Universidad Nacional de Misiones. Facultad de Ciencias Exactas, Químicas y Naturales. Secretaría de Investigación y Postgrado. Maestría en Tecnología de la Información

Maestrando
Omar Martín ANTONIO

Gestor de autenticación de usuarios ´ centralizado para entornos de red heterogéneos.
Caso de Estudio: Tribunal Electoral de Misiones

Trabajo final de Maestría presentada para obtener el título de "Magíster en Tecnología de la Información"

"Este documento es resultado del financiamiento otorgado por el Estado Nacional, por lo tanto, queda sujeto al complimiento de la Ley N° 26.899".

Director
Dr. Marcelo Julio Marinelli
Co Director
Emanuel Irrazabal

Posadas, 2022

Universidad Nacional de Misiones

Facultad de Ciencias Exactas, Químicas y Naturales

Tesis de Maestría en Tecnologías de la Información

Gestor de autenticación de usuarios centralizado para entornos de red heterogéneos. Caso de Estudio: Tribunal Electoral de Misiones

Autor: Omar Martín Antonio
Director: Marcelo Marinelli
Codirector: Emanuel Irrazabal

Año 2022

A mi familia por el aguante en todos estos años de estudio. *Omar*

Agredecimientos

Agradezco la colaboración de mi director Marcelo Marinelli y codirector Emanuel Irrazabal. También a mis compañeros del Tribunal Electoral de Misiones por su colaboración en este trabajo, en especial a Mauricio Romanini por su empeño y esfuerzo. Gracias a la comunidad GNU/Linux.

Resumen

Con el creciente desarrollo de las tecnologías de la información y la popularidad de Internet, el uso de múltiples sistemas y aplicaciones informáticas se ha extendido considerablemente en las empresas y organismos públicos, constituyendo además un factor importante en el crecimiento de los mismos.

Dentro de este contexto, la confidencialidad e integridad de los datos es fundamental para asegurar el acceso a los recursos informáticos, donde la autenticación y autorización se destacan como dos factores claves para lograrlo. Sin embargo, esto trae aparejado otros tipos de problemas relacionados con la dificultad que los usuarios y administradores tienen para mantener, recordar y gestionar las credenciales de acceso a cada uno de los mencionados sistemas y aplicaciones.

Una solución a estos problemas es utilizar un sistema de inicio de sesión único o Single Sign-On (SSO). En un sistema SSO se comparte un servidor central de autenticación entre todas las aplicaciones, donde se almacenan las credenciales de acceso de todos los usuarios. Esto permite iniciar sesión una sola vez y mantener en cache las credenciales para autenticar automáticamente con las demás aplicaciones, quitándole un trabajo repetitivo al usuario.

Implementar un sistema de SSO no es una tarea sencilla debido a la heterogeneidad de aplicaciones y sistemas involucrados. En este sentido, no existe una solución o herramienta universal que se ajuste a todos los escenarios o situaciones.

En el presente trabajo, luego de un estudio detallado de los diferentes mecanismos, protocolos de autenticación y aplicaciones de gestión de identidad y acceso, se presenta una posible solución a este problema utilizando el protocolo Kerberos como base del sistema de SSO. Como caso de estudio se propone la red informática del Tribunal Electoral de la Provincia de Misiones (TEPM), representativo de un entorno de red heterogéneo.

El resultado final del trabajo es una aplicación que permite de manera sencilla y segura gestionar las credenciales de acceso de los usuarios en un servidor central de autenticación.

Palabras clave: Autenticación de Usuarios, Inicio de Sesión Único, Gestión de Identidad y Acceso, Kerberos, Entornos Heterogéneos

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Descripción

Por lo general, por cuestiones de seguridad, el uso de cualquier sistema informático involucra un paso previo donde se valida la identidad del usuario y los permisos que posee. Este paso previo se denomina control de acceso, y cada aplicación implementa su propio mecanismo de autenticación y autorización para ello. Los usuarios y administradores de sistemas informáticos conviven en un entorno heterogéneo de mecanismos de autenticación y autorización, con la consecuente dificultad para mantener, recordar y gestionar las credenciales de acceso a cada una de las aplicaciones que se utilizan [1].

La confidencialidad e integridad es un aspecto importante de la seguridad informática y dependen principalmente de la autenticación y autorización para resguardar el acceso a la información. El control de acceso se apoya en el proceso de autenticación para identificar y validar la identidad del usuario, así como en el proceso de autorización, para otorgar los permisos y privilegios necesarios para la utilización de los recursos informáticos.

Las organizaciones actuales requieren que los empleados accedan a múltiples sistemas informáticos para llevar adelante sus actividades diarias. La mayoría de los sistemas y aplicaciones validan la identidad del usuario utilizando la clásica combinación de "nombre de usuario" y "contraseña" (*password*). El empleado se ve obligado a memorizar las credenciales de acceso para cada sistema que utiliza. Esto termina fastidiando al usuario y, en la mayoría de los casos, llevándolo a utilizar contraseñas simples, fáciles de recordar, pero vulnerables a numerosos ataques, comprometiendo la seguridad general del sistema.

Si bien, la utilización del nombre de usuario y contraseña estática es vulnerable, sigue siendo el método de autenticación de usuario más ampliamente aceptado y utilizado [2], principalmente por su usabilidad frente a otros mecanismo teóricamente más seguros. Para reforzar la seguridad de las contraseñas estáticas, cada vez es más común complementar con un segundo factor de autenticación como las contraseñas de un solo uso One-Time Password (OTP), o las credenciales biométricas como las huellas dactilares o el reconocimiento facial.

Para la organización el uso de múltiples sistemas y aplicaciones implica gestionar individualmente las credenciales para cada uno de ellos, lo que conlleva a una mayor carga de trabajo y complejidad para el administrador de los sistemas. Para paliar con estos problemas se suelen utilizar sistemas de inicio de sesión único o SSO. Un sistema SSO elimina los inicios de sesión individuales, centralizando la autenticación de usuario y administración de identidad en un proveedor de identidad central [3].

En un estudio realizado por [4] se observa que dos de las principales causas del mal uso de las contraseñas se debe a: la utilización de múltiples contraseñas y la falta de concientización en los usuarios sobre la seguridad de los sistemas de información. Por ello se recomienda, en los lugares donde se trabaja con varios sistemas, que la autenticación a los mismos se realice con mecanismos de SSO y/o dispositivos seguros, como tokens criptográficos, para aliviar la memoria de los usuarios.

Existen diversas implementaciones de un sistema SSO. Estas implementaciones dependen principalmente del sistema operativo utilizado y del tipo de aplicaciones y servicios a integrar en el SSO. Entre los sistemas SSO más utilizados se puede nombrar a los sistemas que utilizan Kerberos como protocolo de autenticación. Tal es el caso del Active Directory de Windows y el FreeIPA disponible para sistemas Linux RedHat/Fedora.

Kerberos es un protocolo de autenticación de usuarios para entornos de red potencialmente inseguros, que permite integrar la autenticación de diversos servicios de red y proveer de un inicio de sesión único a través del uso de tickets de sesión [5]. Kerberos centraliza la gestión de usuarios mediante la utilización de un Centro de Distribución de Claves o Key Distribution Center (KDC). Cuando se utiliza Kerberos, el usuario al iniciar sesión en una computadora o host, recibe un ticket de sesión por parte del KDC, que se guardará localmente en caché para autenticar el acceso de futuras aplicaciones.

La experiencia del usuario y la seguridad son dos de los atributos de calidad más importantes para un sistema informático. En un esfuerzo por mejorar, las organizaciones intentan implementar un SSO que facilite la experiencia del usuario y mejore la gestión de la seguridad. El concepto de SSO no es nuevo y hay muchas soluciones que se utilizan; sin embargo, la realidad es que aunque existan varias configuraciones y tecnologías relacionadas al SSO que han estado disponible durante años, el tema en cuestión todavía presenta desafíos y dificultades para su implementación [6].

El desarrollo del presente trabajo se focaliza en los protocolos y mecanismos de autenticación de usuarios utilizados en los organismos y entidades públicas tomando como caso de estudio el TEPM. Luego de un estudio detallado de los mencionados mecanismos, se propone integrar la autenticación de los diferentes sistemas y aplicaciones en una solución centralizada que permita gestionar la

identidad de los usuarios de una manera simple y segura. Esta solución permite además implementar SSO para un entorno de red heterogéneo, como es el caso del TEPM, donde conviven aplicaciones de escritorio/web y sistemas operativos Linux y Windows.

El resultado final del trabajo es una aplicación que permite administrar los usuarios y sus credenciales de acceso a los diferentes sistemas del TEPM.

1.2. Motivación

En al actualidad, el desarrollo de aplicaciones web se ha extendido a todos los ámbitos, incluyendo, entre otras, a las aplicaciones de gestión interna para organismos de gobierno. Las aplicaciones nativas o de escritorio son cada vez menos frecuentes, y la mayoría de las actividades dentro de una computadora se realizan a través un navegador web o algún otro sistema distribuido del tipo cliente-servidor.

Tal es el caso del TEPM, que cuenta con un Sistema de Gestión Interno (SGI) web, donde se desarrollan las mayoría de la actividades operativas y de soporte del organismo:

Operativas:

- Mesa de Entradas
- Padrón de Extranjeros
- Partidos Políticos
- Proceso Electoral
- Notificaciones Electrónicas

Soporte:

- Sistema de Expedientes
- Archivo
- Inventario
- Personal
- Sueldos
- Sistema de Tickets
- Gestión de Usuarios

El sistema de gestión de usuarios actual del TEPM permite gestionar únicamente los usuarios y permisos del SGI.

Además del SGI, se utilizan otros sistemas como correo electrónico, gestor de contenidos web y archivos compartidos. Todos estos sistemas necesitan sus propias credenciales de acceso (usuario y clave) que el administrador de la red debe gestionar en forma separada para cada sistema. A esto se suman las credenciales para el acceso a los equipos informáticos (PCs, notebooks, equipos de networking) y los servicios de red (VPN, WiFi, 802.1X).

Servicios de red comunes como terminales virtuales, archivos compartidos, bases de datos, servidores web, correo electrónico, etc., se pueden configurar para utilizar Kerberos como mecanismo de autenticación (kerberizar). Sin embargo, la mayoría de las aplicaciones web de uso interno en las organizaciones, no se han diseñado para utilizar Kerberos como protocolo de autenticación.

Esto motiva a buscar una solución que integre dichas aplicaciones a un sistema SSO en base a Kerberos, y lograr con esto la posibilidad de implementar un gestor único de autenticación de usuarios.

1.3. Objetivos

El objetivo principal del presente trabajo es diseñar e implementar una aplicación que permita gestionar la autenticación de usuarios de manera centralizada en entornos de red heterogéneos. Como caso de estudio se propone la red informática del TEPM, donde existe una variedad de aplicaciones de tipo escritorio/web que se ejecutan sobre sistemas Linux y/o Windows.

Se deberá para ello considerar los siguientes objetivos específicos:

- Estudiar los conceptos, protocolos y mecanismos relacionados a la autenticación de usuarios, como ser: SSO, NTLM, Kerberos, LDAP, RADIUS, PAM, SASL, GSS-API, SSPI y SPNEGO.
- Analizar y comprender los mecanismos de autenticación que utilizan los sistemas informáticos típicos de un organismo de gobierno y su contexto.
- Proponer una solución para la gestión de autenticación centralizada de usuarios en entornos de red heterogéneos.
- Validar la propuesta con el desarrollo de un gestor de autenticación de usuarios centralizado para el organismo del caso de estudio.
- Realizar las pruebas de funcionamiento y seguridad del gestor de autenticación.

- Obtener las conclusiones del trabajo realizado y proponer posibles líneas de trabajo a futuro.

1.4. Metodología

Para lograr los objetivos planteados anteriormente se llevaron adelante los siguientes pasos:

- Estudio exploratorio bibliográfico sobre los protocolos y mecanismos de autenticación de usuarios. Mediante una revisión sistemática de la literatura se identificaron y analizaron artículos académicos, libros, estándares y otros reportes técnicos relacionados a IAM, SSO, Kerberos, LDAP, GSS-API y SPNEGO. Para ello, se utilizaron herramientas como buscadores de artículos académicos, repositorios y gestores de referencias bibliográficas.

- Estudio descriptivo del entorno del caso de estudio. Mediante la observación directa y recopilación de datos se obtuvo información de las diferentes variables que influyen en dicho entorno, como ser: actividades llevadas adelante por el organismo, sistemas informáticos utilizados que dan soporte a dichas actividades, protocolos y mecanismos de autenticación implementados en las aplicaciones, servicios e infraestructura de red presentes.

- Estudio descriptivo de herramientas para la gestión de identidad y acceso en entornos de red heterogéneos. A través del análisis y comparación de diferentes alternativas de Identity and Access Management (IAM) se determinaron las características de diseño necesarias para el gestor de autenticación centralizado de usuarios.

- Diseño de una arquitectura de autenticación centralizada de usuarios según los requerimientos del caso de estudio. Para ello, se tuvo en cuenta al análisis y comparación de alternativas de IAM del paso anterior y los trabajos de otros autores referidos al tema, con el fin de determinar la solución que mejor se ajustaba al caso de estudio. Una vez definida la arquitectura, se trabajó en la configuración y puesta a punto de los diferentes componentes del sistema. Los pasos utilizados se describen en el anexo A.

- Desarrollo e implementación de un gestor de usuarios centralizados para la arquitectura propuesta. Se investigó la forma de integrar un módulo de gestión de usuarios unificado al sistema de gestión interno del organismo, manteniendo los principios de diseño de una arquitectura Application Programming Interface (API) REpresentational State Transfer (REST).

Definido el lenguaje de desarrollo del gestor, se realizaron pruebas con diferentes librerías para lograr la conexión con los principales componentes de la arquitectura de autenticación propuesta. Luego, en la implementación se utilizó un modelo de desarrollo tradicional en cascada.

Parte del desarrollo del gestor de usuarios fue incorporar un mecanismo de SSO al SGI, para lo cual se definió una ruta especial de la API que utiliza un módulo de autenticación Kerberos para el servidor web Apache.

Para evaluar los resultados del nuevo sistema de autenticación se realizó una encuesta a los usuarios luego de un año de uso del mismo. En la encuesta se pidió a los usuarios que respondan preguntas comparando el nuevo sistema con el anterior según lo descripto en el anexo B.

Se realizaron pruebas de tareas típicas del sector de IT, sin el gestor centralizado de usuarios y con la utilización del mismo, con el fin de estimar los costos operativos del antes y después de la implementación del sistema unificado de autenticación.

Otra de las variables que se tuvo en cuenta fue el registro del sistema de tickets del TEPM, donde se identificaron solicitudes de los usuarios referidas a errores de acceso a los sistemas.

1.5. Estructura de la Tesis

El resto de la tesis está organizada de la siguiente manera: En el segundo capítulo se presenta una revisión bibliográfica de los conceptos, protocolos y mecanismos relacionados a la identificación y autenticación de usuarios. En el tercer capítulo se analiza el estado del arte de los mecanismos de autenticación en organismos de gobierno y se propone una solución para la gestión de autenticación de usuarios centralizada. En el cuarto capítulo se describe el desarrollo de una aplicación que permite gestionar la arquitectura de autenticación de usuarios presentada en el capítulo anterior. El quinto y último capítulo expone los resultados y conclusiones obtenidos de las pruebas de campo realizadas en el organismo de estudio al aplicar el gestor de autenticación de usuarios.

2. Identificación y Autenticación de Usuarios

En este capítulo se exponen los fundamentos de la identificación y autenticación de usuarios. El capítulo comienza con una definición de la Seguridad de la Información y una descripción de los términos Autenticación, Autorización y Trazabilidad. Luego, se detallan las diferentes credenciales de acceso, se clasifican los sistemas de inicio de sesión únicos y se describen los protocolos más ampliamente utilizados en la autenticación de usuarios para servicios e infraestructuras de red.

El aporte de conocimientos de esté capítulo será de vital importancia para analizar y comprender los mecanismos de autenticación utilizados en el caso de estudio y luego poder elaborar la solución de autenticación unificada pretendida.

2.1. Seguridad de la Información

La información es un activo y, como cualquier otro activo importante de una organización, debe estar debidamente protegido. La información se puede almacenar de muchas formas, incluyendo: formato digital (por ejemplo, archivos de datos almacenados en medios electrónicos u ópticos), formato material (por ejemplo, en papel o una pizarra), así como información en forma de conocimiento de los empleados. Esta información puede transmitirse por diversos medios, incluidos: mensajería, comunicación electrónica o verbal [7]. Sea cual fuera el tipo de información y el medio de transmisión utilizado, se debe proteger adecuadamente asegurando la Confidencialidad, Integridad y Disponibilidad de la información:

Confidencialidad Proteger que la información no este disponible o se divulgue a individuos, entidades o procesos no autorizados.

Integridad Proteger la información de modificaciones intencionales o accidentales que comprometan la validez de la misma.

Disponibilidad Característica que permite acceder y/o utilizar la información por entidades autorizadas en los momentos necesarios.

Por la tanto, se puede definir a la **seguridad de la información** como la preservación de estos tres principios o características básicas de la seguridad.

Para lograr la protección adecuada se deben fortalecer dichas características aplicando controles sobre los activos involucrados en el manejo de la información. Todos los activos de información presentan vulnerabilidades que pueden ser explotadas por una amenaza. Existe una cierta probabilidad de que la amenaza se materialice y afecte con un determinado impacto o daño sobre un activo. La probabilidad de ocurrencia de la amenaza y el grado de afectación del activo se conoce como **riesgo**.

Por lo general, se puede cuantificar el nivel de riesgo como el producto del impacto (daño causado al activo) por la probabilidad de ocurrencia. Esto se conoce como análisis o evaluación del riesgo.

Diferentes controles o medidas de seguridad se aplican sobre diferentes principios de seguridad para mitigar o reducir el riesgo. Por ejemplo, la generación de copias de respaldo permite recuperar información perdida o dañada, fortaleciendo la disponibilidad de la misma. Así mismo, la autenticación y autorización de usuarios garantiza un acceso controlado a la información, evitando que se comprometa la confidencialidad e integridad de los datos.

