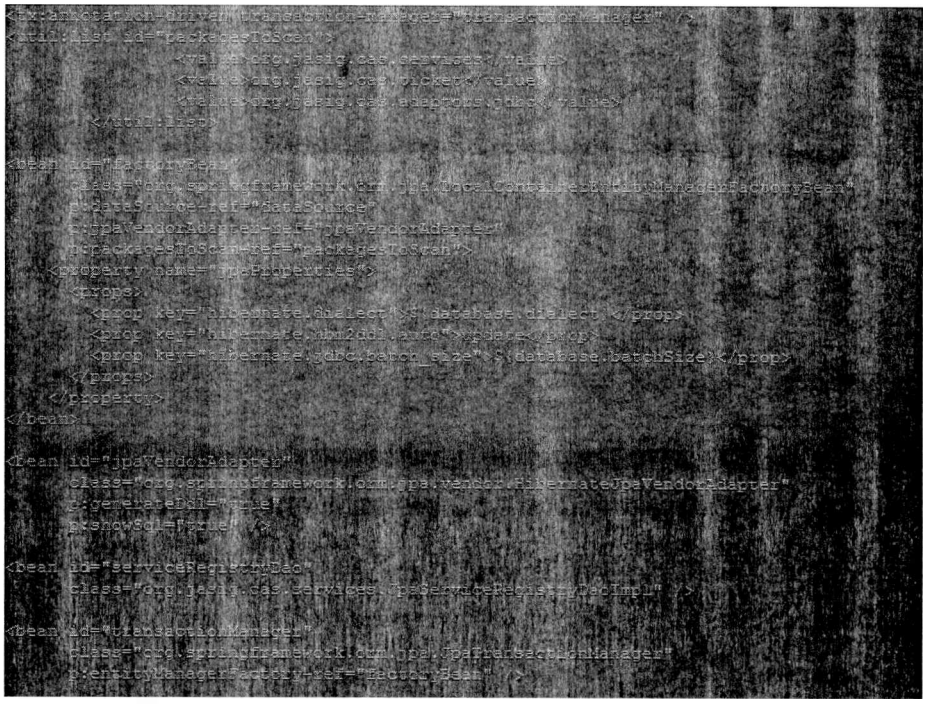
```
database.hibernate.dialect=org.hibernate.dialect.MySQLDialect
database.dialect=org.hibernate.dialect.MySQLDialect
database.batchSize=10
```

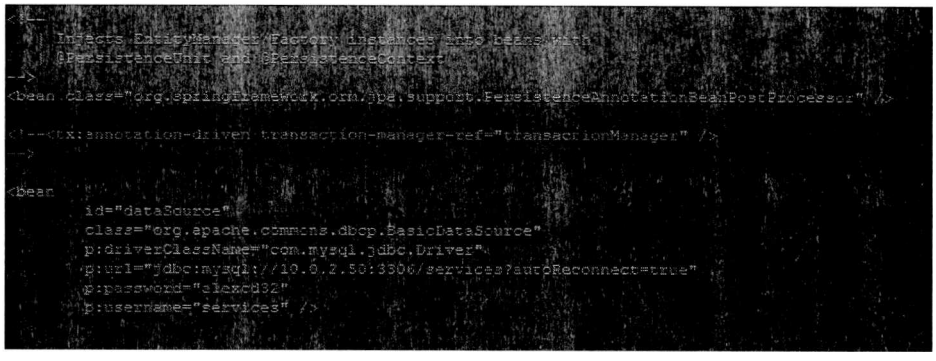
Luego de estas modificaciones fue necesario proceder a configurar los servicios con persistencia de datos tanto en el servidor CAS como en el módulo de CAS Management, ya que ambos deben tener la misma configuración y usar el mismo repositorio de servicios. Bajo esta premisa se procedió en primer lugar a adaptar el servidor CAS para el soporte de persistencia modificando los siguientes archivos:

Archivo: deployerConfigContext.xml

Localizado en: /root/cas-4.0.1/cas-server-webapp/src/main/webapp/WEB-INF

Se agregaron las siguientes líneas:





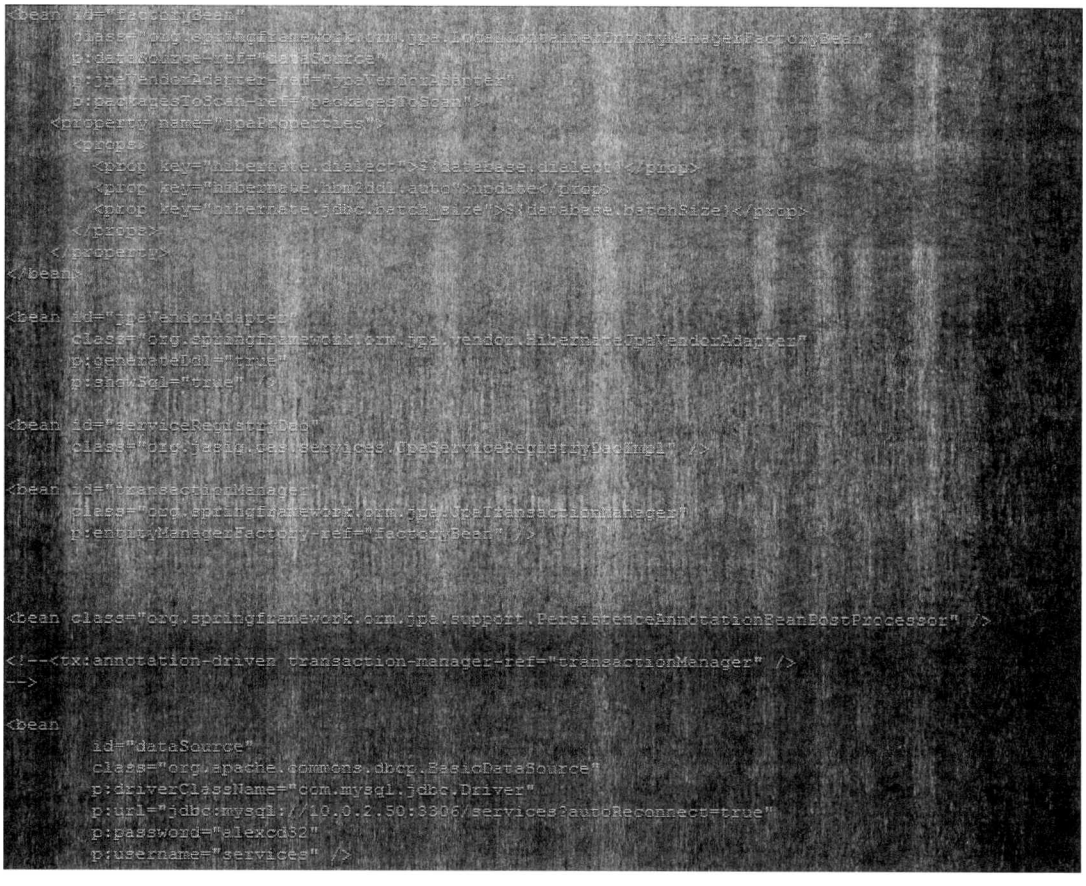
En este archivo se puede observar por un lado la definicion de los servicios y por otro de que manera se establece la configuración correspondiente al datasource definiendo: IP del servidor, puerto, nombre de base y otros datos adicionales.