2.2. Autenticación, Autorización y Trazabilidad

Como se mencionó en la sección anterior, el objetivo de un sistema de seguridad de la información es asegurar la continuidad de la triada Confidencialidad-Integridad-Disponibilidad sobre los activos de una organización. El **control de acceso** juega un rol importante en este sentido. La administración de acceso a los activos de información es fundamental para prevenir la divulgación y acceso no autorizado a los datos, controlando quien puede ver, usar, modificar o destruir estos activos. Es normal tener tres procesos de seguridad en los que se apoya el control de acceso[8]. Estos son:

Autenticación Determina y valida la identidad del usuario. Consta de dos fases: la *identificación* y la *autenticación* propiamente dicha. En la fase de identificación se provee de la identidad del usuario a los sistemas de seguridad. El mecanismo básico de identificación es utilizando un identificador o ID de usuario. En la fase de autenticación se valida la identidad que reclama el usuario en base a las evidencias que presenta para demostrarlo. Esta evidencia se conoce con el nombre de **credencial** (ver sección 2.3). El usuario

puede demostrar su identidad a través de algo que conoce (contraseña o *password*), posee (*token, smart card*) y/o es (huella digital, iris, voz) [9].

Autorización Provee al usuario el acceso a los recursos que tiene autorizado acceder, otorgando los permisos y/o privilegios necesarios para ello. Así mismo, previene al usuario el acceso a los recursos a los cuales no posee autorización.

Trazabilidad Provee de un registro de auditoria y seguimiento de las actividades del usuario dentro del sistema de información.

Es común hacer referencia a estos tres procesos con las siglas **AAA** por sus nombres en inglés Authenticaction, Authorization and Accounting (AAA). Existen diferentes protocolos que cumplen con uno o varios de estos procesos y que serán descriptos en secciones posteriores de este capítulo.

La identificación y autenticación del usuario es a menudo responsabilidad del sistema operativo. Antes de permitir crear un sólo proceso, el usuario debe identificarse con el mismo. Las aplicaciones y servicios pueden valerse de esta autenticación o solicitar una autenticación adicional (ver sección 2.5).

La mayoría de las veces, la autenticación y la autorización trabajan en conjunto, y resulta difícil distinguir donde termina una y donde comienza la otra. Para disponer de una independencia lógica entre la autenticación y la autorización los sistemas operativos y aplicaciones implementan lo que se conoce como proceso de *inicio de sesión*, *login* o *sign-in*. Este proceso se utiliza para identificar y autenticar al usuario iniciando un diálogo entre éste y el sistema para generar el entorno de seguridad necesario, denominado **token de acceso**. Luego, dicho token se añade a cada proceso que ejecuta el usuario y es utilizado por el proceso de autorización para determinar el nivel de acceso correspondiente.

Un sistema de autenticación consta principalmente de tres componentes, como se observa en la figura 2.1.

Suplicante La parte del proceso de autenticación que provee su identidad y las evidencias de esto para solicitar la autenticación al sistema. El suplicante también es referido como usuario autenticador o simplemente cliente.

Autenticador Parte del proceso que provee los recursos al cliente (suplicante) y necesita asegurar la identidad del mismo para realizar la autorización y los registros de auditoria sobre los recursos que se acceden. También referido como servidor o proveedor del servicio (Service Provider (SP)).

Autoridad/Base de Datos de Seguridad Mecanismo o almacenamiento que verifica las credenciales del cliente. Puede ser un simple archivo con las credenciales de los usuarios almacenado localmente en cada computadora, o

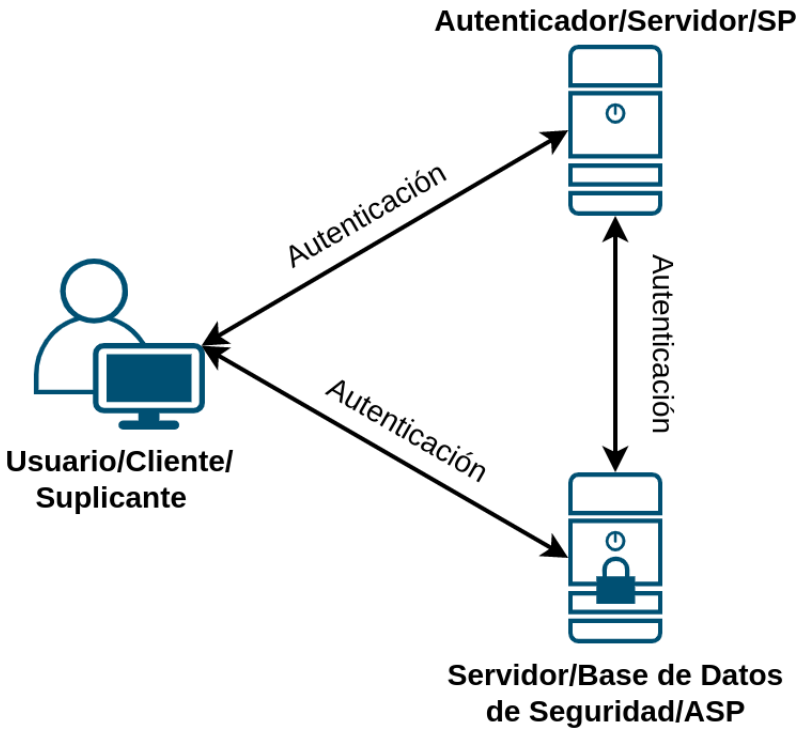

Figura 2.1: Componentes de un sistema de autenticación

un servidor central de autenticación que provee los mecanismos necesarios para la autenticación de los usuarios a través de la red. También referido como proveedor del servicio de autenticación (Authentication Service Provider (ASP)) o proveedor de identidad (Identity Provider (IdP)).

En un escenario simple, el suplicante, el autenticador y la autoridad de seguridad pueden residir en la misma computadora o, en esquemas más complejos, estar distribuidos en múltiples computadoras. Lo importante es entender que las tres partes se pueden comunicar de manera independiente. Dependiendo del mecanismo de autenticación y el modelo de confianza utilizado alguno de estos canales de comunicación pueden o no estar presentes. Por ejemplo el protocolo de autenticación Kerberos (ver sección 2.7.3), involucra comunicación directa entre el suplicante y el servidor de seguridad y entre el suplicante y el autenticador, pero no existe tal comunicación entre el servidor de seguridad y el autenticador.

Es importante diferenciar entre la autenticación *humano a máquina* (autenticación de usuario) y *máquina a máquina* (autenticación Machine-to-Machine (M2M)). En la autenticación M2M se utilizan protocolos bien establecidos que permiten la autenticación sobre canales seguros con claves lo suficientemente

grandes para evitar ser vulnerables. Un ejemplo es el protocolo Transport Layer Security (TLS)/Secure Socket Layer (SSL) utilizado para la navegación web en sitios seguros de Internet. En este caso la máquina cliente verifica la identidad del servidor web sin intervención humana, haciendo uso de claves y certificados de identidad almacenados digitalmente.

Sin embargo, la autenticación de usuarios resulta más vulnerable debido a que los humanos utilizan contraseñas que se almacenan en su memoria, por lo que suelen ser más cortas y simples que las utilizadas por las máquinas. Lo que se pretende en la autenticación de usuarios es validar la identidad de la persona que va a utilizar la máquina o acceder al sistema informático, posterior a una eventual autenticación M2M. En la figura 2.2 se observa la interacción de ambos tipos de autenticación.

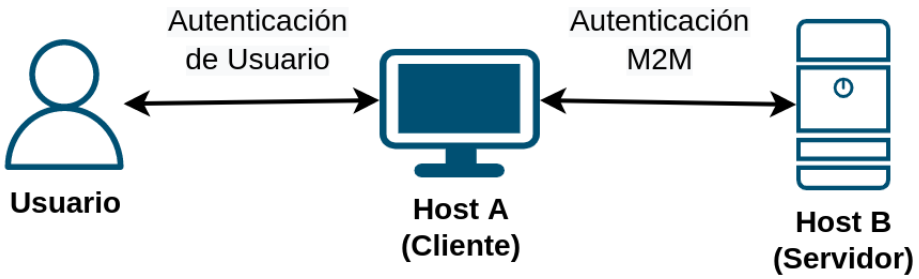

Figura 2.2: Autenticación de Usuario y M2M

La autenticación M2M también se suele presentar durante el proceso de autenticación a la infraestructura de red (sección 2.8). En estos casos, la autenticación de usuarios se extiende a una autenticación de pares, donde es necesario validar tanto la identidad del dispositivo cliente (suplicante) que se conecta a la red, como la identidad del dispositivo servidor (Network Access Server (NAS)). Cuando la autenticación es en ambos sentidos se lo denomina *autenticación mutua*.

Algunos mecanismos de autenticación admiten la autenticación mutua de forma predeterminada; otros mecanismos pueden soportar la autenticación mutua usando dos diálogos de autenticación separados en ambas direcciones; y algunos mecanismos de autenticación no admiten la autenticación mutua en absoluto.

2.3. Credenciales de acceso

La credencial de acceso o *factor de autenticación* es la evidencia que presenta el cliente (suplicante) al servidor (autenticador) para demostrar su identidad y obtener el acceso correspondiente a los recursos informáticos. Las credenciales de

acceso se agrupan comúnmente en tres categorías: algo que se conoce, algo que se tiene y algo que se es. Para [9] los factores de autenticación deberían agruparse con las siguientes etiquetas:

Basado-en-conocimiento Caracterizado por el secreto u obscuridad, como una contraseña o la respuesta a una pregunta que solo el usuario conoce.

Basado-en-objeto Caracterizado por la posesión física de un objeto, como una tarjeta inteligente o un token.

Basado-en-ID Caracterizado por la unicidad a una persona, como un documento de identidad, pasaporte o rasgos biométricos del usuario.

La tabla 2.1 resume las características más importantes de las credenciales de acceso según [9]

	Basado-en-conocimiento	**Basado-en-objeto**	**Basado-en-ID**
Conocido como:	Password (contraseña), secreto	Token	Biométrico
Autenticación soportada por:	Secreto u obscuridad	Posesión	Unicidad y personalización
Fortaleza:	Custodia del secreto	Custodia del objeto	Resistencia a la falsificación
Ejemplo tradicional:	Cerradura de combinación	Llave metálica	Documento de identidad
Ejemplo digital:	Password de computadora	Llave inalámbrica de automóvil	Lector de huellas dactilares
Debilidad:	Menos secreto con cada uso	Inseguro si se pierde el objeto	Difícil de reemplazar

Tabla 2.1: Características de las credenciales de acceso

Las credenciales de acceso se pueden combinar para incrementar la seguridad en el proceso de autenticación. Cuando se combinan dos o más tipos de credenciales se conoce como **autenticación multifactor**. Es común combinar dos tipos de credenciales, como por ejemplo un password con un token: algo que se conoce con algo que se tiene, o un password con una evidencia biométrica: algo que se conoce con algo que se es. En este caso la autenticación se llama Two Factor Authentication (2FA). Un ejemplo de uso cotidiano en un proceso 2FA es el de una tarjeta magnética protegida con un Personal Identification Number (PIN) para acceder a la cuenta bancaria.

Como contra partida, la autenticación de doble factor disminuye la usabilidad del sistema y aumenta el tiempo en el proceso de login. Debido a esto muchas veces el usuario se resiste a utilizar este tipo de mecanismo.

En las siguientes secciones describiremos con más detalle las credenciales de acceso mayormente utilizadas en los sistemas de autenticación actuales.

2.3.1. Passwords

La utilización de passwords o contraseñas es la forma más primitiva de validar la identidad del usuario en un proceso de autenticación. El modelo de confianza para este método de autenticación se basa en el secreto compartido entre el suplicante y el autenticador. Este secreto puede ser válido por un período prolongado de tiempo: password estático, o cambiar con cada uso (OTP). A pesar del creciente número de novedosos métodos de autenticación, el basado en password es aún el más popular de todos [10].

En [11] se realizó una evaluación de 35 métodos de autenticación web diferentes, utilizando un framework con 25 criterios de comparación, y no se logró hallar una propuesta superadora con respecto a los passwords estáticos.

La autenticación basada en passwords estáticos es muy sencilla de implementar, no necesita ninguna configuración especial en el cliente o en el servidor y prácticamente cualquier sistema operativo, aplicación o servicio lo soporta. Sin embargo, este tipo de autenticación suele ser vulnerable a varias formas de ataque cuando la clave elegida como password es débil. Para generar claves fuertes, se debe establecer una política de contraseñas donde se especifique, entre otras cosas, la longitud mínima de la clave (por lo general de 8 caracteres), el tipo de combinación de letras, números y símbolos, y la frecuencia con la que se debe cambiar la misma [12].

Un estudio realizado por [13], sugiere que la política utilizada priorice el uso de palabras o frases largas como clave (*passphrase*), en lugar de claves cortas y complejas.

Con ello se consigue que las contraseñas sean fáciles de recordar para los usuarios, y que los ataques de diccionario o fuerza bruta no sean tan efectivos. Lamentablemente, esto no quita que la contraseña sea descubierta por otros métodos como el husmeo o sniffing de la red, keyloggers, shoulder surfing, phishing y otro tipos de ataques de ingeniería social.

Por otro lado, la autenticación con OTP no es vulnerable a estos tipos de ataques porque las contraseñas se descartan con cada uso, pero requiere de software y/o hardware especiales tanto en el cliente como en el servidor. La manera más sencilla de implementar OTP es teniendo una lista de passwords compartida entre el suplicante y el autenticador, que se utilizará de manera secuencial hasta el final de la misma. Este mecanismo no resulta práctico para utilizarlo en la vida real.

Una manera más elegante de autenticar con OTP es mediante el esquema desarrollado por Bellcore para los sistemas UNIX denominado S/KEY. Con S/KEY el cliente y el servidor comparten una clave inicial o maestra que permite, por medio de un índice i, generar el password de un solo uso según una función hash (Message-Digest Algorithm 4 (MD4) o Message-Digest Algorithm 5 (MD5)) que se aplica $N - i$ veces sobre la clave maestra. La fortaleza de este mecanismo de

autenticación radica en la enorme dificultad para obtener el resultado inverso de la función hash.

El calculo del password, por parte del suplicante, por lo general se realiza en un dispositivo seguro desconectado de la red, donde se almacena la clave maestra. Por lo tanto, este mecanismo de autenticación pasa a depender de algo que se tiene (lista de passwords o dispositivo seguro) en lugar de algo que se conoce, como sucede con los passwords estáticos.

Aunque S/KEY resulta un mecanismo de autenticación interesante para ser utilizado como OTP, actualmente se utilizan otros métodos más sencillos y estandarizados como HMAC-Based One-Time Password (HOTP) [14] y Time-Based One-Time Password (TOTP) [15].

2.3.2. Claves asimétricas

La criptografía asimétrica representa un método poderoso para proteger información. Los usuarios que requieran utilizar este tipo de criptografía como mecanismo de autenticación necesitan poseer una par de claves: pública y privada. La clave privada es secreta y puede ser considerada equivalente al password del usuario. La clave pública no es secreta y puede estar disponible a todos los usuarios o dispositivos que lo requieran.

Un par de claves asimétricas, sea del tipo Rivest, Shamir and Adleman (RSA) o Digital Signature Algorithm (DSA), por sí mismas no poseen información sobre a que usuario pertenecen. Uno de los enfoques utilizados por las aplicaciones, como SSH, es utilizar la clave pública para identificar al usuario y la clave privada como medio para autenticarlo (sección 2.7.6). Debido a que la clave pública y privada se encuentran matemáticamente ligadas, la identidad del usuario (clave pública) está asociada a su clave secreta (clave privada).

Sin embargo, utilizar sólo el par de claves no provee la flexibilidad necesaria, como por ejemplo, para obtener información sobre cuando fueron creadas o por cuánto tiempo son válidas. Para ello, existe el *certificado* que es un conjunto de atributos de usuario públicamente disponible (metadata), verificado y firmado por una entidad de confianza que garantiza que la información contenida en el mismo es correcta. Esta entidad de confianza se conoce como Certification Authority (CA).

Cada certificado contiene, entre otros atributos, la identidad del usuario asociado a la clave pública correspondiente. El estándar X.509 define el formato de los certificados digitales y el algoritmo utilizado para validar la ruta de certificación dentro de la infraestructura de clave pública o Public Key Infrastructure (PKI).

El formato de un certificado X.509v3 viene dado por los siguientes campos de interés:

- Versión: 3
- Número de Serie
- Algoritmo de la Firma: por ejemplo *sha1WithRSAEncryption*.
- Emisor (CA)
- Período de Validez
- Sujeto: expresado en notación Distinguished Name (DN). Puede ser una persona, un servidor o un servicio.
- Algoritmo y Clave Pública del Sujeto.
- Extensiones (opcional): Atributos extras que permiten personalizar el certificado.
- Firma del Certificado

Un ejemplo en el uso de criptografía asimétrica para la autenticación es la propia firma del certificado. El CA firma el certificado utilizando los algoritmos especificados en el campo Algoritmo de la Firma. Para ello, realiza un hash sobre los datos del certificado y luego encripta el resultado con su clave privada.

Luego, el destinatario del certificado puede validar el contenido del mismo desencriptando la firma con la clave pública del CA y comparando la integridad de los datos contra su propio calculo de hash. De esta manera, se asegura que el certificado no ha sido adulterado y que además corresponde a un CA de confianza.

El certificado X.509 y su correspondiente clave privada pueden ser utilizados como credenciales de autenticación muy seguras, especialmente si la clave privada se almacena en tarjetas inteligentes o tokens como medios para proveer 2FA.

Pretty Good Privacy (PGP) es un sistema de clave pública similar en varios aspectos a X.509. La principal diferencia radica que en lugar de apoyarse sobre un modelo de confianza jerárquico, PGP utiliza el modelo de *redes de confianza*.

En PGP cualquier usuario puede hacer el rol de CA y firmar el certificado de otro usuario, validando su identidad por algún otro mecanismo externo al sistema. Si un usuario A confía/conoce a otro usuario B, por transitividad, va a confiar en todos los certificados firmados por éste último.

La confianza empresarial suele ser más fácil de implementar utilizando el modelo X.509 jerárquico y centralizado, basado en autoridades de certificación raíz e intermedias. PGP, por otro lado, suele ser más conveniente para las comunidades de usuarios abiertas que intercambian información a través de Internet. Tanto los certificados X.509 como los PGP se pueden utilizar como credenciales para la autenticación de usuarios, dependiendo de los requisitos y entorno de trabajo.

2.3.3. Credenciales biométricas

Las credenciales biométricas representan el factor de autenticación *algo que eres*. Se basan en las características físicas o de comportamiento del usuario. La idea detrás de las credenciales biométricas es que las características medibles, como la huella dactilar o la dinámica en la escritura del usuario, no cambian o casi nunca cambian y, por lo tanto, pueden usarse para autenticar la identidad de un usuario.

Las credenciales biométricas se pueden usar solo para autenticación (verificación) o para identificación y autenticación (reconocimiento). En ambos caso, la base de datos de autenticación debe completarse con los perfiles biométricos de todas las personas que serán identificadas o autenticadas. Este proceso se conoce como registración o enrolamiento.

Cuando se utilizan las credenciales biométricas para identificar al usuario (reconocimiento), éste no necesita presentar una identificación al sistema. Se capturan y analizan las credenciales biométricas del usuario y luego se realiza una búsqueda en la base de datos de autenticación para determinar si existe un usuario conocido con el perfil biométrico obtenido. Si se encuentra una coincidencia, el usuario se identifica y autentica.

Las medidas de precisión y eficacia de los dispositivos biométricos son los parámetros de tasa de falsa aceptación (FAR) y tasa de falso rechazo (FRR), que son específicos para cada tipo de credencial biométrica e implementación.

El objetivo de los sistemas de autenticación biométrica es mantener el FAR y FRR bajos y, al mismo tiempo, brindar una autenticación conveniente y rápida.

Existen dos tipos principales de credenciales biométricas:

Estática (basada en patrones) Se basa en el patrón estático de una característica biométrica, como una huella dactilar, un patrón de retina o un patrón de iris. El patrón generalmente se almacena como una imagen rasterizada o vectorial en una base de datos de autenticación. En el momento de la autenticación del usuario, el reconocimiento se basa en el número de puntos coincidentes entre la imagen almacenada y la imagen capturada. Más puntos coincidentes significan una mayor precisión.

Dinámica (basada en comportamiento) La autenticación mediante credenciales biométricas dinámicas se basa en el reconocimiento del comportamiento específico del usuario, como la dinámica de la escritura a mano o la forma en que tipea un texto específico, como su contraseña.

Algunos de los métodos de autenticación biométrica más comunes son:

- Autenticación de huellas dactilares. Este método de autenticación basado en

patrones es, por lejos, el más popular. Se basa en el hecho de que las huellas dactilares de los usuarios son prácticamente únicas.

- Escaneo de retina. Este es un método de autenticación basado en patrones y se basa en la unicidad de la formación de vasos sanguíneos en la parte posterior del ojo.
- Exploración del iris. Similar al escaneo de retina, este método se basa en la singularidad del anillo de color del tejido alrededor de la pupila (el iris).
- Geometría de la mano. Este método se basa en la unicidad de las dimensiones y proporciones de la mano y los dedos del individuo.
- Geometría de la cara. Este mecanismo fue desarrollado para imitar la forma humana natural de identificar a las personas en función de sus rostros.
- Patrón de piel. Basado en la unicidad de la textura de la piel, este método crea una huella de la piel.
- Patrón de voz. Este método se basa en la unicidad de la voz humana.
- Escritura. Basado en el comportamiento del usuario cuando escribe a mano.

Las tecnologías de autenticación biométrica presentan más potencial que efectividad real y amplia implementación. A medida que evolucionen las tecnologías, es probable que aumente la precisión y la comodidad para los usuarios, y los dispositivos biométricos pueden tomar la delantera natural y bien merecida como credenciales de autenticación de usuarios.

2.4. Sistemas Single Sign-On

Existe una gran variedad de sistemas de autenticación los cuales se clasifican en dos grandes categorías: centralizados y descentralizados.

Un sistema de autenticación descentralizado permite a cada computadora mantener su propia base de datos de usuario y política de acceso de forma local. Por ejemplo, cada computadora con Linux utiliza de manera independiente sus propios archivos *passwd* y *shadow* como base de datos para las cuentas de usuario. De igual manera, las computadoras con Windows gestionan las cuentas de usuario locales en la base de datos Security Account Manager (SAM).

Los esquemas de autenticación centralizados requieren que todas las computadoras de un determinado dominio confíen en una autoridad de de autenticación central. Por ejemplo, en Linux eso se puede lograr mediante la interfaz Name Services Switch (NSS) que permite consolidar diferentes bases de datos de usuario,

independientemente de su formato y método de acceso. En Windows, los esquemas de autenticación mediante Windows NT y Active Directory son ejemplos de sistemas de autenticación centralizados. En las secciones 2.6.1 y 2.6.2 se describen con mayor detalle los procesos de autenticación en los sistemas Linux y Windows respectivamente.

La autenticación centralizada es una forma genérica de referirnos a los sistemas SSO, los cuales permiten de una manera unificada autenticar y acceder tanto a los servicios (email, archivos compartidos, portales webs, sistemas de gestión, etc.), como a la infraestructura (seguridad de puertos, VPN, wifi, etc.).

En un mundo ideal, todas las aplicaciones y servicios de una organización deberían utilizar una única base de datos de autenticación centralizada pero, desafortunadamente, el acceso a la capa de infraestructura, así como las aplicaciones y servicios en las capas superiores, tienen sus propios métodos de autenticación y bases de datos de usuario.

SSO a menudo se conoce como el *Santo Grial* de la autenticación y autorización para la industria de la seguridad en las tecnologías de la información [16]. Los beneficios para los usuarios son obvios si una solución de SSO está implementada: los usuarios tendrán que usar un conjunto de credenciales para autenticarse solo una vez y acceder a todos los sistemas que tengan autorización.

Según algunos estudios, el usuario promedio tiene de 8 a 13 pares de credenciales que utiliza diariamente. Además, otro estudio sugiere que podría tomar hasta 20 minutos para que un usuario restablezca una contraseña olvidada [6]. Considerando que el 25 al 50 por ciento de las llamadas al servicio de asistencia técnica están relacionadas con contraseñas, las empresas pueden ahorrar una cantidad significativa de gastos, si optan por invertir en soluciones SSO.

Las ventajas de un sistema SSO no sólo se relacionan con la facilidad de uso y administración, también acarrea importantes beneficios relacionados a la seguridad. Centralizar la autenticación facilita el aseguramiento de una política de autenticación consistente en toda la organización.

Una buena solución SSO es neutral a las plataformas y aplicaciones utilizadas: debe ocultar al usuario los detalles en los diferentes mecanismos de autenticación de los sistemas operativos y proveer las interfaces necesarias para integrar las aplicaciones a una autoridad de autenticación central.

Un típico argumento en contra de los sistemas SSO es que las credenciales SSO se convierten en la ”llave del reino”. Si uno pudiera obtener las credenciales SSO de un usuario en particular, puede acceder a todos los recursos y servicios que éste tiene autorización. Pero este riesgo puede ser mitigado o reducido implementando autenticación de doble factor, como se describe en la sección 2.3.

Según [17] se distinguen dos tipos de sistemas SSO. El primer tipo se conoce como **pseudo-SSO** ya que utiliza un componente SSO intermedio para el proceso de autenticación entre el usuario y el proveedor del servicio (SP). Al comienzo de

la sesión SSO, el usuario realiza una *autenticación primaria* con el componente pseudo-SSO. Luego, dicho componente ejecuta una autenticación separada para cada SP que el usuario desea acceder, según los requerimientos de autenticación e identidades SSO específicas de cada servicio.

El segundo tipo es considerado un sistema **SSO verdadero**. En este caso, el usuario se autentica a un proveedor de servicio de autenticación (ASP), el cual mantiene una relación de confianza con los SP. La característica principal que distingue a un sistema SSO verdadero de un pseudo-SSO, es que el proceso de autenticación ocurre sólo una vez entre el usuario y el ASP; los SP son notificados del estado de la autenticación del usuario a través de las llamadas *aserciones de seguridad* (token de acceso), que proveen sólo autorización y no autenticación. Otra diferencia es que las identidades SSO se mantienen homogéneas dentro de todo el sistema SSO verdadero.

[17] clasifica a todos los sistemas SSO en verdadero o pseudo-SSO, que a su vez pueden implementarse de manera local a la plataforma del usuario o como un servicio externo (proxy SSO). En la figura 2.3 se observa el proceso de autenticación de ambos tipos de sistemas SSO.

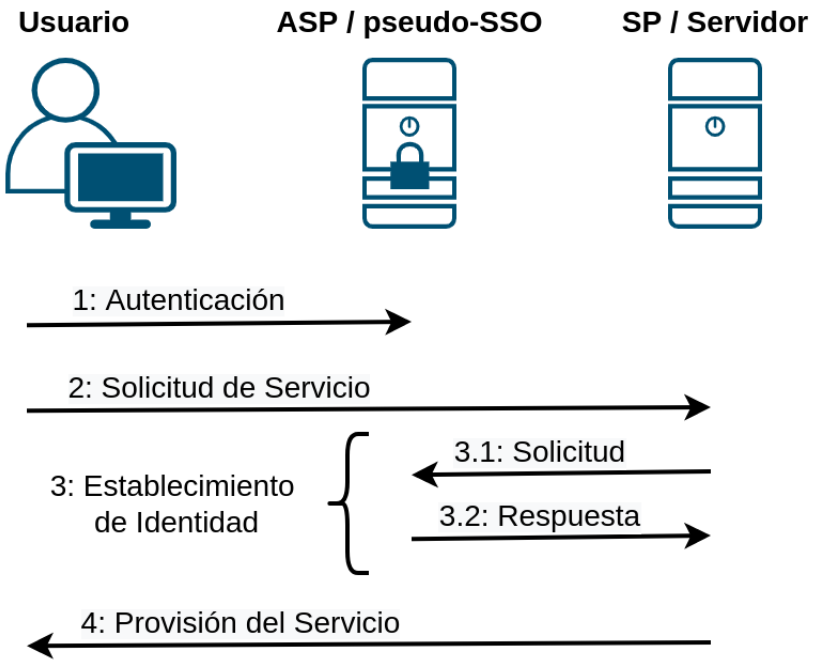

Figura 2.3: Flujo de mensajes en un sistema de autenticación SSO genérico

Desde una perspectiva de implementación, los sistemas SSO se pueden clasificar en simples y complejos dependiendo de su arquitectura [18].

Una arquitectura **SSO simple** implica el despliegue de una única autoridad de autenticación y un único conjunto de credenciales por usuario. Este tipo de arquitectura puede ser implementada con facilidad en redes de área local o intranets homogéneas, donde las computadoras se ejecutan sobre la misma plataforma operativa y confían en la misma autoridad de autenticación.

Por otro lado, una arquitectura **SSO compleja** es gobernada por más de una autoridad de autenticación y con uno o mas conjuntos de credenciales por usuario. Esta diversidad de contexto implica que el sistema SSO puede implementarse sobre varias plataformas y cubierto por varias organizaciones, con diferentes protocolos y credenciales de autenticación. A su vez, una arquitectura SSO compleja se puede clasificar, según la cantidad de credenciales que se necesitan por usuario, en credencial única o credenciales múltiples.

Según [18], las arquitecturas **SSO complejas de credencial única** pueden ser basadas en Token o en PKI. Ambas arquitecturas proveen SSO en entornos *bastante* homogéneos; esto significa usar un único formato para las cuentas de usuario y un único protocolo de autenticación soportado por cada entidad, aplicación y servicio que participa del entorno SSO. Además, ambas arquitecturas son consideradas sistemas SSO verdaderos ya que dependen de un único proceso de autenticación primario al igual que las arquitecturas SSO simples.

En una arquitectura SSO basada en token el usuario obtiene, luego de una autenticación exitosa con la autoridad de autenticación primaria, un token temporal que puede ser utilizado para acceder a los recursos del dominio primario y secundarios, en caso de que el usuario participe en más de un dominio. La validación del token por parte de las entidades participantes se realiza mediante criptografía de clave simétrica. Un ejemplo clásico de este tipo de autenticación es el protocolo Kerberos (ver 2.7.3)

En la arquitectura SSO basada en PKI el usuario genera un par de claves asimétricas: pública y privada. La clave pública es enviada a la autoridad de autenticación, llamada CA en este caso, para ser certificada luego de validar la identidad del usuario. El CA devuelve al usuario un certificado de clave pública que será utilizado, junto con la clave privada, como un tipo de token para acceder a los recursos del dominio primario o secundarios en caso de ser necesario. La principal diferencia de esta arquitectura con la basada en token es que la validación de la identidad se realiza mediante criptografía asimétrica en lugar de simétrica.

Por último, se tienen las arquitecturas **SSO complejas multi-credenciales**, las cuales pueden ser del tipo sincronizadas o basadas en cache. Estas arquitecturas son consideradas pseudo-SSO ya que utilizan un componente de autenticación que mantiene las credenciales del usuario sincronizadas en los diferentes SP o almacenadas en cache según sea el caso.

Los trabajos de [19], [20] y [6] realizan un estudio y análisis más profundo de las diferentes arquitecturas de SSO, del cual se obtiene una clasificación o

taxonomía por capas, ampliando así los trabajos de clasificación realizados por [18] y [17].

Según [6] existen cinco capas de clasificación para los sistemas SSO: Entorno, Implementación, Tipo de Credencial, Protocolo y Hospedaje. A su vez, dentro de la capa de Entorno existen tres variantes donde se pueden implementar los sistemas de SSO: Intranet, Extranet e Internet.

El entorno de Intranet se conoce comúnmente como Enterprise SSO (ESSO), donde los usuarios pueden acceder mediante SSO a múltiples sistemas y aplicaciones dentro de una misma organización. En cambio, Extranet es un entorno multi-dominio donde se permite a un usuario autenticarse en su domino y acceder a los recursos de otro dominio.

Para finalizar, se tiene el entorno de Internet el cual es considerado un sistema SSO basado en Web, donde los usuarios acceden a los recursos utilizando su navegador web. En la figura 2.4 se resume la taxonomía ampliada de [6].

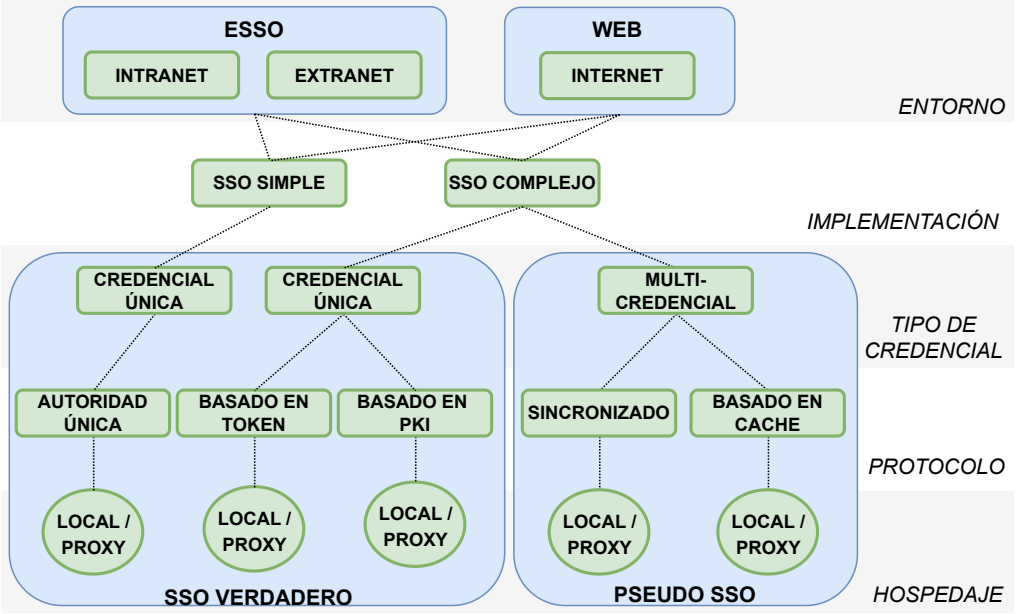

Figura 2.4: Taxonomía de los sistemas SSO

Los sistemas SSO son vitales para las empresas y organismos públicos como el TEPM, los cuales deben esforzarse por implementar soluciones que sean las más apropiadas para el entorno técnico y necesidades de negocio.

No existe una solución SSO de ”talla única”. En general, cada caso requiere un conjunto de tecnologías compuesto por una combinación de sistemas SSO.

2.5. Autenticación a la Infraestructura y los Servicios

Mientras que la infraestructura de red es solo una capa de comunicación, las aplicaciones y los servicios son los recursos reales que los usuarios necesitan para realizar sus trabajos diarios. Estos últimos incluyen sistemas de gestión de información, aplicaciones financieras, correo electrónico, servicios de archivos e impresión, portales web corporativos, etc., que se valen de la infraestructura de red para proporcionar conectividad de forma segura.

Se debe solicitar a los usuarios que autentiquen el acceso tanto a los servicios como a la infraestructura, al intentar acceder a aplicaciones de la organización o establecer una conexión Virtual Private Network (VPN) desde el hogar a la red corporativa. Es posible que los usuarios incluso necesiten utilizar el mismo nombre de usuario y contraseña u otras credenciales tanto para acceder a las aplicaciones como a la conexión de VPN. Sin embargo, estos dos tipos de acceso requieren de dos procesos de autenticación independientes.

Las aplicaciones y los servicios suelen requerir su propia autenticación, independientemente de si se ha realizado o no la autenticación de la infraestructura. Por lo general, estos servicios no tienen conocimiento de si el usuario ha sido autenticado para acceder a la infraestructura o si se le ha otorgado acceso debido al hecho de que se encuentra físicamente en el edificio.

Como resultado, se considera que la autenticación de usuarios consta de al menos una capa de autenticación para la infraestructura y una para las aplicaciones. Por lo tanto, un usuario que intente acceder a las aplicaciones puede necesitar autenticarse dos veces: a la infraestructura y luego a las aplicaciones. En el caso de sistemas SSO, como se vio en la sección 2.4, dependerá del nivel de implementación de dicho sistema.

2.5.1. Suplantación y Delegación

El sistema operativo es el responsable de ejecutar los servicios y aplicaciones así como las conexiones a la infraestructura de red, gestionando de manera segura todos los recursos del sistema. Cuando un usuario utiliza una computadora se debe autenticar al sistema utilizando su cuenta de usuario. El proceso que realiza el login lanza un *shell* de usuario utilizando la identidad del usuario autenticado. A partir de ahí, todos los procesos que ejecuta el usuario asumen la identidad y privilegios de éste a medida que se accede a los diferentes recursos del sistema.