Esta misma configuración luego debió ser replicada en el siguiente archivo para que el modulo de CAS-Management pueda estar alineado con la configuración de servicios en el servidor CAS.

Archivo: managementConfigContext.xml

Localizado en: ./cas-management-webapp-4.0.1/WEB-INF/

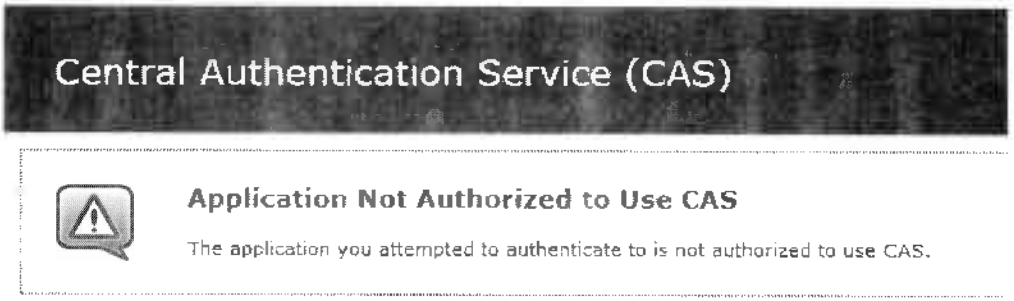
Se agregaron las siguientes líneas:



Finalmente luego de la construcción, depuración de errores, movimiento del deploy (.war) al home del servidor Tomcat y reiniciar los servicios involucrados, se observaron inconvenientes relacionados con los permisos y la no existencia de un servicio relacionado con la consola CAS-Management en el repositorio de

servicios. Al intentar acceder al modulo de CAS-Management se observaba el siguiente error:

Error ante inexistencia de servicio autorizado en la plataforma CAS



Luego de revisar la documentación se observó que este comportamiento se debe a un registro faltante en la tabla "registeredserviceimpl" (en el repositorio de servicios-base "services"). Este registro faltante es el que permite autorizar el ingreso a una url ya sea especificando la url de forma completa o utilizando un patron parecido a la dirección de la consola de administración como ser: "cas-management".

Luego de revisar los logs de la aplicación los cuales son principalmente en este caso **cas.log y cas-management.log** según se detalla a continuación:

Se pudo observar que el servidor CAS consultaba al repositorio y mencionaba la no existencia del servicio al cual se estaba accediendo según los detalles en el archivo **cas.log**:

Al observar esta situación se procedió a revisar la documentación y entender como se componían los campos de la tabla relacionada para ver como se podría generar un registro capaz de brindar la validez inicial del servicio de CAS-Management con el objetivo de luego poder administrar los distintos servicios según sea necesario.

A continuación el detalle de campos de la tabla según documentación:

- "ID (`id`)
- Name (`name`)
- Description (`description`)
- Service URL (`serviceId`) .
- Theme Name (`theme`).
- Enabled (`enabled`).
- SSO Participant (`ssoEnabled`)
- Anonymous Access (`anonymousAccess`)
- Allowed to Proxy (`allowedToProxy`).
- User Attributes (`allowedAttributes`).
- Ignore Attribute Management (`ignoreAttributes`).
- Evaluation Order (`evaluationOrder`).
- Username Attribute (`usernameAttribute`).
- Required Handlers (`requiredHandlers`).
- Attribute Filter (`attributeFilter`).
- Logout Type (`logoutType`)" [8]

Con la información de campos esperados y analizando la estructura de la tabla registeredserviceimpl existente en la plataforma, la cual se detalla campo por campo a continuación, se complementó la situación actual:

Tabla registeredserviceimpl del módulo CAS-Management

☐	1	**expression_type**	varchar(15)	No	ant	
☐	2	**id**	bigint(20)	No	*Ninguna*	AUTO_INCREMENT
☐	3	**allowedToProxy**	tinyint(1)	No	*Ninguna*	
☐	4	**anonymousAccess**	tinyint(1)	No	*Ninguna*	
☐	5	**description**	varchar(255)	No	*Ninguna*	
☐	6	**enabled**	tinyint(1)	No	*Ninguna*	
☐	7	**evaluation_order**	int(11)	No	*Ninguna*	
☐	8	**ignoreAttributes**	tinyint(1)	No	*Ninguna*	
☐	9	**name**	varchar(255)	No	*Ninguna*	
☐	10	**required_handlers**	longblob	Sí	*NULL*	
☐	11	**serviceId**	varchar(255)	No	*Ninguna*	
☐	12	**ssoEnabled**	tinyint(1)	No	*Ninguna*	
☐	13	**theme**	varchar(255)	Sí	*NULL*	
☐	14	**username_attr**	longtext	Sí	*NULL*	

A continuación se procedió a ejecutar una operación de insert, la cual termino agregando el siguiente registro para resolver el inconveniente del registro faltante:

expression_ty	id	allowedToProx	[illegible]	description	enabled	[illegible]	[illegible]	name	serviceId	ssoEnabled	theme	[illegible]
ant	1	1	0	HTTP CAS MANAGEMENT	1	0	0	HTTP BASIC	[illegible]	1		

Luego de realizar esta operación, podemos observar que fue posible acceder a la consola de administración de servicios dando por finalizada esta parte de la instalación de la plataforma la cual resultó muy compleja.

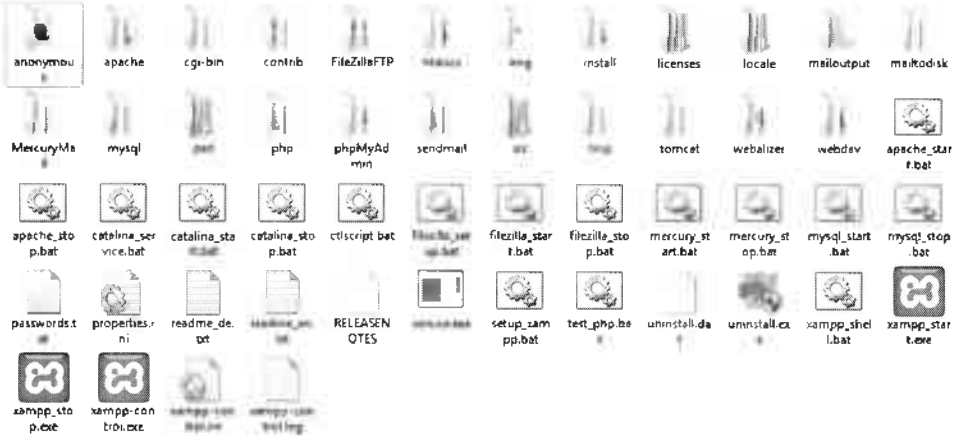
Esto se puede evidenciar a continuación donde se accede a la url ***https://casserver.tp.com:8443/cas-management/manage.html*** según el siguiente detalle:

Consola CAS-Management. Vista de Servicios.