Los servicios y aplicaciones de red trabajan de la misma manera: típicamente tienen una cuenta de usuario dedicada para ellos (o utilizan una cuenta de máquina, como es el caso de varios servicios en los sistemas Windows). Cuando un

usuario intenta acceder a un servicio de red, éste último tomará alguno de los siguientes enfoques:

Acceso anónimo El servicio no necesita conocer la identidad del usuario y provee el mismo tipo de acceso a todos los usuarios.

Identificación El servicio necesita conocer la identidad del usuario para implementar su propio mecanismo de autorización y determinar los niveles de acceso otorgados al usuario.

Suplantación El servicio asume la identidad del usuario y genera un nuevo proceso o hilo con dicha identidad y los privilegios correspondientes (similar al proceso de login del sistema operativo). Luego, el servicio intentará acceder a los recursos locales en nombre del usuario autenticado, delegando la tarea de autorización al sistema operativo donde se ejecuta el servicio. En este caso, el servicio no puede acceder a recursos en otro servidor en nombre del usuario.

Delegación El usuario delega sus credenciales de acceso al servicio (por ejemplo un password o ticket), para que éste lo utilice para suplantar su identidad en la computadora donde se está ejecutando. Además el servicio puede acceder a otros recursos en la red en nombre del usuario, pasando las credenciales que le fueron provistas.

Diferentes aplicaciones pueden utilizar diferentes enfoques de suplantación y delegación dependiendo de su arquitectura y requerimientos de seguridad.

En las siguientes secciones se describe primero la autenticación a los sistemas operativos Linux y Windows, para luego comprender los procesos de autenticación de las aplicaciones y servicios de red. Para finalizar, se aborda la autenticación a la infraestructura, proceso previo necesario para acceder a cualquier tipo de servicio en la red.

2.6. Autenticación de usuarios en Linux y Windows

2.6.1. Autenticación en Linux

Para que un usuario pueda acceder a un recurso en una computadora con Linux, éste necesita una cuenta de usuario y su correspondiente credencial de acceso. Una cuenta de usuario consta de al menos tres atributos importantes:

Username un nombre que identifica al usuario en el sistema.

User ID (UID) un número entero que identifica de manera unívoca al usuario en el sistema.

Group ID (GID) un número entero que especifica el grupo primario al que pertenece el usuario.

Cuando un usuario se autentica de manera interactiva o accede a recursos del sistema desde la red, estos tres atributos deben estar disponibles para determinar el nivel de acceso a dichos recursos (autorización) y potencialmente reflejar el acceso en los archivos de registro y auditoría del sistema. Los tres atributos son el resultado del proceso de autenticación y se los conoce en conjunto como privilegio o **token de acceso** [8].

Existen cuentas de usuario predefinidas dependiendo de las diferentes distribuciones de Linux. Estas cuentas de sistema utilizan, por lo general, UIDs menores a 500 para distinguir de las cuentas de usuario estándar. Las cuentas de sistema son utilizadas por *demonios* o servicios para el correcto funcionamiento del sistema operativo. Otra de las cuentas importantes es la cuenta del superusuario **root**, identificada con el UID 0. El superusuario no tiene restricciones para acceder o modificar cualquier recurso del sistema.

La información de las cuentas de usuario en Linux se almacenan en cuatro archivos de sistema diferentes ubicados en el directorio */etc* :

passwd Contiene la información básica de una cuenta de usuario. Cada línea del archivo representa una cuenta de usuario con el siguiente formato:

```
username:password:uid:gid:comment:home_directory:shell
```

shadow Contiene el *password* encriptado del usuario y toda la información extra referida al mismo en el siguiente formato:

```
username:password:last_change:min:max:warn:inactive:expire:reserved
```

group Contiene información relacionada a los grupos en el siguiente formato:

```
group_name:password:gid:user_members
```

gshadow Contiene el *password* encriptado para el grupo en el siguiente formato:

```
group_name:password:administrators:user_members
```

El password de un grupo se utiliza cuando un usuario que no es miembro del grupo necesita obtener permisos en nombre de dicho grupo (comando *newgrp*).

Los archivos *passwd*, *shadow* y *group* son una alternativa sencilla para la autenticación de usuarios en un solo host pero no resulta escalable para entornos de múltiples hosts. Además de estos archivos utilizados para la autenticación, existen otros archivos como */etc/hosts* y */etc/networks* que son necesarios para la resolución de nombres en entornos corporativos.

Una extensión a estos archivos es la interfaz **NSS**, implementada por primera vez por Sun Microsystems y luego adoptada por los sistemas Linux. La interfaz NSS permite definir en el archivo */etc/nsswitch.conf* diferentes fuentes de información para los procesos de autenticación y resolución de nombres. Los archivos locales en el directorio */etc* son una de las posibles fuentes, junto con otras como el Network Information System (NIS), Lightweight Directory Access Protocol (LDAP) y Domain Name System (DNS). La función de la interfaz NSS es consolidar todas estas fuentes, independientemente de su formato y método de acceso, y retornar el resultado de una llamada en un formato estándar. La herramienta `getent <source>` permite obtener información consolidada de la fuente especificada en el comando.

La autenticación basada en los archivos *passwd*/*shadow* o en la interfaz NSS tiene como desventaja que las aplicaciones deben implementar su propio código para la obtención y validación de las credenciales de acceso. Además, no existe una manera estándar de administrar dichas credenciales. La interfaz NSS provee información de sólo lectura; la aplicación de usuario es la responsable de implementar una interfaz para modificar las credenciales dependiendo de la ubicación donde se encuentren las cuentas de usuario.

Para resolver tales inconvenientes se desarrolló la autenticación basada en **Pluggable Authentication Modules (PAM)**. PAM separa los procesos de identificación y autenticación de usuarios, otorgando una plataforma de autenticación universal y configurable para las aplicaciones y sistema operativo [21]. Los siguientes servicios son ofrecidos por PAM:

auth encargado de validar las credenciales de usuario. Existen diferentes módulos de PAM que permiten autenticar al usuario contra varias bases de datos distintas, no sólo los archivos *passwd*/*shadow*.

account encargado de validar el estado de la cuenta de usuario: si el usuario ya se encuentra autenticado, si el password no expiró, si no existen restricciones de tiempo para la cuenta, etc.

session encargado de registrar el acceso del usuario al sistema.

passwd Permite al usuario modificar su password como así también que el administrador pueda reiniciar el password de otros usuarios.

PAM no provee identificación de usuarios, por lo que las aplicaciones se siguen valiendo de NSS para ello. Las aplicaciones legadas utilizan las funciones *getpwent()* y *getpwuid()* que resultan en llamadas a la interfaz NSS. Sin embargo, las aplicaciones compiladas con soporte para PAM utilizan las funciones específicas *pam_start()*, *pam_authenticate()*, *pam_end()* y *pam_setcred()*, según la configuración disponible en los archivos */etc/pam.d/*. La configuración por defecto de PAM utiliza el módulo estándar *pam_unix.so* para una autenticación legada basada en NSS. En la figura 2.5 se resumen las opciones de autenticación disponibles en Linux.

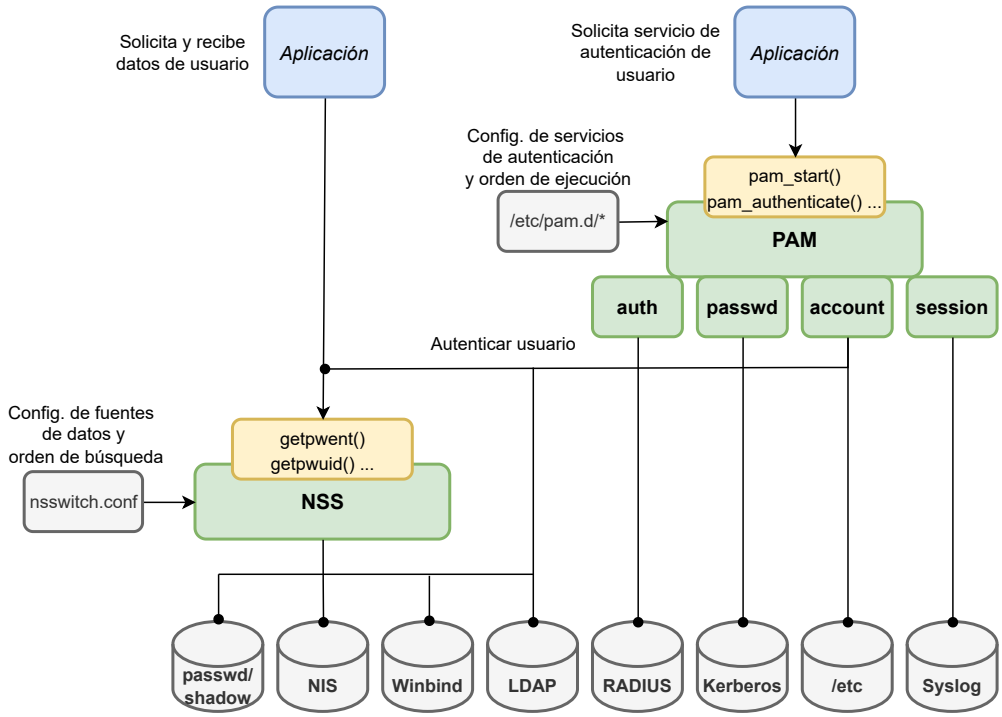

Figura 2.5: Autenticación en Linux

A menudo, la suplantación de identidad es utilizado por las aplicaciones o demonios del sistema que, a medida que los usuarios se conectan a través de la red, necesitan asumir los privilegios de los mismos. Las llamadas al sistema estándar *setuid()* y *setgid()* permiten a un proceso suplantar la identidad de un usuario y un grupo respectivamente.

Cuando un usuario lanza un ejecutable, el UID y el GID se asignan a los atributos Real User ID (RUID) y Real Group ID (RGID) respectivamente del proceso invocado. El *setuid* y el *setgid* son dos permisos especiales en cada ejecutable que indican al sistema operativo si debe utilizar el UID y GID actual o cambiar al UID

y GID del dueño del ejecutable.

Si un usuario lanza un ejecutable con los permisos *setuid* o *setgid* configurados, el RUID y RGID siguen siendo asignados al UID y GID del usuario actual, pero ahora se utiliza un nuevo conjunto de privilegios conocidos como Effective User ID (EUID) y Effective Group ID (EGID). Estos últimos son los que permiten al usuario asumir los permisos del dueño del ejecutable mientras esté en ejecución el proceso.

Un programa típico que utiliza los permisos *setuid* es la herramienta *passwd* que permite a un usuario modificar su contraseña; acción que requiere permisos de superusuario. El dueño del programa *passwd* es el usuario root, por lo tanto se encuentra configurado como *setuid-root*.

Otra herramienta que utilizan los administradores y demonios del sistema es la utilidad **su** que permite suplantar la identidad de cualquier usuario. Al igual que *passwd*, el comando *su* tiene configurado el flag *"s"* de setuid y pertenece al usuario root:

```
[root@pc1 ~]# ls -la /bin/su
  -rwsr-xr-x 1 root root 59740 May 25 10:49 /bin/su
```

2.6.2. Autenticación en Windows

Windows utiliza el término *security principal* para hacer referencia a un sujeto de seguridad que puede acceder a recursos del sistema (objetos de seguridad). Se considera que los usuarios, grupos y computadoras son *security principals*. Los archivos, impresoras, dispositivos, puertos y otros recursos que son accedidos, se consideran objetos.

Un *security principal* se identifica de manera única con un **Security Identifier (SID)**, elemento fundamental para la autenticación y autorización en Windows. Existen *security principals* de sistema que tienen SIDs bien conocidos y que son iguales en todas las instalaciones de Windows. Estos SIDs son locales a la computadora donde fueron creados y no se propagan a través de la red. Esto difiere de los SIDs de usuario y grupo que requieren unicidad en todo el dominio. En la figura 2.6 se observa el formato de un SID.

Windows provee de dos tipos diferentes de cuentas de usuario:

Built-in accounts Son creadas automáticamente por el sistema operativo y no pueden ser eliminadas. Las dos cuentas más importantes que pertenecen a esta categoría son la cuenta **Administrator** (S-1-5-domainID-500), que posee privilegios completos sobre el sistema, y la cuenta **Guest** (S-1-5-domainID-501) que tiene privilegios limitados y se encuentra deshabilitada por defecto en sistemas con Windows 2000 y posteriores.

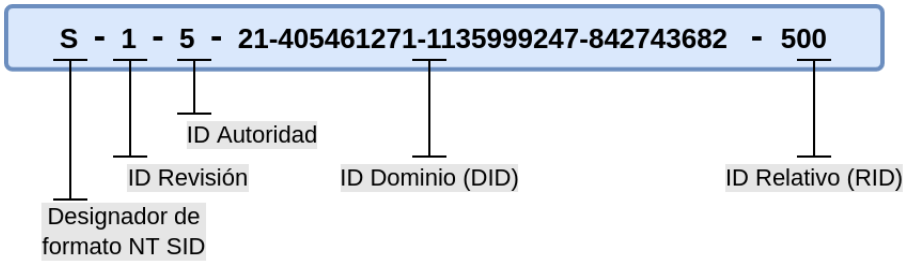

Figura 2.6: Formato de un SID

User-defined accounts Son creadas por el administrador del sistema para cada usuario que necesita acceder al mismo (S-1-5-domainID-RID).

Los tipos de grupos en Windows son los siguientes:

System groups Son creados automáticamente por el sistema operativo y la membresía a los mismos es automática y dinámica, dependiendo de las acciones del usuario en el sistema. *Everyone* (S-1-1-0), *Local* (S-1-2-0), *Creator owner ID* (S-1-3-0), *Creator group ID* (S-1-3-1), *Network* (S-1-5-2), *Interactive* (S-1-5-4) y *Authenticated Users* (S-1-5-11) son ejemplos de grupos que pertencen a esta categoría.

Built-in groups Son creados automáticamente por el sistema operativo durante la instalación. Contienen cuentas importantes para el normal funcionamiento del modelo de seguridad del sistema. *Administrators* (S-1-5-32-544), *Users* (S-1-5-32-545), *Guests* (S-1-5-32-546) y *Domain Admins* (S-1-5-domainID-512) son ejemplos de grupos que pertencen a esta categoría.

User-defined groups Son creados por el administrador para contener usuarios regulares del sistema y potencialmente otros grupos (S-1-5-domainID-RID).

Debido a que la parte *domainID* del SID es única para un dominio particular o computadora dentro del universo, y que la parte Relative Identifier (RID) es única dentro del dominio o computadora, el SID es también único para todo el universo.

Cuando un usuario se autentica el proceso de inicio de sesión devuelve un **token de acceso** compuesto por el SID del usuario, los SIDs de grupo a los que pertenece y los derechos o privilegios del mismo.

El token de acceso se asocia a cada proceso o subproceso que se ejecuta en nombre del usuario. Cada vez que éste interactúa con un objeto o realiza una tarea que requiere de derechos de usuario, el sistema operativo verifica el token de acceso asociado para determinar el nivel de autorización.

Los *derechos de usuario* son permisos asociados al usuario o grupo para permitir realizar tareas específicas a nivel de sistema operativo, como reiniciar el sistema, cambiar la hora, realizar tareas de respaldo y/o permitir iniciar sesión en el sistema. Se distinguen dos tipos de derechos de usuario: (1) *privilegios*, destinados a proveer al usuario de permisos para realizar tareas administrativas en el sistema, y (2) *derechos de logon*, utilizados para permitir el inicio de sesión.

Además de los derechos de usuario, Windows utiliza los *permisos de recursos* para determinar el nivel de acceso de un usuario a un recurso. Un recurso posee un Access Control List (ACL) compuesto de uno o más Access Control Entry (ACE), que proveen permisos discretos para permitir o denegar acciones especificas de los usuarios o grupos sobre dicho recurso. A diferencia de los derechos de usuario, que se almacenan en el token de acceso y son asignados por el Administrador del sistema, los permisos se almacenan donde reside el recurso u objeto y son definidos a discreción del propietario del mismo (Discretionary Access Control (DAC)) [22].

En la figura 2.7 se ilustra el proceso de autorización en Windows y de que manera se relacionan el token de acceso asignado al usuario y el descriptor de seguridad asociado al objeto para controlar el acceso a los recursos del sistema.

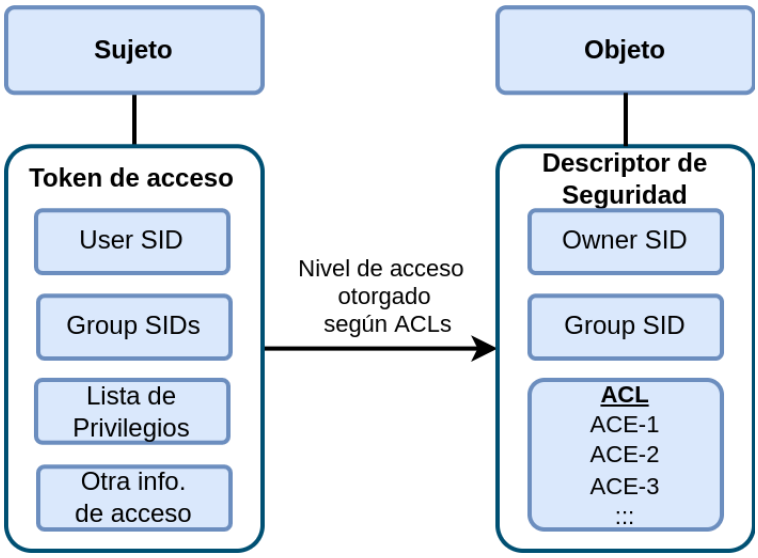

Figura 2.7: Autorización en Windows

Las cuentas de usuario y grupos que son creadas localmente en una computadora con Windows son almacenadas y administradas por el SAM. Dichas cuentas pueden ser utilizadas para acceder únicamente a recursos locales. La base de datos SAM se almacena en la porción del registro HKEY_LOCAL_MACHINE\SAM

y contiene, entre otros, los siguientes atributos de una cuenta de usuario:

- Username
- Password
- User SID
- Primary Group ID
- Full Name
- Home Drive
- Profile Path
- Logon Script
- Account active
- Last login time

Las cuentas y grupos que son creadas en un dominio con Active Directory (AD) son almacenadas y administradas por el controlador del dominio. Estas cuentas son almacenadas como objetos de directorio en una base de datos LDAP, y pueden ser utilizadas para acceder a cualquier recurso del dominio.

Existen dos escenarios distintos para la autenticación en Windows: (1) *login interactivo*, cuando el usuario inicia sesión en una computadora de manera local o remota a través del protocolo Remote Desktop Protocol (RDP); y (2) *login de red*, cuando el usuario intenta acceder a recursos a través de la red.

En un login interactivo, las credenciales de un usuario son validadas contra la base de datos SAM local o contra el servidor AD, en una computadora miembro de un dominio, a través del servicio *Winlogon*.

El servicio *Winlogon*, mediante la librería de identificación gráfica Graphical Identification and Authentication (GINA), inicia el proceso de login pasando las credenciales de usuario al subsistema de Autoridad de Seguridad Local (Local Security Authority (LSA)), que a su vez determina que mecanismo o paquete de autenticación utilizar (Security Support Provider (SSP)): NTLM, Kerberos, TLS/SSL, etc. [23].

El servicio *Netlogon* se utiliza en el caso de que la autenticación se realice contra un controlador de dominio.

Si la autenticación es exitosa, el servicio LSA devuelve el token de acceso que es utilizado por el servicio Winlogon para invocar al proceso *UserInit.exe* que

prepara el entorno inicial y ejecuta *Explorer.exe* para permitir el acceso interactivo del usuario al sistema.

La figura 2.8 ilustra los componentes necesarios para el proceso de autenticación en Windows.

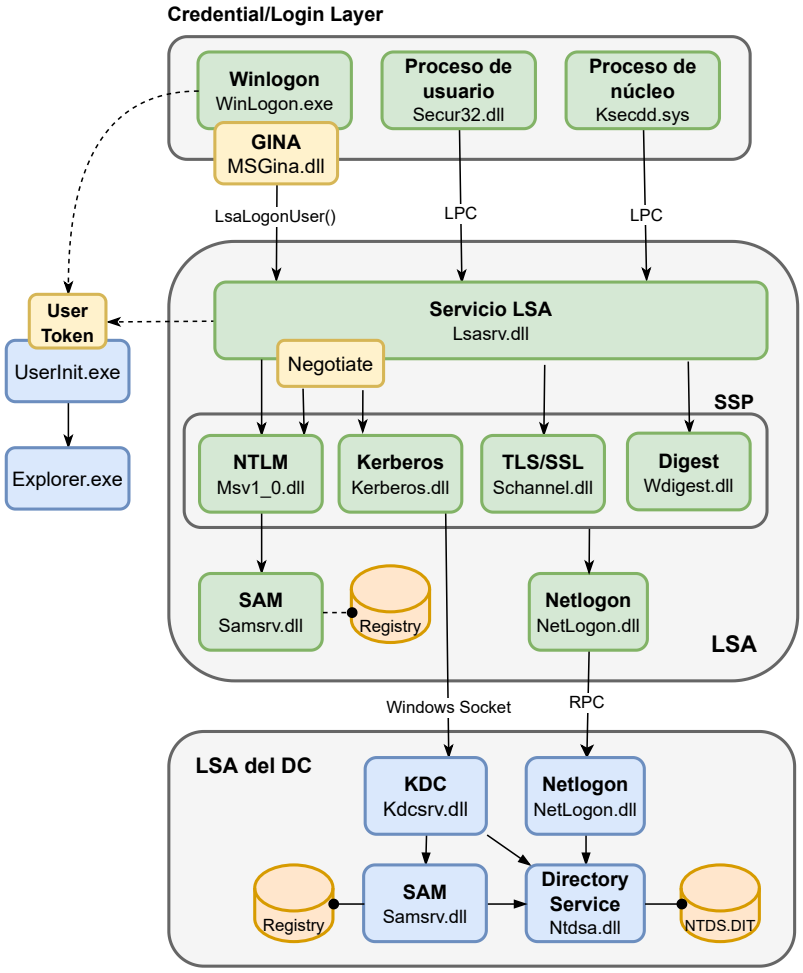

Figura 2.8: Autenticación en Windows

Cuando un usuario intenta acceder a un recurso a través de la red (login de red), sucede lo siguiente:

- El servicio de red al que se intenta acceder, solicita al usuario utilizar un mecanismo específico de autenticación. Algunas aplicaciones cliente, como el protocolo Server Message Block (SMB)/Common Internet File System (CIFS), intentan autenticarse sin esperar a que el servidor solicite la

autenticación; otras aplicaciones, como los navegadores web, pueden intentar conectarse primero de forma anónima.

- El proceso servidor pasa las credenciales del usuario al subsistema LSA, el cual selecciona el método de autenticación a utilizar similar al proceso de login interactivo.
- El subsistema LSA proporciona al servicio de red el token de acceso para el usuario autenticado.
- El servicio de red elige entre uno de dos enfoques típicos:
  - Suplantar la identidad del usuario utilizando el token de acceso proporcionado y delegar la responsabilidad de autorización para acceder a los recursos al sistema operativo local. Ver sección 2.5.1
  - Extraer del token de acceso los SID de usuario y grupo, y cuando el usuario intente acceder a un archivo o algún otro objeto proporcionado por el servicio, compararlos contra una lista de autorización específica de la aplicación. En este caso es responsabilidad del servicio de red controlar el acceso a los recursos que ofrece.

2.7. Autenticación de las Aplicaciones y Servicios

A continuación se describen los mecanismos de autenticación comúnmente utilizados por las aplicaciones y servicios de red. Además de los mecanismos de autenticación específicos para algunas aplicaciones, se describen primero las interfaces de programación estándar para la implementación segura de los servicios de autenticación.

2.7.1. GSS-API, SPNEGO y SSPI

La interfaz Generic Security Services API (GSS-API) es una API estándar de Internet diseñada para proporcionar a las aplicaciones una capa de abstracción para los servicios de autenticación y protección de datos [24]. Con GSS-API las aplicaciones no necesitan implementar sus propios mecanismos de autenticación de usuario, o métodos de encriptación para el tráfico de red. Las aplicaciones simplemente pueden usar la librería de GSS-API disponible en C, Java y otros lenguajes, para integrar a la perfección los servicios de seguridad de red estándar de la industria de diferentes proveedores. La última versión de GSS-API se describe en [25].

La interfaz GSS-API ofrece cuatro tipos de portabilidad para las aplicaciones: 1- independencia de la plataforma o sistema operativo donde se ejecuta la aplicación, 2- independencia del protocolo que se utiliza para la comunicación entre las aplicaciones, 3- independencia del mecanismo de seguridad subyacente y 4- independencia del Quality of Protection (QoP) utilizando el mecanismo de seguridad por defecto de GSS-API.

En la figura 2.9 se ilustra la comunicación entre dos aplicaciones que utilizan GSS-API

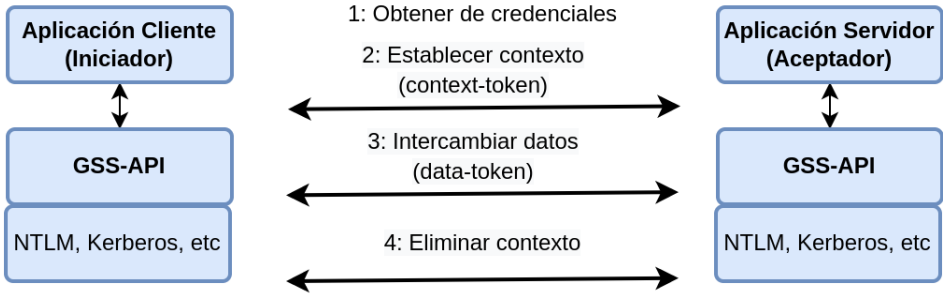

Figura 2.9: Autenticación con GSS-API

Las librerías de GSS-API en los dos hosts son las responsables de generar y procesar los *tokens*, unidades de mensaje para intercambiar información de seguridad. La aplicación es la responsable de transportar los tokens entre los dos pares utilizando su propio protocolo de comunicación. En consecuencia, los mecanismos de seguridad subyacentes son los encargados de proveer los servicios de seguridad a la aplicación: autenticación, integridad y confidencialidad (los dos últimos, opcionales y dependientes del mecanismo de seguridad utilizado).

Generalmente, las aplicaciones que utilizan GSS-API siguen los pasos a continuación:

1. Obtener credenciales: las aplicaciones deben obtener las credenciales del usuario; por ejemplo, luego de un login exitoso. Se utiliza la rutina `gss_acquire_cred()`

2. Establecer contexto de seguridad: las dos aplicaciones que se comunican deben establecer un contexto de seguridad intercambiando los tokens de GSS-API mediante las rutinas `gss_init_sec_context()` y `gss_accept_sec_context()`.

3. Proteger la comunicación de datos: la aplicación llama a las rutinas `gss_get_mic()` y `gss_wrap()` para porteger los datos y enviar el token resultante a su par. La aplicación par pasa el token recibido a las rutinas

`gss_verify_mic()` y `gss_unwrap()` para verificar la integridad y remover la protección respectivamente.

4. Eliminiar el contexto de seguridad: al final de la comunicación, cada aplicación llama a la rutina `gss_delete_sec_context()` para eliminar el contexto de seguridad establecido anteriormente.

Aunque GSS-API es independiente del mecanismo de autenticación en uso, las implementaciones para Linux son a menudo distribuidas con Kerberos. Por lo tanto, GSS-API es considerada la interfaz principal a nivel de aplicación para acceder a una autenticación basada en Kerberos. En [26] se describe la implementación GSS-API para la versión 5 del protocolo Kerberos. Su compatibilidad (parcial) con la interfaz Security Support Provider Interface (SSPI) la vuelve conveniente para ser usada en entornos heterogéneos con clientes Windows y servidores AD.

Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) es una interfaz utilizada por GSS-API para negociar un mecanismo de autenticación que se utilizará entre el cliente y el servidor, como se observa en la figura 2.10. SPNEGO no es un mecanismo de autenticación independiente, es un pseudo-mecanismo utilizado para negociar el mejor mecanismo de autenticación compatible con el cliente y el servidor. La última actualización del protocolo se describe en [27].

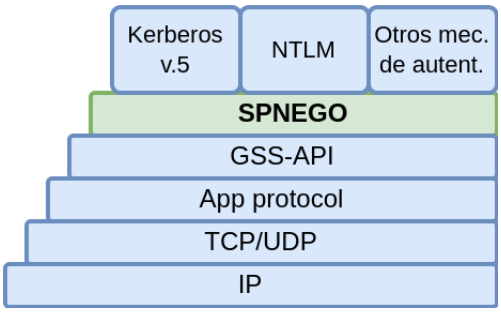

Figura 2.10: Pila de protocolos para SPNEGO

Una vez que SPNEGO ha negociado un mecanismo de autenticación común, comienza la transferencia de los tokens de autenticación entre el cliente y el servidor.

SPNEGO es compatible con Windows 2000 y sistemas operativos posteriores, donde funciona por encima de SSPI. Windows usa SPNEGO para negociar un mecanismo de autenticación para los servicios de archivos SMB/CIFS, Remote Procedure Call (RPC) de asociación directa y autenticación integrada de Windows Hyper Text Transfer Protocol (HTTP) para *Internet Information Server*. También

existe un módulo de SPNEGO para los servidores HTTP Apache y Nginx. Tanto Internet Explorer, como Mozilla Firefox y Chrome también son compatibles con SPNEGO. En la figura 2.11 se muestra una secuencia de mensajes típica para autenticar HTTP mediante SPNEGO. En este ejemplo el protocolo de autenticación seleccionado es Kerberos.

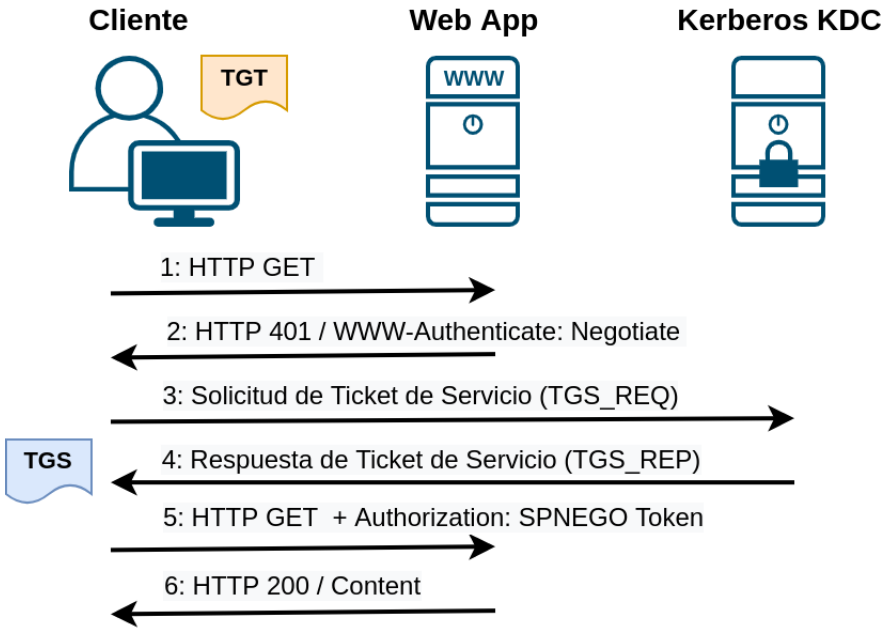

Figura 2.11: Autenticación HTTP utilizando SPNEGO

La interfaz SSPI de Windows es un componente del sistema operativo que provee a las aplicaciones de usuario y servicios del sistema una capa de abstracción para una comunicación segura en entornos de red distribuidos. SSPI fue desarrollado al mismo tiempo que GSS-API y es muy similar a éste. De hecho, ambas implementaciones son compatibles hasta cierto punto.

Los servicios básicos que provee SSPI son los siguientes:

- Gestión de credenciales: las aplicaciones pueden trabajar con nombres de usuario, contraseñas y tickets.
- Gestión de contexto: permite a los clientes y servidores preparase para realizar la autenticación, suplantación de identidad y finalmente volver al contexto original.
- Soporte de mensajes: provee firma y encriptación de mensajes.

- Gestión de paquetes: permite a las aplicaciones utilizar los paquetes SSP.

La interfaz SSPI de Windows soporta diferentes paquetes de seguridad que se conocen como SSP:

Negotiate SSP Implementa el protocolo SPNEGO y puede seleccionar y utilizar el paquete New Technology LAN Manager (NTLM) SSP o Kerberos SSP

NTLM SSP Maneja la autenticación LM, NTLMv1 y NTLMv2.

Kerberos SSP Administra la autenticación Kerberos [5].

Digest SSP Implementa el protocolo de autenticación Digest.

Secure Channel (SChannel) Maneja la autenticación TLS/SSL.

GSS-API y SSPI se desarrollaron en paralelo por la Internet Engineering Task Force (IETF) y Microsoft respectivamente. Por lo tanto, el primero es un estándar abierto y el segundo es propietario. Ambos proveen la misma funcionalidad, y aunque la implementación de las rutinas de la API pueden ser algo diferentes, tienen comportamientos similares en la red, permitiendo a las aplicaciones interoperar en escenarios específicos y fallar en otros.

Los mecanismos de protección de mensajes en términos de integridad y encriptación son implementados de manera diferente en GSS-API y SSPI, y por lo tanto no son compatibles. Sin embargo, el protocolo Kerberos es soportado por ambas API y generan las mismas claves de sesión según [5], por lo que pueden interoperar efectivamente en la autenticación de usuarios.

2.7.2. NTLM

NTLM es el nombre de una familia de protocolos utilizados como mecanismo de autenticación principalmente por los sistemas de Microsoft Windows y otros productos compatibles como SAMBA. Fue introducido como parte del protocolo SMB para compartir archivos entre computadoras LAN Manager y clientes de red Microsoft. Luego fue adoptado para la autenticación de usuarios de otras aplicaciones y protocolos como HTTP. Los detalles sobre el funcionamiento de NTLM no han sido publicados por Microsoft, aunque han contribuido de alguna manera en publicaciones como [28]. Una de las mejores fuentes que describe el protocolo es una documentación no oficial [29].

NTLM es un protocolo de autenticación del estilo desafío-respuesta. Esto significa que para autenticar a un usuario, el servidor envía un desafío al cliente. El cliente luego envía una respuesta que es una función del desafío y la contraseña del usuario (hash). El servidor puede validar la respuesta consultando una base de

datos de la cuenta para obtener la contraseña del usuario y calcular la respuesta adecuada para ese desafío. Para autenticar NTLM utiliza tres tipos de mensajes:

1. El cliente establece una conexión con el servidor y envía un *NEGOTIATE_MESSAGE* anunciando las características soportadas.

2. El servidor responde con un *CHALLENGE_MESSAGE* que contiene una lista de características admitidas y el desafío aleatorio generado por el servidor.

3. El cliente responde al desafío con un *AUTHENTICATE_MESSAGE* que contiene además información sobre el cliente, como el nombre de usuario y dominio al que pertenece.

NTLM utiliza uno o dos valores de *hash* para encriptar el desafío. Uno es el *hash LM* que encripta mediante el protocolo Data Encryption Standard (DES) la contraseña del usuario y el desafío. El otro es el *hash NT* que utiliza la función MD4. Ambos generan un valor de *hash* de 16 bytes que es almacenado en la base de datos SAM, para cuentas de usuario locales, o en el servidor AD para cuentas de dominio.

NTLM tiene tres versiones principales: LM, NTLMv1 y NTLMv2. El flujo de mensajes para los tres es el mismo, las únicas diferencias son la función utilizada para calcular varios campos de respuesta del desafío y qué campos de respuesta se configuran.

Además de la autenticación, el protocolo NTLM opcionalmente proporciona seguridad de sesión, específicamente integridad y confidencialidad de los mensajes mediante funciones de firma y sellado [30].

Como se describe en 2.7.1, la interfaz SSPI provee un paquete de seguridad para autenticar con NTLM. Las aplicaciones no deberían utilizar directamente dicho paquete; en su lugar, deben usar Negotiate SSP que permite a las aplicaciones aprovechar protocolos de seguridad más avanzados si son compatibles con los sistemas involucrados en la autenticación. Actualmente, el paquete Negotiate SSP selecciona entre Kerberos y NTLM. Por defecto, Negotiate SSP selecciona Kerberos a menos que no pueda ser utilizado por uno de los sistemas involucrados en la autenticación.