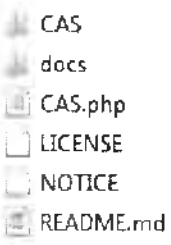
Habiendo arribado a este punto, ya nos encontramos con: el servidor CAS, los servicios LDAP, el servidor de monitoreo, y la integración con dicho servidor funcionando, así como también con la persistencia de datos implementada sobre el servidor CAS para el módulo de gestión de servicios del servidor CAS (CAS-Management Module). Solo nos resta mencionar como proceder con la instalación del servidor XAMPP de forma breve (ya que no es el objetivo de nuestro trabajo) y luego la integración a nivel PHP destacando la instalación de la API "PHP CASClient".

La instalación del servidor XAMPP ha resultado sencilla ya que únicamente se descarga el producto y se instala como cualquier aplicación para el sistema operativo Windows 7.

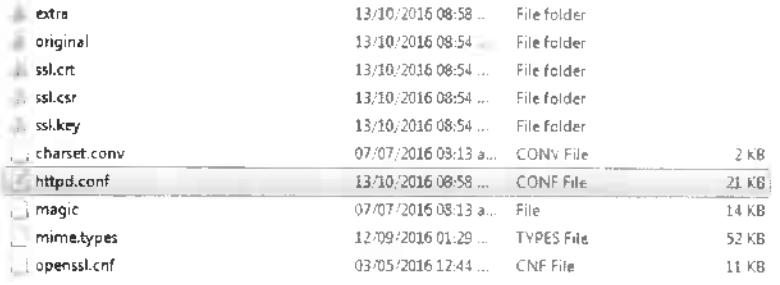
En nuestro caso se eligió alojar el producto en la unidad "C:", generando la siguiente estructura de directorios:



En lo que respecta a la configuración de la integración "PHPCAS Client", se procedió a descargar el paquete CAS-1.3.4 del sitio oficial de CAS, lo cual resultó en la siguiente estructura de directorio:



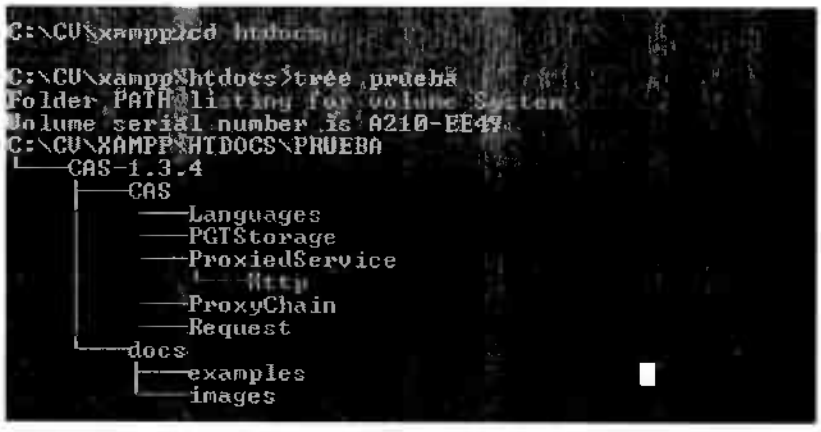
Al ser una aplicación PHP se procedió a revisar cual es el root de directorios del servicio apache de XAMPP, para lo cual se procedió a localizar el fichero httpd.conf



En este fichero se localiza la ruta relacionada con DocumentRoot:

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "C:/xampp/htdocs"
<Directory "C:/xampp/htdocs">
```

Luego se procede a mover el conjunto de directorios de la integración PHPCAS a esta ubicación de forma de que cuando se le pida la pagina al servidor web, este sepa donde ubicarla nuestro servidor web-php.



2.8 Desarrollo de la prueba de concepto y casos de uso básicos.

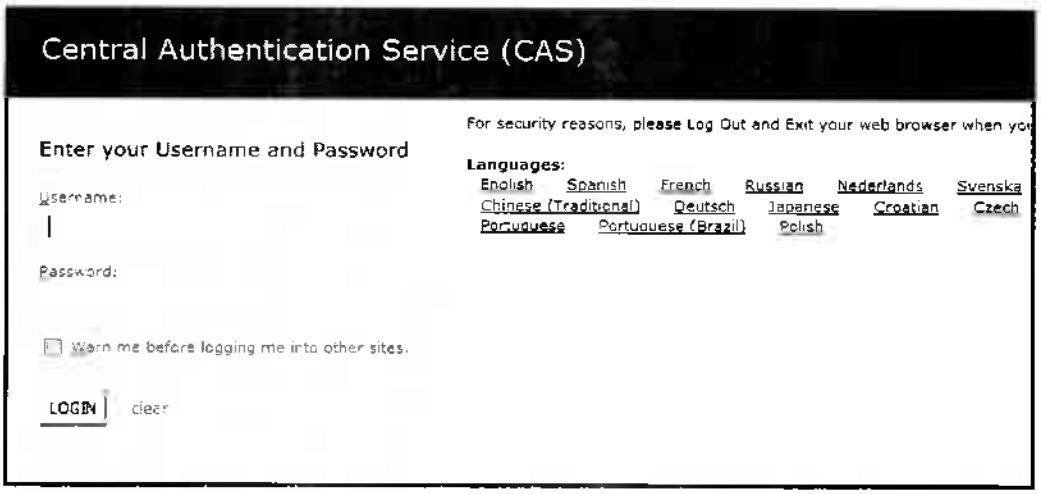
Teniendo presente la arquitectura implementada, pasaremos a realizar distintas pruebas para comprobar la respuesta del servidor CAS según los siguientes casos:

2.8.1 Usuario con contraseña incorrecta y servicio habilitado.

Empezando por el caso de uso "2.8.1 (2.8.1 Usuario con contraseña incorrecta y servicio habilitado.)" se utilizará el usuario "flavio" intentando acceder al servicio de consola "cas-management" a través de la url https://casserver.tp.com:8443/cas-management. Como se puede observar en la siguiente pantalla este servicio se encuentra habilitado según la lista de servicios configurados en el servidor CAS:

Al ser un servicio habilitado el servidor CAS el usuario final es redireccionado hacia la pantalla de login del sistema de authenticación en la cual se le requerirá usuario y password:

Consola de login inicial a los servicios CAS



Por otra parte, en el servidor se procede a generar el correspondiente ticket para permitir el login según el siguiente detalle.

Archivo cas.log del servidor CAS:

2016-11-19 16:19:36,761 DEBUG [org.jasig.cas.web.flow.InitialFlowSetupAction] - Placing service in FlowScope: https://casserver.tp.com:8443/cas-management/j_spring_cas_security_check

2016-11-19 16:19:36,763 DEBUG [org.jasig.cas.web.flow.GenerateLoginTicketAction] - Generated login ticket LT-5-rcZZKWncBGwhZQqyfQYHsaKWWpVCD3-cas01.example.org

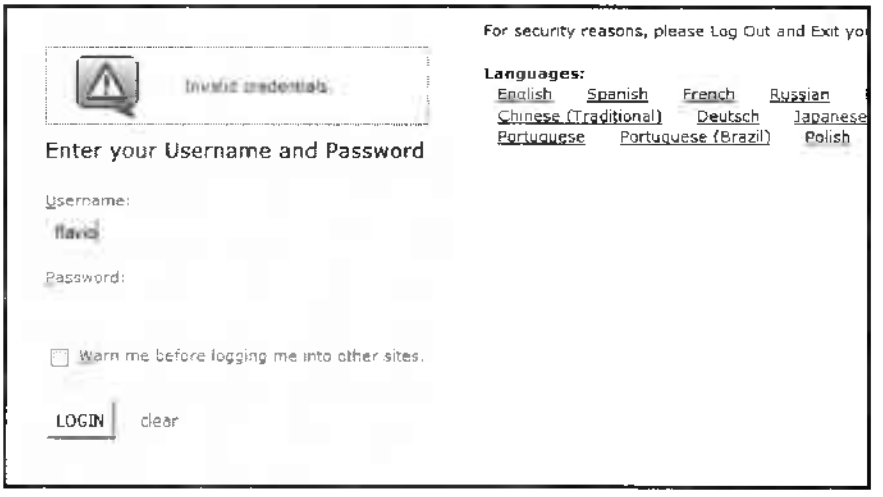
En el lado cliente podemos consultar las cookies generadas (en nuestro caso bajo la herramienta Firefox) según el siguiente detalle:

Al intentar ingresar con una contraseña incorrecta observamos que del lado servidor se reporta una falla al utilizar el modo de autenticación LDAP que tiene como backend el servidor CAS y por otro lado también se informa que no fue posible generar el ticket llamado "Granting Ticket" que hubiese permitido el acceso a la URL requerida.

Esta información es detallada en el log del servidor CAS donde también se pueden observar las direcciones IP y el usuario ingresado durante el acceso, lo cual puede ser utilizado posteriormente con propósitos de auditoria.

A su vez del lado cliente se observa la siguiente pantalla:

Error de Credenciales en página inicial de Login.



2.8.2 Usuario con contraseña correcta y servicio no habilitado.

En este caso de uso procederemos a tratar de acceder a un servicio no habilitado para utilizar el soporte de autenticación CAS (no está incluido el servicio perteneciente a la aplicación example_simple.php). Sin embargo, la aplicación final si tiene configurada la integración con el servidor cas por lo cual intentará realizar el ingreso.

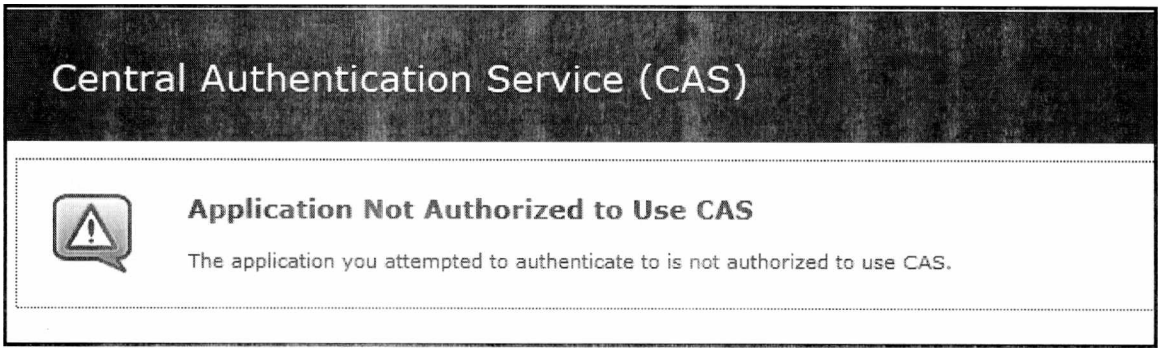
En primer lugar, invocamos desde el lado cliente a la url:

http://10.0.2.50/prueba/CAS-1.3.4/docs/examples/example_simple.php

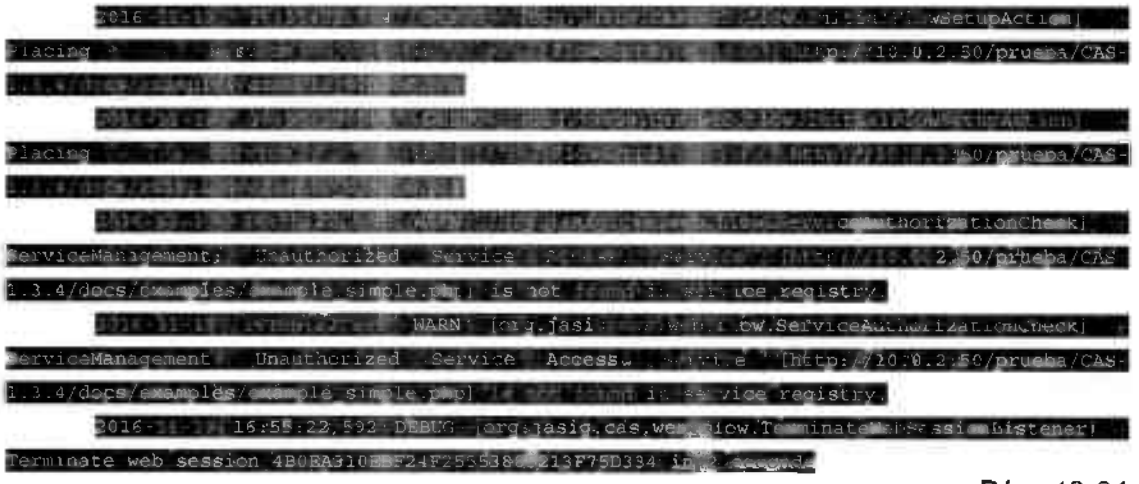
Dicha página está configurada para utilizar el antes mencionado cliente de cas para PHP según el siguiente detalle:

```
// Load the settings from the central config file
require_once 'config.php';
// Load the CAS lib
require_once $phpcas_path . '/CAS.php';

// Enable debugging
phpCAS::setDebug();
// Enable verbose error messages. Disable in production!
phpCAS::setVerbose(true);