El paquete Kerberos SSP agrega mayor seguridad en la autenticación que NTLM. Aunque Kerberos es el protocolo por elección, NTLM aún se sigue utilizando para la autenticación en sistemas *stand-alone*.

2.7.3. Kerberos

El protocolo de autenticación Kerberos fue desarrollado por el Massachusetts Institute of Technology (MIT) en la década de 1980 como parte del proyecto At-

hena. Las primeras tres versiones del protocolo fueron internas. La versión 4 se liberó al público y ganó popularidad rápidamente. Sin embargo, debido a fallas de seguridad en su diseño, enseguida se liberó la versión 5 que se convirtió en el estándar de facto para la autenticación de usuarios en entornos de red heterogéneos [5].

Además de la implementación del MIT, disponible para sistemas Linux, Windows y otras plataformas, existen otras implementaciones como Heimdal y Sun Kerberos que forma parte de la suite de Java. Microsoft Windows Active Directory también proporciona una implementación Kerberos compatible con [5]. Aunque difiere de las implementaciones de MIT y Heimdal en términos de enfoque y herramientas, aún puede interactuar con ellas en la mayoría de los escenarios [8].

Kerberos es más que un protocolo de autenticación: provee un *modelo* de sistema para la autenticación y posterior autorización en entornos de computación distribuida y orientada a pares. Los aspectos más importantes a resaltar del modelo son:

- Opera dentro de sistemas distribuidos basados en un modelo de Internet abierto
- Se integra con las tecnologías de red existentes como la suite TCP/IP
- Separa el servicio de autenticación del resto de los servicios
- Centraliza la administración de las contraseñas de usuario
- Utiliza métodos criptográficos para operar en entornos hostiles y nunca exponer la contraseña del usuario durante la autenticación
- Las partes involucradas (clientes y servidores) establecen una relación de confianza con el servidor de Kerberos (KDC), y se apoyan en éste para autenticarse.

La figura 2.12 muestra un esquema simplificado del modelo de autenticación de Kerberos

Kerberos se basa en el modelo de autenticación presentado por [31]. Marcas de tiempo o *timestamps* se agregaron al modelo original para evitar los ataques de repetición. A diferencia de NTLM, que mantiene conexiones TCP permanentes entre las partes para enviar información sensible, Kerberos construye relaciones de confianza sin estado mediante el intercambio de **tickets**.

Los tickets son estructuras de datos criptográficamente seguras que contienen información de autenticación. Los mismos son solicitados por el cliente al KDC cuando éste los necesita, se guardan en caché y posteriormente son presentados a los servicios y aplicaciones para obtener el acceso correspondiente. Por lo tanto

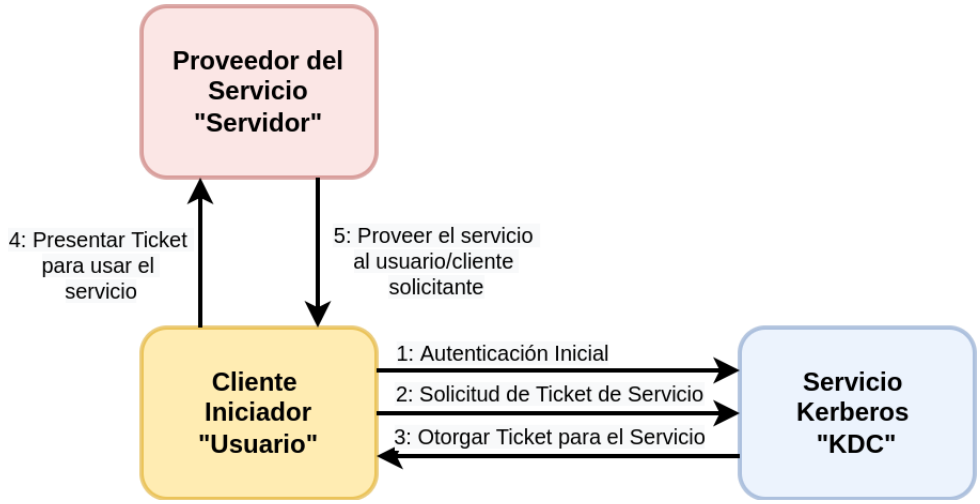

Figura 2.12: Esquema simplificado del modelo de autenticación de Kerberos

Kerberos no necesita construir ni mantener conexiones y estados de autenticación con las partes.

Existen dos tipos de tickets: el Ticket Granting Ticket (TGT) y el Ticket Granting Service (TGS).

Durante la primera fase de autenticación con el KDC, el cliente obtiene el TGT que se utiliza como token de acceso primario para la obtención de los demás tickets de servicio. Luego, el cliente solicita al KDC el TGS correspondiente para cada servicio o aplicación que desea acceder. Mediante el TGS el servicio o aplicación puede verificar la autenticidad del cliente sin contactar con el KDC.

El lugar donde se almacenan los tickets depende del sistema operativo del cliente y su configuración. Por lo geneneral, se guardan como pequeños archivos en una carpeta temporal: */tmp/krb5cc_**.

Además de los tickets, el protocolo Kerberos utiliza un segundo tipo de credencial llamado *autenticador*. A diferencia de los tickets, el autenticador puede ser usado sólo una vez: uno nuevo debe ser generado cada vez que el cliente accede a un servicio. Esto no representa un problema ya que el propio cliente puede construir el autenticador luego de obtener la *clave de sesión* correspondiente. Las claves de sesión son claves temporales que se utilizan durante el tiempo de vida de los tickets para intercambiar mensajes entre las partes de manera segura.

La estructura de datos de los tickets y del autenticador viene dado por:

```
Tickets:
T(k,c) = {k, c, addr, timestamp, life, K(k,c)}Kk = TGT
T(s,c) = {s, c, addr, timestamp, life, K(s,c)}Ks = TGS
```

```
Autenticador:
As = {c, addr, timestamp}K(s,c)

donde:
c  -> nombre del cliente/usuario
s  -> nombre del servicio/servidor
k -> nombre del servidor Kerberos
addr  -> dirección de red de la máquina del cliente
life  -> tiempo de vida del ticket
Kx  -> Clave secreta de 'x'
K(x,y) -> Clave de sesión para 'x' e 'y'
{abc}Kx -> 'abc' encriptado con la clave secreta de 'x'
T(x,y) -> ticket de x para ser usado por y
Ax  -> autenticador para x
```

El componente más importante en la infraestructura de Kerberos es el servidor KDC. El mismo se constituye de tres partes:

- un servidor de autenticación (Authentication Server (AS)), que responde a las solicitudes de autenticación de los clientes y devuelve el TGT.
- un servidor de TGS, que genera los tickets TGS los cuales son utilizados por los clientes para acceder a los servicios y aplicaciones kerberizadas.
- una base de datos, que almacena las claves secretas de los clientes y servidores e información relacionada a las cuentas de Kerberos (fecha de creación, tiempo de vida del ticket, política de contraseñas, etc.)

Los tickets tienen un tiempo de vida que determina el período de validez de los mismos. Durante todo ese tiempo pueden ser utilizados sin necesidad de volver a comunicarse con el KDC, por lo que constituyen un elemento importante para la implementación de un sistema SSO.

La contraseña del usuario se solicita por única vez durante el inicio de sesión. A partir de dicha contraseña se deriva la *clave secreta* del cliente, que va a permitir obtener el TGT para luego ser utilizado como credencial de acceso durante el resto de la sesión. A diferencia de las claves de sesión, las claves secretas son perdurables en el tiempo y dependen de la política de contraseñas establecida en el KDC.

Los servicios o aplicaciones kerberizadas también necesitan una clave secreta con la cual se encripta el TGS. Dicha clave es compartida con el KDC y se almacena, del lado del servicio, en un archivo especial denominado **keytab**. El keytab debe ser accesible únicamente por el proceso que ejecuta el servicio en cuestión.

Por último, el propio servidor KDC necesita una clave secreta para encriptar los TGT. La clave secreta del KDC se guarda en la misma base de datos donde se encuentran las claves secretas de los usuarios y servicios. Para proteger las claves secretas, la base de datos se encuentra encriptada con una clave maestra que se genera al crear el *Realm* o dominio de Kerberos. Para evitar interacción humana cada vez que inicia el servicio de Kerberos, dicha clave se guarda en un archivo especial denominado *stash*.

Las partes involucradas en la autenticación con Kerberos (clientes y servicios) se conocen como **principals**. El nombre utilizado para identificar a los principals tiene el siguiente formato: *name/instance@Realm*.

La parte *name* representa el nombre principal del usuario o servicio. La parte *instance* se utiliza para distinguir un privilegio especial (root, admin, etc) de un usuario, o el nombre de la máquina donde se ejecuta el servicio. La parte *Realm* identifica al dominio de Kerberos donde residen los principals. Por convención, el Realm corresponde al nombre de dominio DNS en letras mayúsculas.

Resumiendo, se puede dividir el funcionamiento de Kerberos en tres etapas o fases principales:

1. Proceso de autenticación, donde el cliente obtiene el TGT y la clave de sesión K(k,c) mediante el intercambio de los mensajes AS_REQUEST y AS_REPLY con el KDC:

   ```
   c --> KDC(AS): AS_REQUEST = c,k,timestamp
   c <-- KDC(AS): AS_REPLY = {K(k,c),life,k,TGT}Kc
   ```

   Esta fase esta presente durante el proceso de inicio de sesión del usuario y cada vez que es necesario renovar el TGT. Del proceso de autenticación se obtiene la clave secreta del cliente (Kc) a partir de la contraseña proporcionada por el usuario. Dicha clave se utiliza únicamente durante esta fase para desencriptar el mensaje AS_REPLY y extraer el TGT y la clave de sesión K(k,c).

2. Proceso de solicitud de servicio, donde el cliente obtiene el TGS y la clave de sesión correspondiente mediante el intercambio de los mensajes TGS_REQUEST y TGS_REPLY con el KDC:

   ```
   c --> KDC(TGS): TGS_REQUEST = s,TGT,Ak
   c <-- KDC(TGS): TGS_REPLY = {K(s,c),life,s,TGS}K(k,c)
   ```

   Antes de responder, el KDC verifica la autenticidad de la solicitud comparando la información contenida en el TGT y el autenticador Ak. Para ello, primero desencripta el TGT con su clave secreta Kk y luego extrae la clave

se sesión K(k,c), previamente compartida con el cliente, para poder desencriptar el autenticador. La respuesta del KDC contiene una nueva clave de sesión K(s,c) y el TGS cuyo contenido también incluye la clave K(s,c) pero encriptada con la clave secreta del servicio Ks.

3. Acceso al Servicio, donde el cliente utiliza el TGS para autenticarse y obtener acceso al servicio solicitado mediante el intercambio de los mensajes AP_REQUEST y AP_REPLY:

```
c --> s: AP_REQUEST = TGS,As
c <-- s: AP_REPLY = {timestamp+1}K(s,c)
```

   Antes de conceder el servicio solicitado se verifica la autenticidad del cliente comparando la información contenida en el TGS y el autenticador As. Para ello, primero se desencripta el TGS con la clave secreta del servidor Ks y luego se extrae la clave se sesión K(s,c) para poder desencriptar el autenticador. La respuesta del servidor contiene el timestamp enviado por el cliente dentro del autenticador, encriptado con la clave de sesión compartida entra ambos. De esta manera el cliente también puede verificar la autenticidad del servicio.

Por lo general, la fase 1 se realiza una sola vez durante el proceso de inicio de sesión del usuario y las fases 2 y 3 se repiten tantas veces como servicios o aplicaciones se necesita acceder. Si el usuario ya pose un TGS válido para un servicio utilizado con anterioridad lo puede volver a utilizar en la fase 3 sin necesidad de repetir la fase 2. En la figura 2.13 se grafica con mayor de detalle el intercambio de mensajes que realiza Kerberos para las tres fases descriptas con anterioridad.

Un potencial problema que se puede presentar en la fase 1 es el ataque de *texto plano conocido* contra el KDC. Para ello, el atacante genera una serie de solicitudes cuyo contenido conoce y espera por las respuestas encriptadas del KDC para ser analizadas y deducir la clave secreta del usuario. Para evitar tal ataque, el KDC puede requerir al cliente que se autentique antes de enviar el AS_REPLY. Esto se conoce como *preautenticación* y existen diferentes métodos para implementarlo. El más utilizado es el PA-ENC-TIMESTAMP donde el cliente encripta con su clave privada Kc el timestamp enviado en el mensaje AS_REQUEST. De esta manera el KDC puede verificar la autenticidad de la solicitud antes de responder.

Kerberos efectivamente usa la hora actual de los clientes y servidores (timestamps) como parte del proceso de autenticación y para evitar ataques de repetición. Esto requiere que todas las partes se encuentren sincronizadas en tiempo ya que superado un sesgo mínimo (por defecto de 5 minutos) la autenticación fallará. Es común utilizar el servicio Network Time Protocol (NTP) para mantener a los clientes y servidores sincronizados.

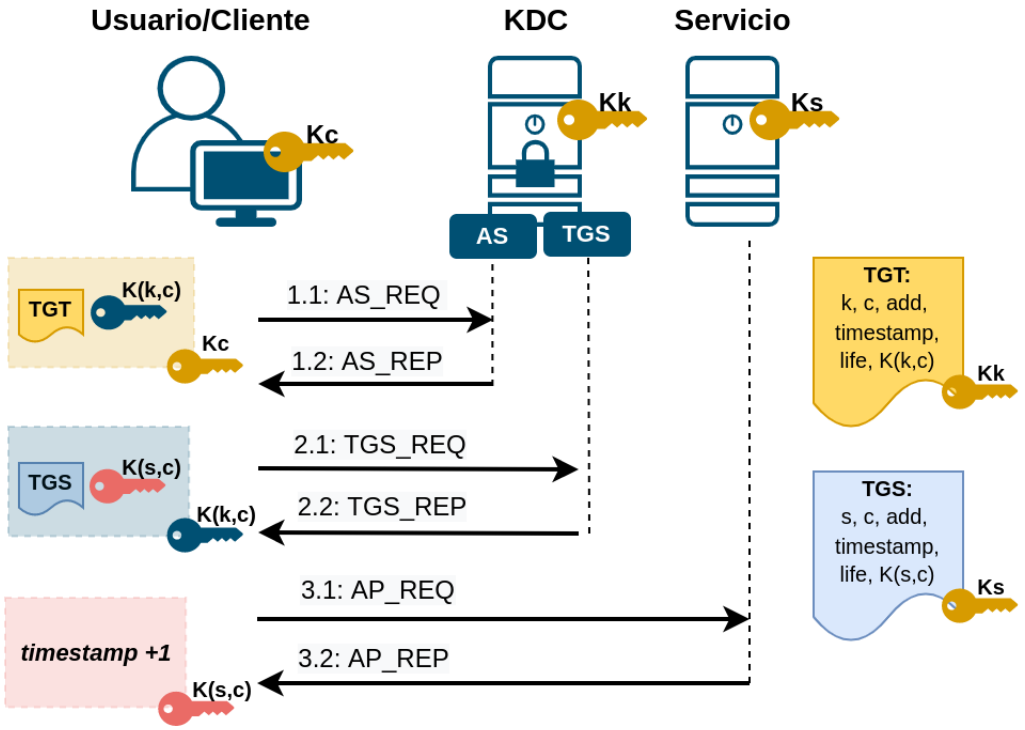

Figura 2.13: Intercambio de mensajes para Kerberos

Otro servicio que se recomienda utilizar para un funcionamiento correcto de Kerberos es el DNS. Es importante que la parte *instance* del nombre de los principals de servicio coincida con el nombre completo de la máquina (Fully Qualified Domain Name (FQDN)) configurado en el servidor DNS. Los clientes de Kerberos por defecto realizan una consulta de DNS directa y otra inversa con el fin de verificar el nombre correcto de los principals de servicio antes de solicitar el TGS correspondiente.

2.7.4. SASL

Simple Authentication and Security Layer (SASL) es un framework que provee autenticación y seguridad de datos (confidencialidad e integridad) a protocolos orientados a la conexión a través de una interfaz estructurada para acceder a mecanismos de autenticación intercambiables. SASL fue primero definido en [32] y luego actualizado en [33] y [34].

El diseño de SASL permite a los nuevos protocolos de aplicación reutilizar los mecanismos de autenticación existentes sin requerir una modificación de los mismos. Además, permite a los protocolos existentes hacer uso de nuevos meca-

nismos sin un rediseño de los protocolos. En la figura 2.14 se puede observar como SASL provee una capa de abstracción entre los protocolos de aplicación y los mecanismos de autenticación, similar a lo que ocurre con las interfaces GSS-API y SSPI vistas en la sección 2.7.1.

Los protocolos de aplicación que utilizan SASL a menudo también soportan TLS para complementar los servicios ofrecidos por SASL

Figura 2.14: Capa de abstracción SASL

Cada mecanismo SASL tiene un nombre simbólico bien conocido, que los protocolos de aplicación utilizan para negociar la autenticación. A continuación se describen alguno de estos nombres más utilizados:

- EXTERNAL, la autenticación se encuentra implícita dentro del contexto de la aplicación: por ejemplo para protocolos que ya se encuentran utilizando IPsec o TLS.
- ANONYMOUS, para un acceso invitado sin autenticación.
- PLAIN, mecanismo simple que utiliza password en texto claro según lo definido en [35].
- OTP, mecanismos de autenticación mediante passwords de un sólo uso. Deja obsoleto al antiguo mecanismo SKEY.
- CRAM-MD5, esquema sencillo de desafío-respuesta basado en HMAC-MD5.
- DIGEST-MD5, esquema de desafío-respuesta basado en MD5 y que además ofrece seguridad en el transporte de datos de la aplicación.