// Initialize phpCAS
phpCAS::client(CAS_VERSION_2_0, $cas_host, $cas_port, $cas_context);
```

Como resultado de la invocación observamos del lado cliente la siguiente pantalla:



Y del lado servidor consultando nuevamente al log observamos lo siguiente:



En este caso se evidencia que el datasource definido (repositorio MySQL de soporte a los servicios CAS) no tiene disponible el servicio

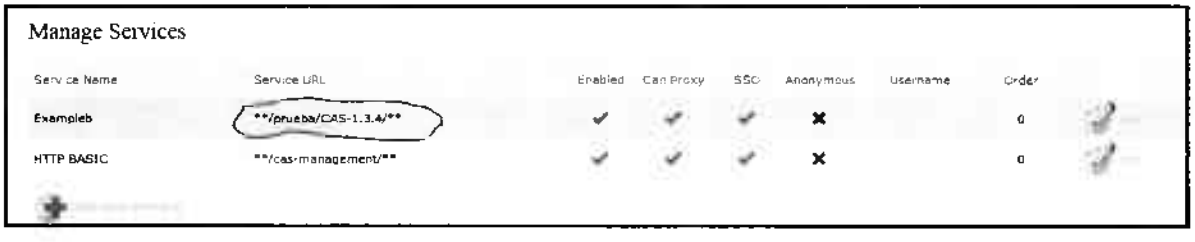
"http://10.0.2.50/prueba/CAS-1.3.4/docs/examples/example_simple.php".

Esto nos da la pauta de que por un lado el servidor recibió el request, y por otro el flujo de comportamiento dado al consultar a través del datasource (según los servicios definidos) no encontró una coincidencia por lo que denegó el acceso incluso terminando la sesión web.

2.8.3 Usuario con contraseña correcta y servicio habilitado.

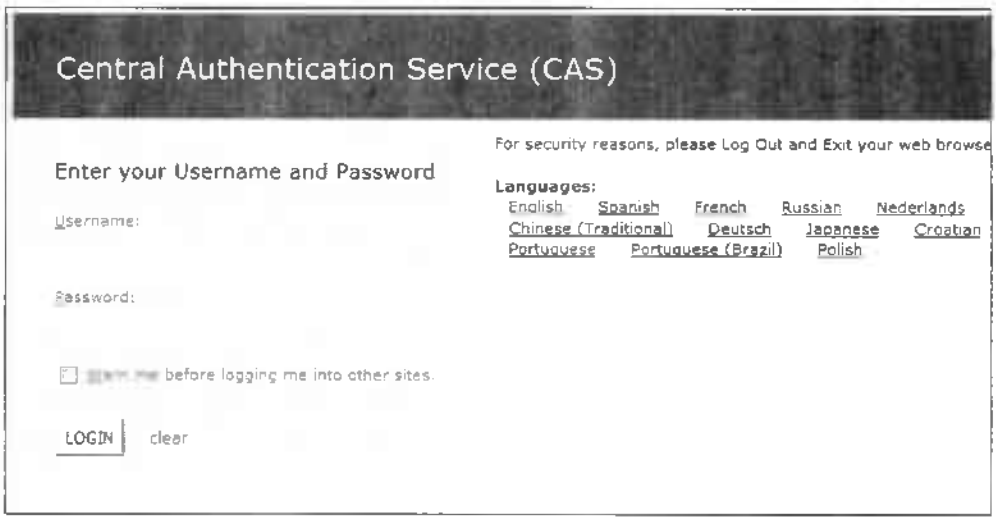
Ahora en tercer lugar procederemos a intentar ingresar con un usuario y password válidos a la url http://10.0.2.50/prueba/CAS-1.3.4/docs/examples/example_simple.php, la cual anteriormente no había permitido el acceso. Para que esto sea posible, debemos en primer lugar proceder a agregar el servicio al entorno CAS (mediante la herramienta CAS Management) dejando la nueva línea indicada a continuación:

Consola de Gestión de Servicios del módulo CAS Management



Como resultado de la invocación inicial observamos del lado cliente la presentación de la pantalla de Login en la cual ingresaremos el usuario "flavio" y la password correcta.

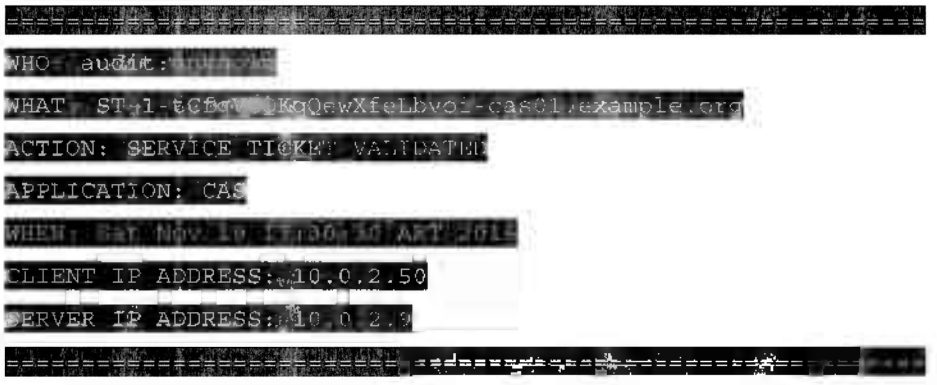
Ventana inicial de login del servidor CAS.



En primer lugar del lado servidor observamos que se genera el ticket correspondiente al servicio de login según el siguiente detalle:

```
2016-11-19                    :18:35                    DEBUG
[org.jasig.cas.web.flow.InitialFlowSetupAction]    - Placing service in
FlowScope                              https://                    :8443/cas-
management/j_spring_cas_security_check
2016-11-1                                                        DEBUG
[org.jasig.cas.web.flow.InitialFlowSetupAction]    - Placing service in
FlowScope:                              https://casserver.tp.com:8443/cas-
management/j_spring_cas_security_check
2016-11-19                    17:18:35,839                       DEBUG
[org.jasig.cas.web.flow.GenerateLoginTicketAction]    - Generated login
ticket LT-10-5bTBucacZVnJ31ZqjuFiEThwuTA93k-cas01.example.org
2016-11-19                    17:18:35,839                       DEBUG
[org.jasig.cas.web.flow.GenerateLoginTicketAction]    - Generated login
ticket LT-10-5bTBucac                1ZqjuFiEThwuTA                 01.example.org
```

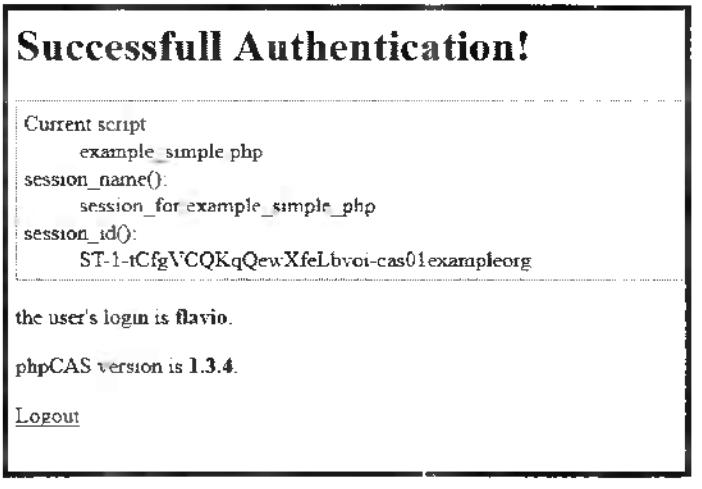
Entonces procedemos a ingresar el usuario y password para confirmar el acceso correcto. Luego de esta acción, observamos por el lado del servidor la acción para validar el servicio considerando el cliente, el destino y el servicio involucrado:



Finalmente, del lado cliente se observa la devolución del resultado del acceso a la página:

http://10.0.2.50/prueba/CAS-1.3.4/docs/examples/example_simple.php, la cual devuelve una serie de indicadores relacionados con el acceso al servidor CAS:

Ejemplo de aplicación en PHP haciendo uso de API PHP y servicio CAS

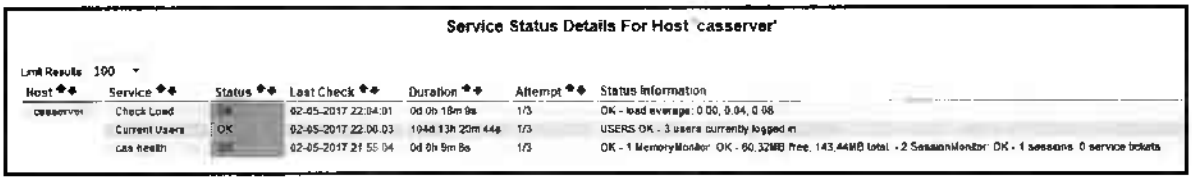


2.8.4 Respuesta del sistema de monitoreo del servidor CAS.

Con respecto a este caso, procedemos en primer lugar a verificar el estado del sistema de monitoreo cuando el servicio CAS se encuentra ok. Para esto, en primer lugar, procedemos a ingresar a la consola de Nagios accediendo a la url: http://10.0.2.16/nagios/ e ingresando con el usuario nagiosadmin.

Posteriormente consultamos el estado de servicios donde podemos observar los distintos chequeos configurados:

Estado de Alarmas del Servidor CAS en Nagios.



Service Status Details For Host 'casserver'

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
casserver	Check Load	[redacted]	02-05-2017 22:04:01	0d 0h 18m 9s	1/3	OK - load average: 0.00, 0.04, 0.08
	Current Users	OK	02-05-2017 22:00:03	104d 13h 20m 44s	1/3	USERS OK - 3 users currently logged in
	cas health	[redacted]	02-05-2017 21:55:04	0d 0h 5m 8s	1/3	OK - 1 MemoryMonitor OK - 60.32MB free, 143.44MB total. - 2 SessionMonitor OK - 1 sessions, 0 service tickets

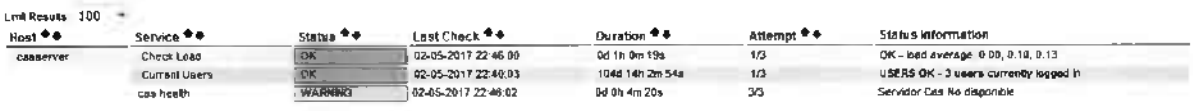
Si bien existen chequeos generales de sistema operativo y monitoreo de actividad y carga de CPU, a nosotros nos interesa particularmente la respuesta de salud del servidor CAS identificada como servicio "cas health".

Nos dirigimos al servidor CAS y provocamos una caída del servidor Tomcat para ver el comportamiento del servidor de monitoreo ante este evento. Esta acción la realizamos mediante el siguiente comando:

```
[root@casserver ~]# systemctl stop Tomcat.service
```

Como resultado podemos observar que en el servidor CAS ya no se encuentra disponible.

Vista alarmada del Servidor CAS en Nagios.



Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
casserver	Check Load	OK	02-05-2017 22:46:00	0d 1h 0m 19s	1/3	OK - load average: 0.00, 0.10, 0.13
	Current Users	OK	02-05-2017 22:40:03	104d 14h 2m 54s	1/3	USERS OK - 3 users currently logged in
	cas health	WARNING	02-05-2017 22:46:02	0d 0h 4m 20s	3/3	Servidor Cas No disponible

Evaluación del Monitoreo LDAP ante caídas del servidor LDAP.

En este punto la idea es realizar un caso de uso en el cual podamos simular una caída del Servidor LDAP y comprobar el funcionamiento del sistema de monitoreo, recibiendo las alertas relacionadas con la falla en el servidor LDAP.

En primer lugar, procederemos a mostrar una imagen de cómo se encuentra el servidor LDAP según nuestro sistema de monitoreo previamente a las acciones para impactarlo:

Vista Ok del Servidor de LDAP en Nagios.

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
casserver	Check Load	OK	02-06-2017 00:45:50	0d 0h 6m 48s	1/3	OK - load average: 0.00, 0.01, 0.05
	Current Users	OK	02-06-2017 00:40:01	104d 16h 3m 10s	1/3	USERS OK - 3 users currently logged in
	cas health	OK	02-06-2017 00:45:58	0d 0h 6m 40s	1/3	OK - 1.MemoryMonitor: OK - 48,95MB free, service tickets
	cas ldap	OK	02-06-2017 00:46:14	0d 0h 11m 22s	1/3	OK - Servidor Ldap

En la imagen se observa el estado OK para el servicio CAS LDAP. Comprobaremos esto realizando una consulta a nuestro servidor LDAP a nivel local en el sistema operativo:

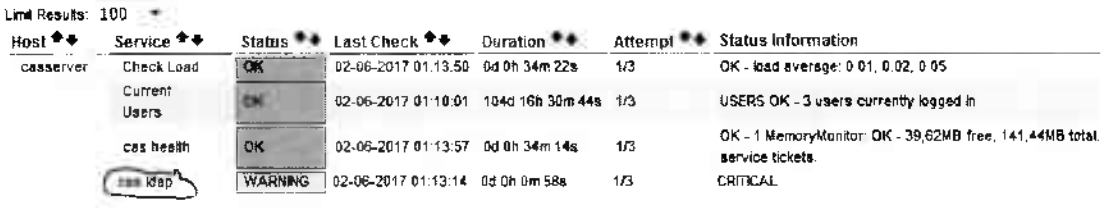
```
dn: ou=People,dc=casserver,dc=tp,dc=com
objectClass: organizationalUnit
ou: People

# flavio, People, casserver.tp.com
dn: uid=flavio,ou=People,dc=casserver,dc=tp,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
sn: Linux
userPassword:: e1NTSEF9...
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/flavio

# search result
search: 2
result: 0 Success

# numResponses:
# numEntries:
```

A continuación, procedemos a realizar un stop del servicio de LDAP mediante la siguiente acción:

```
root@casserver plugins]# systemctl stop slap
```

Mediante esta acción podemos observar como el sistema de monitoreo toma conocimiento de esta afectación detallando el estado warning del servidor LDAP.

Vista alarmada del Servidor de LDAP en Nagios.



Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
casserver	Check Load	OK	02-06-2017 01:13:50	0d 0h 34m 22s	1/3	OK - load average: 0.01, 0.02, 0.05
	Current Users	OK	02-06-2017 01:10:01	104d 16h 30m 44s	1/3	USERS OK - 3 users currently logged in
	cas health	OK	02-06-2017 01:13:57	0d 0h 34m 14s	1/3	OK - 1 MemoryMonitor: OK - 39,62MB free, 141,44MB total. service tickets.
	cas ldap	WARNING	02-06-2017 01:13:14	0d 0h 0m 58s	1/3	CRITICAL

En este caso, el agente nrpe ha ejecutado en varias oportunidades el comando de verificación del servidor LDAP, lo que permite saber en todo momento el estado y funcionamiento de las consultas de autenticación hacia el servidor.

En el apéndice se podrá observar el contenido de este comando, así como detalles del directorio. Lo que para la solución particularmente interesa, es la disponibilidad del servidor y el acceso a las consultas de la OU "People".

2.9 Interacciones de red con el servidor CAS (Foco - LDAP)

En este apartado la idea es mostrar a grandes rasgos los detalles sobre las interacciones a nivel de red y de qué forma se realizan los distintos requests y se emiten las respuestas relacionadas. Se brindarán detalles de protocolos y puestos considerados, así como también ejemplos capturados.

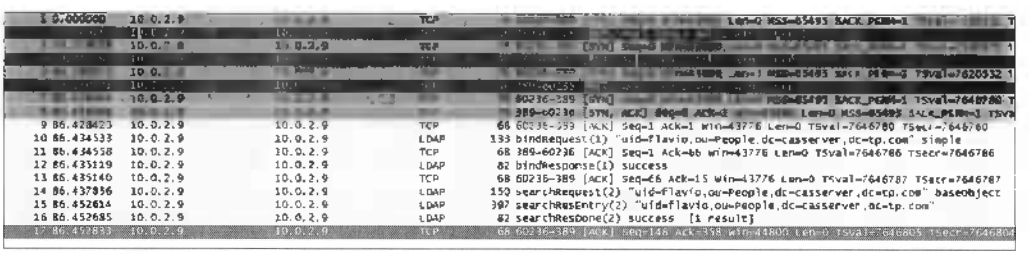
Para este análisis se ha utilizado la herramienta tcpdump; y para la visualización del tráfico la herramienta WireShark, la cual es una herramienta muy útil a la hora de evaluar tráfico de red.

Con respecto a la interacción contra el servidor LDAP, hemos procedido a activar la escucha en el puerto 389 mediante el siguiente comando:

```
[root@casserver ~]# tcpdump -i any port 389 -w dump1.t
```

Como resultado obtenemos un archivo que al abrirlo con la herramienta Wireshark no brinda un detalle completo de la interacción LDAP; si bien todo surgió de un request del lado cliente al solicitar una página e ingresar su usuario y contraseña, en ningún momento se ve un request del cliente al servidor LDAP, sino que todas las interacciones son locales entre el servidor CAS y su backend LDAP. Podemos ver de manera general que el request iniciado por el usuario final termina en una consulta LDAP iniciada por el servidor CAS.

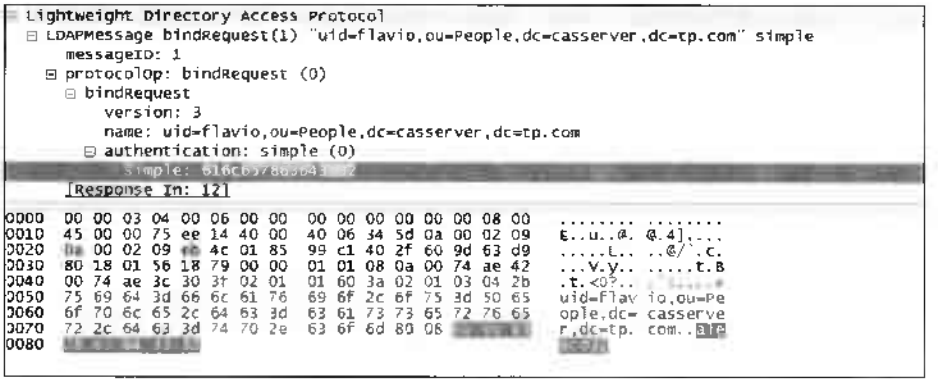
Vista detallada desde la herramienta Wireshark del intercambio LDAP.



9	86.428423	10.0.2.9	10.0.2.9	TCP	66 60236→389 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=7646780 TSecr=7646760
10	86.434533	10.0.2.9	10.0.2.9	LDAP	133 bindRequest(1) "uid=flavio,ou=People,dc=casserver,dc=tp,com" simple
11	86.434558	10.0.2.9	10.0.2.9	TCP	66 389→60236 [ACK] Seq=1 Ack=68 Win=43776 Len=0 TSval=7646786 TSecr=7646786
12	86.435129	10.0.2.9	10.0.2.9	LDAP	82 bindResponse(1) success
13	86.435140	10.0.2.9	10.0.2.9	TCP	66 60236→389 [ACK] Seq=68 Ack=15 Win=43776 Len=0 TSval=7646787 TSecr=7646787
14	86.437856	10.0.2.9	10.0.2.9	LDAP	150 searchRequest(2) "uid=flavio,ou=People,dc=casserver,dc=tp,com" baseobject
15	86.452614	10.0.2.9	10.0.2.9	LDAP	397 searchResEntry(2) "uid=flavio,ou=People,dc=casserver,dc=tp,com"
16	86.452685	10.0.2.9	10.0.2.9	LDAP	82 searchResDone(2) success [1 result]
17	86.452833	10.0.2.9	10.0.2.9	TCP	66 60236→389 [ACK] Seq=148 Ack=348 Win=44800 Len=0 TSval=7646805 TSecr=7646804

En la imagen a continuación se observa el resultado de explotar el bind request, donde se observa de qué manera la información en el cliente es utilizada por el servidor CAS para poder autenticar al usuario final (incluso transmitiendo la contraseña del mismo). El *bind request* es el mecanismo que es utilizado para autenticarse ante el servidor de directorio. En todo *bind request* se encontrarán la versión del protocolo, en nuestro caso 3, el usuario que estamos intentando autenticar y las credenciales asociadas.

Vista detallada desde la herramienta Wireshark del intercambio LDAP.



3 Conclusiones

La realización de este trabajo tenía como objetivo lograr desarrollar una arquitectura inicial que permita ofrecer servicios de SingleSignOn para plataformas web, y a la vez que tenga en cuenta las necesidades de monitorización que hoy en día son requeridas por las distintas organizaciones para poder mantener sus plataformas operando 24x7. Luego de varios meses de analizar la arquitectura, de revisar documentación de la cursada y de entender los distintos componentes que forman la solución, puedo decir que la tarea inicial ha sido cumplida y como fruto de este trabajo se cuenta con un entorno modelo para aplicar una solución de single-sign-on en cualquier organización. Claramente lo expuesto son sólo los componentes iniciales, y por otra parte se deja abierta la puerta a futuros trabajos sobre la tecnología CAS.

Lo que sí debo destacar, es que la configuración inicial ha tomado más trabajo del que inicialmente se pensaba dedicar derivado de complejidades en cuanto a la configuración, así como también falta de documentación precisa al realizar la integración. La complejidad fue incluso mayor al intentar persistir la información relacionada con servicios CAS en una base de datos, lo cual en cualquier entorno facilita su administración y permite que no se pierdan las configuraciones ante reinicios. Otro punto que ofreció complejidad adicional fue tratar de integrar el sistema de monitoreo debido a existencia de bloqueos de seguridad propios de Centos.

Con respecto al componente backend LDAP, una variante que podría simplificar este trabajo, sería utilizar una herramienta que sea más sencilla de configurar como backend. Openldap requiere contar con scripts y formateos para todas las acciones que uno desea realizar en cuanto a los usuarios. Al no contar con una interfaz gráfica, requiere conocimientos de sistemas operativos/scripting que no todos los administradores pueden llegar a conocer si se dedican a una tecnología en particular.

Existen muchas aplicaciones y futuros trabajos posibles sobre esta solución y CAS en general; por ejemplo, se podría explotar la obtención de eventos de auditoría para poder llevar a cabo tareas de control que permitan evaluar los

distintos incidentes de seguridad. También existe un amplio expectro para trabajos relacionados con la tecnología CAS, como ser: analizar las variantes de protocolos existentes o trabajar sobre los distintos backends posibles como Radius, MongoDB, etc.

4 Glosario

Término	Descripción
CAS	Herramienta integrada con soporte a Apache, Tomcat y MySQL Server
LDAP	Lightweight Directory Access Protocol (Protocolo Ligero-Simplificado de Acceso a Directorios). Es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.
Hibernate	Es un framework de java persistente utilizado para mapeos contra bbdd y facilitar la ejecución de consultas.
Beans	Podemos definirlo como un objeto que se encuentra en un contenedor y puede ser instanciado. Es un componente de software java reutilizable y que se presenta como una unidad en la interfaz de desarrollo java.
Maven	Es una herramienta para la gestión y construcción de aplicaciones Java. Tiene un modelo de configuración de construcción más simple, basado en un formato XML.
SingleSignOn	Es un procedimiento de autenticación que permite acceder a varios entornos de forma segura autenticándose en una

	única oportunidad.
NRPE	Es el cliente de nagios que permite la ejecución de comandos y plugins en el servidor siendo monitorizado.
OU	Unidad Organizativa dentro de un directorio LDAP que facilita la separación y agrupamiento de objetos.

5 *Apéndices y Anexos*

- Contenidos de Archivos de Configuración del Servidor CAS

Archivo de configuración principal del servidor CAS.	**Por cuestiones de tamaño estos archivos serán entregados en un CD/DVD.**
Archivo de Configuración de la conectividad con los Backends y los datasources en BBDD.	
Archivo de configuración de Nivel de Log.	
Archivo de configuración para la construcción de la plataforma via Maven.	
Archivo de Log del Servidor CAS.	

- Contenidos de Configuración del Servidor CAS-Management

Archivo de configuración principal del módulo CAS-Management	**Por cuestiones de tamaño estos archivos serán entregados en un CD/DVD.**
Archivo de configuración relacionado con los servicios y la persistencia de datos.	
Archivo de configuración para permitir la utilización de acceso a datos y servicios utilizando la opción package to scan.	
Archivos de utilidad para la configuración de la seguridad y acceso a la interfaz de administración de servicios	

- Contenidos de Configuración del Servidor Nagios

Archivos de definición de servicios y host monitorizados en el servidor CAS.	**Por cuestiones de tamaño estos archivos serán entregados en un CD/DVD.**
Archivos de configuración de los comandos para monitorizar el servidor LDAP y el Servidor CAS	
Archivos localizados en el directorio nagios/plugins del servidor CAS.	

- Contenidos de Configuración del Servicio PHP

Archivos de Configuración del módulo PHP Client que permite utilizar un servidor CAS como parte de la autenticación en una aplicación PHP.	**Por cuestiones de tamaño estos archivos serán entregados en un CD/DVD.**
Archivos de la Aplicación cliente PHP.	

- Contenido del Esquema de BBDD MySQL

Contenido de la base que da soporte a los servicios de CAS-Management y permite la persistencia de los datos.	**Por cuestiones de tamaño este archivo será entregado en un CD/DVD.**

6 *Bibliografía*

[1] Arquitectura General CAS. https://apereo.github.io/cas/4.1.x/
(Información consultada el 04-07-2016).

[2] Guía de Administración OpenLDAP,
http://www.openldap.org/doc/admin24/OpenLDAP-Admin-Guide.pdf
(Información consultada el 04-10-2016).

[3] Portal de Centos, https://wiki.centos.org
(Información consultada el 03-07-2016).

[4] Nagios Core, https://www.nagios.org/projects/nagios-core/
(Información consultada el 10-02-2017)

[5] Configure LDAP Server, http://www.server-world.info
(Información consultada el 10-11-2016).

[6] Arquitectura NRPE, https://support.nagios.com
(Información consultada el 10-02-2017)

[7] PHP CAS Library, https://wiki.jasig.org
(Información consultada el 04-07-2016).

[8] Service Management,
https://apereo.github.io/cas/4.0.x/installation/Service-Management.html
(Información consultada el 15-05-2017)

[9] Single Sign On y CAS, http://www.slideshare.net/barseghyanartur/sso-using-cas-twofactor-authentication-pygrunn-2014-talk
(Información consultada el 03-11-2016)

[10] Definición y generalidades del protocolo LDAP,
http://es.wikipedia.org/wiki/Protocolo_Ligero_de_Acceso_a_Directorios
(Información consultada el 04-10-2016).

[11] Guía de LDAP sobre RedHat y Centos, http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-LDAP.html
(Información consultada el 15-11-2016)

[12] RFCs de LDAP, estructura y atributos de un directorio de información, http://es.kioskea.net/contents/269-protocolo-LDAP
(Información consultada el 10-10-2016)

[13] Authenticacion LDAP,
http://www.ipcop.org/2.0.0/es/admin/html/proxy-auth-LDAP.html
(Información consultada el 15-11-2016)

[14] Uso de plugins en Nagios,
https://wiki.jasig.org/display/CASUM/Nagios+plugins
(Información consultada el 15-03-2017)

[15] Shon Harris, CISSP All in One, 5th Edition, Mc Graw Hill, USA,2013.