- NTLM, mecanismo de autenticación NTLM
- GSSAPI, para autenticación con Kerberos V5 via GSS-API y con soporte para protección de datos de la aplicación.
- OAUTHBEARER, utiliza bearer tokens OAuth 2.0 a través de TLS

La negociación de autenticación SASL es utilizado por protocolos orientados a la conexión donde existe una sesión interactiva entre el cliente y el servidor, como por ejemplo: Post Office Protocol 3 (POP3), Internet Message Access Protocol (IMAP), Simple Mail Transfer Protocol (SMTP) y LDAP. Por lo general, la negociación de SASL involucra los siguientes pasos:

1. Inicialización: intercambio de mecanismos de autenticación soportados entre el cliente y el servidor.
2. Autenticación: selección del mecanismo de autenticación a utilizar e intercambio de mensajes desafío-respuesta según el mecanismo seleccionado. Si la autenticación es exitosa continua con el siguiente paso.
3. Integridad y privacidad: opcional en caso de que el mecanismo seleccionado soporte la protección de datos de la aplicación.
4. Liberación: cierre de la sesión SASL. Sin embargo, dependiendo del mecanismo utilizado, la comunicación entre el cliente y servidor puede continuar.

2.7.5. Autenticación en LDAP

El protocolo LDAP es ampliamente utilizado como servicio de directorio en las organizaciones, por lo que requiere de un debido control de acceso y autenticación.

Un directorio es un conjunto de objetos con atributos organizados de una manera lógica y jerárquica. Estos objetos suelen representar unidades organizativas, personas, usuarios, grupos, computadoras, impresoras, etc. Además, LDAP se a transformado en el punto central de las organizaciones para administrar las identidades y la autenticación de usuarios. Microsoft Active Directory y OpenLDAP son ejemplos de servidores LDAP que pueden trabajar en conjunto dentro de una organización.

La versión actual del protocolo es LDAPv3, y se encuentra definido en [36]. Allí se especifica que la autenticación debe realizarse utilizando el mecanismo nativo de LDAP de autenticación simple, mecanismos SASL o mediante mapeo de certificados SSL. También, se recomienda proteger con SSL/TLS el canal de comunicación entre el cliente y servidor LDAP.

A diferencia de POP3/IMAP, HTTP y SMTP, LDAP no es un protocolo basado en el diálogo en texto claro. El cliente y el servidor intercambian mensajes utilizando estructuras de datos definidas en Abstract Syntax Notation One (ASN.1) y codificadas en formato binario Basic Encoding Rules (BER). Existen herramientas que permiten interactuar con el servidor mediante un formato más amigable conocido como Lightweight Directory Interchange Format (LDIF).

Un cliente se conecta con un servidor LDAP por defecto en el puerto TCP 389 o 636 en el caso de utilizar ldaps://. Luego, el cliente envía una petición de operación al servidor, y el servidor envía respuestas. El cliente puede requerir las siguientes operaciones:

- Start TLS: usar la extensión TLS de LDAPv3 para una conexión segura en lugar de ldaps://
- Bind: autenticarse y especificar una versión del protocolo LDAP
- Search: buscar y obtener entradas de directorio
- Compare: probar si una entrada nombrada contiene un valor de atributo dado
- Add: Añadir una nueva entrada
- Delete: Borrar una entrada
- Modify: Modificar una entrada
- Modify Distinguished Name (DN): renombrar el nombre de una entrada
- Abandon: abortar una petición previa
- Extended Operation: operación genérica usada para definir otras operaciones
- Unbind: cerrar la sesión LDAP

Los mensajes del tipo Bind contienen los campos necesarios para incorporar los datos de los diferentes mecanismo de autenticación soportados por LDAP. Por ejemplo, en el mecanismo de autenticación simple el mensaje Bind contiene el campo DN y el campo password utilizados para enviar el nombre de usuario y clave en texto claro al servidor. En una autenticación anónima el cliente envía dichos campos con el valor NULL. La autenticación anónima es un requisito en las especificaciones de LDAP para poder conectarse por primera vez al servidor y consultar las capacidades del mismo accediendo a la raíz del directorio *RootDSE*:

```
$ldapsearch -h localhost -x -D "" -w "" -b "" -s base "(objectclass=*)" * +

# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectclass=*)
# requesting: addou.ldif adduser.ldif editgroup.ldif kdc.cap kdc-spnego.cap \\
krb5.keytab ssh.debug user-template.ldif +
#
dn:
structuralObjectClass: OpenLDAProotDSE
configContext: cn=config
namingContexts: dc=lab,dc=net
supportedControl: 2.16.840.1.113730.3.4.18
supportedControl: 2.16.840.1.113730.3.4.2
supportedControl: 1.3.6.1.4.1.4203.1.10.1
...
supportedExtension: 1.3.6.1.4.1.4203.1.11.1
supportedExtension: 1.3.6.1.4.1.4203.1.11.3
supportedExtension: 1.3.6.1.1.8
supportedFeatures: 1.3.6.1.1.14
supportedFeatures: 1.3.6.1.4.1.4203.1.5.1
supportedFeatures: 1.3.6.1.4.1.4203.1.5.2
supportedLDAPVersion: 3
supportedSASLMechanisms: DIGEST-MD5
supportedSASLMechanisms: CRAM-MD5
supportedSASLMechanisms: NTLM
entryDN:
subschemaSubentry: cn=Subschema
# search result
search: 2
result: 0 Success
# numResponses: 2
# numEntries: 1
```

En este caso, el atributo *supportedSASLMechanisms* indica que el servidor soporta los mecanismos de autenticación SASL DIGEST-MD5, CRAM-MD5 y NTLM, además de la autenticación simple. La opción -x en el comando ldapsearch obliga a utilizar la autenticación simple en lugar de SASL. Las opciones -D y -w al estar vacías implican una autenticación anónima al RootDSE.

Para utilizar LDAP como servidor de autenticación se agrega a los objetos de tipo usuario el atributo **userPassword**. Los passwords pueden o no almacenarse en texto claro. En caso de almacenarse encriptados, se almacena también como prefijo el esquema de encriptación utilizado. Por ejemplo, un password almacenado como *Salted SHA1* se vería así:

```
userPassword: {SSHA}DkMTwBl+a/3DQTxCYEApdUtNXGgdUac3
```

En el caso servidores Microsoft Active Directory, se utiliza el atributo **UnicodePWD** en lugar de userPassword.

2.7.6. Autenticación en SSH

El protocolo Secure SHell (SSH) fue desarrollado para superar las limitaciones de los protocolos de terminal remoto, como *Telnet* y *Rlogin*, que carecen de sesiones y autenticación encriptadas. Actualmente existen dos versiones: SSHv1 y SSHv2. La principal diferencia entre ambas versiones es la forma en que son generadas las claves de sesión.

En concreto, SSH crea un túnel protegido (encriptado y autenticado) entre un cliente y un servidor sobre una conexión TCP en el puerto 22. El servidor posee un par de claves pública y privada que son utilizadas por el cliente para verificar su identidad, similar a lo que sucede entre un cliente y un servidor HTTP que utilizan TLS/SSL. Luego, las partes negocian una clave de sesión y los mecanismos a utilizar para la encriptación y autenticación del usuario.

La arquitectura de SSH definida en [37] especifica tres capas funcionales principales como se observa en la figura 2.15:

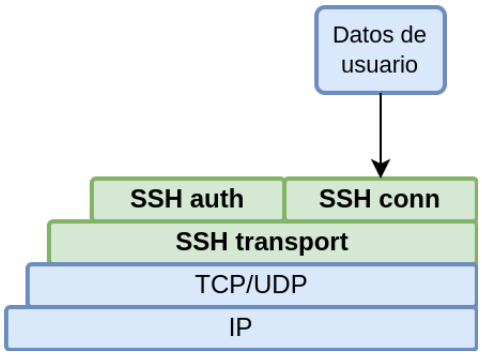

Figura 2.15: Capas del protocolo SSH

1. Capa de transporte SSH: Se utiliza el par de claves del host para autenticar el servidor y proteger el canal de comunicación mediante el intercambio de claves de sesión. Las otras capas de SSH dependen de esta capa. Una vez establecida la conexión de la capa de transporte SSH se obtiene el identificador único de la sesión (*sessionID*).

2. Capa de autenticación de usuario: Realiza la autenticación de usuario utilizando alguno de los siguientes mecanismos: clave pública, password, host o ninguno. En [38] se describen los métodos para utilizar GSS-API en la autenticación e intercambio de claves de SSH.

3. Capa de conexión: Multiplexa varios canales lógicos en una única conexión

o túnel SSH. Las aplicaciones utilizan estos canales para enviar los datos de manera segura.

Resulta interesante comparar SSH con TLS/SSL. Si bien ambos protocolos autentican y protegen los datos de las aplicaciones mediante criptografía de clave pública, su uso se enfoca a contextos bien diferenciados.

TLS/SSL actúa como una subcapa entre los protocolos de aplicación y la capa de transporte para proporcionar la seguridad necesaria a las aplicaciones que carecen de la misma, como HTTP, POP3, IMAP, SMTP y LDAP. Estos protocolos agregan la 's' al final de sus nombres para identificar de que se trata de la versión segura de los mismos y que utilizan un puerto TCP diferente al original.

La autenticación en TLS/SSL es del tipo M2M donde el servidor y el cliente intercambian certificados X.509 para verificar sus identidades. Se requiere una infraestructura PKI donde un CA valide los certificados emitidos para clientes y servidores.

En cambio, SSH por lo general no utiliza certificados X.509 y por ende, las claves públicas no son validadas contra un CA. Para operar el cliente SSH consulta al usuario si confía en la clave pública entregada por el servidor la primera vez que conecta, y en caso de aceptar, dicha clave se almacena para ser utilizada en futuras conexiones sin requerir una nueva interacción del usuario. En linux las claves públicas de los servidores se guardan en el archivo */etc/ssh/known_hosts* y en *∼/.ssh/known_hosts* del directorio personal de cada usuario.

De la misma manera, la autenticación del usuario se puede realizar sin interacción de éste utilizando el mecanismo de clave pública. Para ello, se almacenan en el servidor las claves públicas de los usuarios autorizados en el archivo *∼/.ssh/authorized_keys* del directorio personal de cada uno. Cuando el cliente SSH realiza la autenticación del usuario mediante clave pública envía al servidor un hash encriptado con la clave privada del usuario, que el propio servidor podrá desencriptar con la clave pública previamente almacenada en *∼/.ssh/authorized_keys*, y así verificar su identidad.

El mensaje que el cliente SSH envía al servidor para la autenticación de usuarios contiene los siguientes campos según [39]:

SSH Message ID (Type) SSH_MSG_USERAUTH_REQUEST (50)

Username El nombre del usuario que está solicitando la autenticación.

Service Name Nombre del servicio para el cual el usuario se está autenticando, ya que una misma conexión SSH puede multiplexar varios servicios sobre su capa de transporte.

Authentication method Nombre del método de autenticación a utilizar: gssapi, publickey, password, etc.

Authentication message Mensaje específico dependiendo del método de autenticación seleccionado.

2.7.7. Autenticación en SMB/CIFS

El protocolo SMB/CIFS es utilizado principalmente en entornos Windows para acceder a archivos e impresoras compartidos a través de la red. También se lo utiliza como mecanismo de comunicación entre procesos (Inter-Process Communication (IPC)), como las llamadas Microsoft/Distributed Computing Environment (DCE) RPC. Además de la implementación de Microsoft, la suite de programas SAMBA implementa el protocolo SMB/CIFS para entornos Linux compatible con Windows.

El servicio de sesión SMB utiliza Transport Control Protocol (TCP)/139 como protocolo de transporte. Como se observa en la figura 2.16, sobre TCP se ubica la capa NetBIOS y luego SMB. La autenticación NTLM está directamente incorporada en los paquetes SMB, donde se utiliza un mecanismo de negociación para seleccionar la semántica y parámetros específicos que el cliente y el servidor usarán durante la sesión.

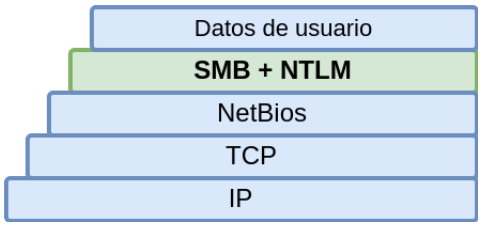

Figura 2.16: Capas del protocolo SMB

Los sistemas Windows 2000 y posteriores implementan algunas extensiones al protocolo SMB conocidas como CIFS. El protocolo de sesión SMB/CIFS utiliza TCP/445 como protocolo de transporte. En general, CIFS sobre TCP/445 no implementa la capa NetBIOS y utiliza SSPI para la autenticación y seguridad de la sesión, en lugar del protocolo NTLM incorporado en los mensajes SMB (figura 2.17). SSPI provee una capa de abstracción que permite a SMB/CIFS utilizar Kerberos u otros protocolos de autenticación en lugar de NTLM.

En la actualidad, la primera versión del protocolo SMB/CIFS es considerada obsoleta, en favor de versiones más modernas y robustas como SMB 2.0 y 3.0, incorporadas a partir de 2006 en Windows Vista, Windows Server 2008 y posteriores. Recién a partir de la versión 3 del protocolo, introducida con Windows 8 y Windows Server 2012, SMB soporta encriptación de los datos de usuario además de la autenticación.

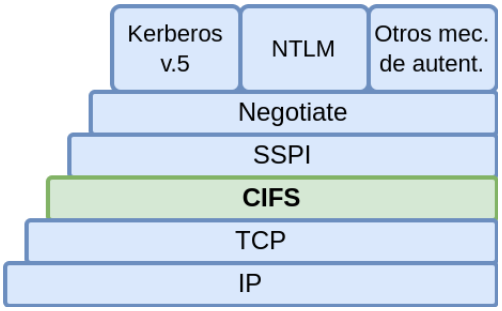

Figura 2.17: Capas del protocolo SMB/CIFS

2.7.8. Autenticación en HTTP

Hoy en día, HTTP es probablemente el protocolo más utilizado en Internet. Los usuarios utilizan este protocolo no solo para acceder a páginas webs con contenido estático a través de su navegador, sino también a verdaderas aplicaciones y servicios enriquecidos alojados en la nube (Rich Internet Application (RIA)).

En sus comienzos HTTP, como la mayoría de los protocolos de Internet tradicionales, no requería de autenticación para acceder a un recurso en el servidor. Gran parte del contenido en Internet es público, lo que permite un acceso anónimo a dichos recursos a través de una simple solicitud *GET* de HTTP. Sin embargo, hoy la autenticación se vuelve una necesidad ya que a través de las aplicaciones web se accede a información restringida por las organizaciones o de uso personal por parte de los usuarios.

Aunque la autenticación puede ser anónima desde la perspectiva del usuario, el servidor web utiliza una cuenta de usuario restringida del sistema operativo para acceder a los recursos solicitados por el cliente. Además, ciertas aplicaciones web suelen extraer información de la sesión HTTP con el fin de identificar al usuario detrás del navegador; como la dirección IP, el agente de usuario (tipo de navegador) y las denominadas *cookies* de sesión.

Otro aspecto importante de HTTP en términos de autenticación es el concepto de sesión. HTTP fue diseñado para trabajar sin estado (*stateless*). Por ello, en la versión 1.0 del protocolo (HTTP/1.0) definido en [40], la sesión como tal contenía una única solicitud HTTP para obtener un único recurso u objeto. Tanto para mejorar la eficiencia, como para evitar múltiples autenticaciones de usuario, se definió en [41] la versión 1.1 del protocolo (HTTP/1.1), el cual permite obtener varios objetos a través de una única conexión o sesión HTTP. Para mantener la sesión activa durante largos períodos de tiempo, HTTP hace uso de los mensajes *keep-alive* permitiendo al cliente intercambiar datos con el servidor sin requerir nuevas autenticaciones.

En [42] se define un marco general para el control de acceso y autenticación en HTTP a través de un conjunto extensible de esquemas de autenticación del tipo desafío-respuesta.

El flujo de mensajes típico entre un cliente y un servidor HTTP para acceder a un recurso protegido se resuelve de la siguiente manera:

- El cliente inicia una solicitud de acceso anónima al recurso.
- El servidor devuelve al cliente un código de respuesta 401 (*Unauthorized*) y el esquema de autenticación a utilizar en el encabezado de tipo **WWW-Authenticate**.
- El cliente solicita al usuario las credenciales de acceso que luego incluye en un encabezado de tipo **Authorization** según el esquema de autenticación indicado por el servidor.
- El servidor verifica las credenciales proporcionadas por el cliente y autoriza el acceso a través de un código de respuesta 200 (*OK*) y el contenido del recurso solicitado. En caso de que las credenciales no sean válidas el servidor responde con el código de respuesta 403 (*Forbidden*)

El esquema de autenticación HTTP más común es la autenticación *Basic*. En este esquema las credenciales se envían en texto claro codificadas en base64 por lo que se recomienda utilizar Hyper Text Transfer Protocol Secure (HTTPS) para que sea seguro.

Además de Basic, existen otros esquemas de autenticación HTTP que se diferencian por su robustez y compatibilidad con navegadores y servidores web:

- Bearer: bearer tokens para acceso a recursos protegidos mediante OAuth 2.0([43]). Se puede utilizar también con JSON Web Token (JWT) [44].
- Digest: Hace uso de funciones hash MD5 y SHA para enviar el password encriptado junto con otra información de seguridad, en lugar de usar base64 como en Basic ([45] y [46]).
- Negotiate: Utilizado por SPNEGO para la autenticación mediante GSS-API y Kerberos como se observa en la figura 2.11 ([47])
- HOBA: HTTP Origin-Bound Authentication basado en firma digital ([48])

A pesar de los esquemas definidos con anterioridad, una de las formas más habituales de autenticación en HTTP es hacer uso de los formularios de login con Hyper Text Markup Language (HTML), donde se envían las credenciales del usuario a través de una solicitud HTTPS *POST* al servidor. Luego, mediante

cualquiera sea el mecanismo de autenticación definido por aplicación, se validan las credenciales y se devuelve al cliente una cookie de sesión o un token de acceso para futuras solicitudes, tal como se observa en la figura 2.18

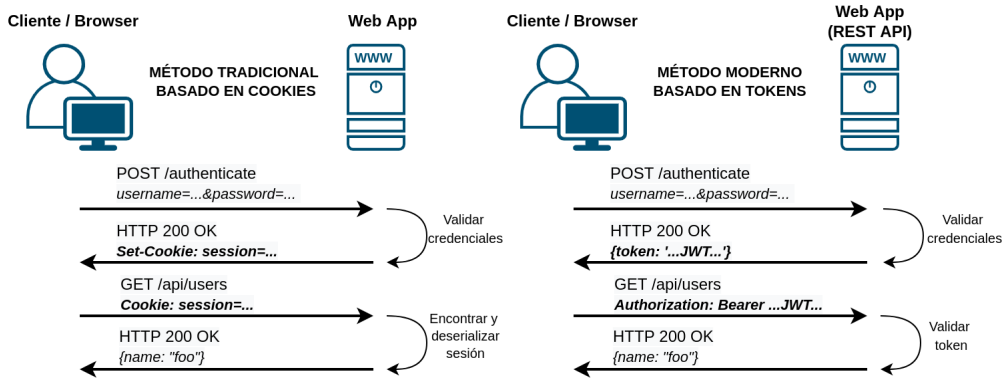

Figura 2.18: Dos variantes de autenticación HTTP basada en formulario HTML

2.7.9. Autenticación SAML, OAuth y OpenID Connect

Security Assertion Markup Language (SAML), desarrollado por Organization for the Advancement of Structured Information Standards (OASIS), es un estándar abierto basado en eXtensible Markup Language (XML), para el intercambio de información de autenticación y autorización del usuario (**aserciones**) entre un IdP (*authority/asserting party*) y un SP (*relying party*) en entornos multi-dominio [49]. La versión 2.0 de SAML permite a las organizaciones establecer relaciones de **confianza** entre ellas (Business to Business (B2B)) y entre clientes (Business to Customers (B2C)) para lograr una autenticación del tipo SSO y federada.

Por lo general, el modelo XML utilizado para la comunicación entre las partes se conoce como Simple Object Access Protocol (SOAP). Un mensaje SOAP es un documento XML ordinario con una estructura definida en las especificaciones del protocolo. Estos mensajes se transportan sobre HTTP/HTTPS dando lugar a una estructura de capas como se observa en la figura 2.19. Dependiendo de la configuración, también es posible utilizar SAML directamente sobre HTTP/HTTPS, prescindiendo de SOAP como protocolo de transporte.

En SAML, el proceso de autenticación y autorización entre el sujeto (usuario) y los proveedores de identidad y servicio se define en términos de los siguientes elementos:

Aserciones Información de autenticación, atributos y derechos (permisos) del sujeto.

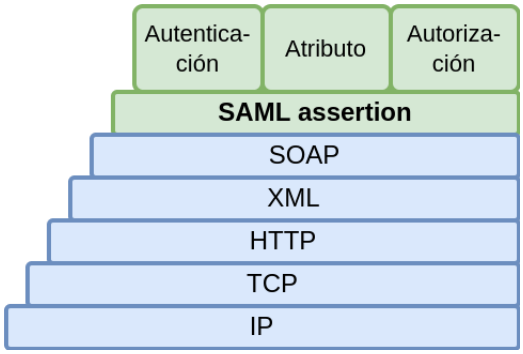

Figura 2.19: Stack de protocolos para SAML

Protocolos Mensajes de solicitud y respuesta para obtener las aserciones y gestionar la identidad del sujeto

Asociaciones Define los protocolos de transporte a utilizar (HTTP/SOAP) para los mensajes del protocolo SAML.

Perfiles Define los requerimientos de las aserciones, protocolos y asociaciones para soportar un determinado caso de uso.

Los principales usos de SAML son para autenticación web SSO e identidad federada como se observa en la figura 2.20.

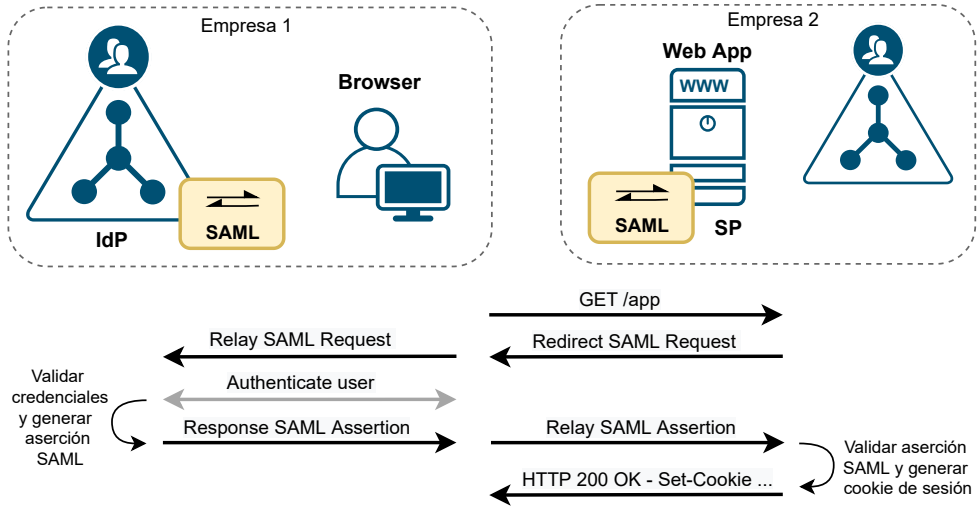

Figura 2.20: Flujo típico de mensajes SAML.

Por ejemplo, el perfil *Web Browser SSO* especifica cómo utilizar el protocolo de autenticación SAML de solicitud/respuesta junto con las diferentes combinaciones de asociaciones HTTP Redirect, POST, Artifact y SOAP. SAML habilita la autenticación web SSO a través de la comunicación de una aserción de autenticación entre el primer sitio web (IdP) y el segundo sitio (SP), si este último confía en el origen de la aserción, permitiendo el login del usuario como si hubiese autenticado directamente.

Para la identidad federada se utiliza el perfil de *Atributo* donde se definen una serie de perfiles con reglas para la interpretación de los atributos de una aserción SAML. Por ejemplo, el perfil X.500/LDAP describe como transportar atributos de este tipo en una aserción SAML.

Similar al escenario de SSO web, el modelo de autorización basado en atributo tiene a un sitio web comunicando información de identidad sobre un sujeto a otro sitio web en apoyo de alguna transacción específica. Sin embargo, la diferencia radica en que la información de identidad puede ser alguna característica del sujeto (como el rol en un escenario B2B) en lugar de, o además de, información sobre cuándo y cómo la persona fue autenticada. El modelo de autorización basado en atributos es importante cuando la identidad particular del individuo no es importante, no debería ser compartida por razones de privacidad, o es insuficiente.

En ambos tipos de escenario, SSO web e identidad federada, los proveedores de servicio no necesitan tener una lista completa de todos los usuarios existentes en el proveedor de identidad. Según cada caso, las cuentas de usuario externas que son autenticadas por el IdP, son mapeadas a las cuentas de usuario local del SP para permitir el modelo de control de acceso impuesto localmente. Este proceso de mapeo se conoce como vinculación de cuenta (*account linking*) y puede ser del tipo uno a uno o muchos a uno [8].

OpenID es otro protocolo de autenticación web similar en algunos aspectos a SAML pero orientado a usuarios finales en lugar de usuarios empresariales. Al ser un estándar de identificación digital descentralizado, el usuario puede elegir su proveedor OpenID al momento de la autenticación y así eliminar la necesidad de establecer relaciones de confianza explícitas entre el IdP y el SP.

Mediante OpenID el usuario se autentica con la cuenta seleccionada en los sitios web que soporten el protocolo sin la necesidad de crear una nueva cuenta de usuario. A diferencia de SAML, la información de autenticación se transporta sobre simples URL (o un eXtensible Resource Identifier (XRI) a partir de la versión 2.0) en lugar de complejos mensajes XML.

Actualmente, OpenID y SAML están cayendo en deshuso a favor de **OAuth** junto a **OpenID Connect (OIDC)**, que se encuentran orientados a una arquitectura REST como la mayoría de las aplicaciones web modernas, que realizan llamadas HTTP a APIs y servicios web via Asynchronous JavaScript And XML (AJAX).

OAuth es un estándar abierto de autorización web que otorga a las aplicaciones, dispositivos y APIs, un acceso seguro y restringido de los recursos HTTP de un usuario, sin la necesidad de que accedan a sus credenciales. Actualmente, la versión 2.0 del protocolo (OAuth2) se encuentra especificada en [50] y [43], donde se definen los mecanismos para obtener y utilizar los *tokens de acceso* que permiten a un tercero acceder a los mencionados recursos del usuario.

Por otro lado, OIDC es una simple capa de autenticación por encima de OAuth2. Permite a las aplicaciones verificar la identidad de los usuarios basado en la autenticación que realiza el IdP o también llamado *Servidor de Autorización*, por el rol principal que realiza según las especificaciones del protocolo OAuth.

En la figura 2.21 se aprecia el flujo de mensajes típico de OAuth2, donde una aplicación web cliente (por ejemplo Linkedin) accede a los recursos de un usuario alojados en otra aplicación (por ejemplo Gmail).

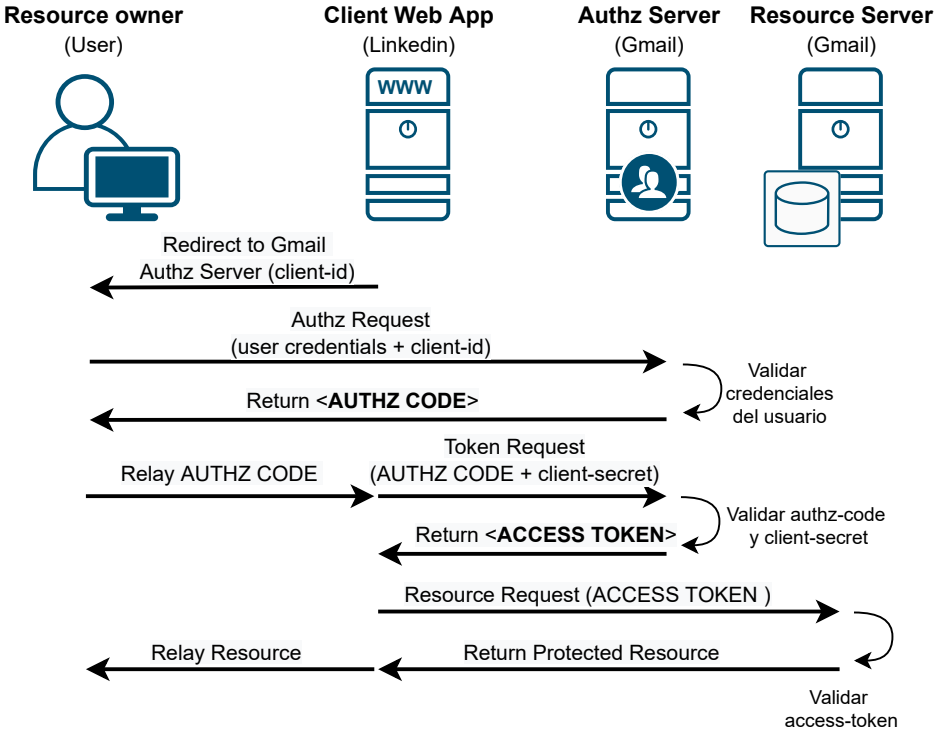

Figura 2.21: Flujo típico de mensajes OAuth2.

Para su funcionamiento el servidor de autorización de la aplicación que aloja los recursos del usuario, posee dos *endpoints*:

authorization endpoint Maneja la parte interactiva con el usuario con el fin de

validar la identidad del mismo y devolver un código de autorización para la aplicación cliente (mensajes 2 y 3 de la figura 2.21).

token endpoint Permite a la aplicación cliente, mediante el código de autorización, solicitar al servidor OAuth2 el token de acceso que luego se usará para acceder a los recursos HTTP permitidos por el usuario (mensajes 5 y 6 de la figura 2.21).

En este flujo de mensajes es importante destacar que la aplicación cliente se debe registrar ante el servidor de autorización como tal. Como parte de este proceso de registro el cliente obtiene un *client-id* y un *client-secret* que son utilizados durante la ejecución de los endpoints mencionados anteriormente [51].

Oauth2 no fue pensado como mecanismo de autenticación o login del usuario, sino como mecanismo para delegar autorización de una aplicación cliente a los recursos del usuario en una aplicación servidora. Sin embargo, mediante un uso abusivo e inseguro de OAuth, muchos desarrollos suponen al usuario autenticado si la negociación OAuth explicada en la figura 2.21 es exitosa.

Para resolver este problema es que se implementa OIDC sobre OAuth. OIDC define nuevos endpoints en el servidor de autorización que permiten obtener, además del token de acceso, un *token ID* con información de autenticación que las aplicaciones cliente pueden utilizar para verificar la identidad del usuario.

Tanto OpenID como OAuth y OIDC, son independientes del método de autenticación llevado a cabo por el IdP. En efecto, aunque son considerados protocolos de autenticación web SSO, ninguno de ellos provee tal característica por sí solos. La experiencia SSO viene dada por el almacenamiento caché en el navegador web de las cookies de sesión [52].

2.8. Autenticación a la Infraestructura

Generalmente, antes de poder utilizar las aplicaciones y servicios de una organización, existe un proceso de autenticación previo, que identifica al usuario y/o a la estación de trabajo, y permite el acceso a la infraestructura de red.

Cuando se habla de infraestructura de red se refiere a los dispositivos de capa 2 y 3, como switches y routers, que proporcionan la conectividad necesaria para las operaciones de la organización.

El hecho de que el usuario tenga acceso físico al edificio donde reside la Local Area Network (LAN) corporativa, y que la identidad de la persona haya sido verificada al ingreso, suele ser un medio de autenticación suficiente para proporcionar al usuario también acceso a la infraestructura de red.

En otros casos, es posible que además se requiera que la estación de trabajo del usuario se autentique en el switch donde está conectada, antes de que se le

permita transferir y recibir tráfico de la red. Normalmente, esto se realiza mediante el protocolo IEEE 802.1x.

Sin embargo, cuando el usuario se encuentra fuera del edificio donde reside la red corporativa, la autenticación a la infraestructura es un requerimiento de seguridad necesario. En estos casos el acceso remoto se suele dar a través de túneles VPN de capa 2 y 3 con la red corporativa, utilizando conexiones privadas, semi-privadas o públicas como Internet.

Otro ejemplo de acceso a la infraestructura son las conexiones inalámbricas o *WiFi*. Debido al hecho de que no se puede asegurar que el usuario se encuentra físicamente en el edificio de la organización, la autenticación en estos casos también es requerida.

Además de la autenticación del usuario para acceder a la infraestructura, los dispositivos de red también requieren de una autenticación para acceder a la gestión de los mismos. Es común que la gestión de estos dispositivos se realice a través de un acceso SSH al Comnand Line Interface (CLI) del equipo, o un acceso web para la administración a través de una interfaz gráfica. La autenticación quedará definida dependiendo del método de acceso elegido y la configuración propia de cada equipo, según lo visto en las secciones 2.7.6 y 2.7.8 referidas a la autenticación de los protocolos SSH y HTTP respectivamente.

A continuación se describen los métodos de autenticación a la infraestructura de red según los diferentes contextos planteados en los párrafos anteriores.

2.8.1. Autenticación para Acceso Remoto

El acceso remoto a la infraestructura de red es una necesidad en la mayoría de las organizaciones modernas. Empleados itinerantes (*road warriors*), trabajo desde la casa (*home office*), administradores de red, desarrolladores de software, etc; son todos candidatos a un acceso remoto.

En sus comienzos, el acceso remoto se realizaba a través de líneas telefónicas de dial-up o Integrated Service Digital Network (ISDN) mediante protocolos de enlace como Serial Line Internet Protocol (SLIP) y Point to Point Protocol (PPP). SLIP era un protocolo que únicamente proveía de transporte para el protocolo Internet Protocol (IP) y carecía de autenticación. Sin embargo, PPP es un protocolo aún vigente, con un diseño modular y soporte para diferentes protocolos de red, además de un mecanismo de negociación que permite implementar la autenticación del usuario a nivel de enlace de datos.

El protocolo PPP consiste de los siguientes componentes:

Protocolo de encapsulación PPP Utilizado para transportar los protocolos de control del enlace y de red, además de los protocolos de red configurados para

el transporte de los datos de usuario. Admite la encapsulación de datos sobre diferentes enlaces físicos seriales síncronos y asíncronos.

Link Control Protocol (LCP) Establece, configura, mantiene y termina las conexiones PPP. Dentro de las opciones de configuración se encuentra la autenticación de los extremos de la conexión.

Network Control Protocol (NCP) Familia de protocolos para el establecimiento y configuración de diferentes protocolos de red (IP, IPX, AppleTalk, NetBEUI, etc), una vez establecido y configurado el enlace por LCP.

Protocolos de Autenticación Implementación de los diferentes mecanismos de autenticación durante la fase de negociación del enlace.

Algunos de los mecanismos de autenticación soportados por PPP son los siguientes:

Password Authentication Protocol (PAP) Es considerado el método de autenticación más sencillo. Prácticamente todos los fabricantes lo soportan. El nombre de usuario y contraseña se transfieren en texto plano por lo que es necesario considerar algún mecanismo de protección adicional.

Challenge-Handshake Authentication Protocol (CHAP) Se trata de un mecanismo de autenticación del tipo desafío-respuesta. Se lo puede utilizar para autenticar al cliente, al servidor o en ambas direcciones. La autenticación se realiza a través de un saludo de tres vías, donde el servidor (autenticador) envía un desafío aleatorio asociado a un identificador único, al cual el cliente responde aplicando una función hash sobre dicho desafío concatenado al identificador y a la contraseña del usuario. Luego el servidor compara la respuesta del cliente contra su propio cálculo y en caso de coincidencia se considera al usuario autenticado.

Microsoft Software CHAP (MS-CHAP) CHAP requiere que el servidor de autenticación tenga acceso a la contraseña en texto claro del usuario. Es por ello que Microsoft implementó su propia versión de CHAP que utiliza los hashes de contraseña nativos NT y LM, como se describe en la sección 2.7.2, en lugar de la contraseña en texto claro. Existen dos variantes del protocolo: MS-CHAP v.1 y v.2.

Extensible Authentication Protocol (EAP) Es un protocolo de autenticación genérico que provee a las aplicaciones una interfaz para la implementación de mecanismos de autenticación más específicos. EAP puede ser utilizado sobre otros protocolos de enlace como IEEE 802.3 e IEEE 802.2, y no solo sobre PPP.

Siguiendo con el modelo de autenticación PPP, EAP soporta cuatro tipos de mensajes: Solicitud, Respuesta, Suceso y Falla. Además provee una capa de abstracción desde la perspectiva del autenticador o NAS, ya que este último no necesita entender todos los mecanismos de autenticación y puede actuar como un dispositivo *pass-through*, reenviando los mensajes EAP entre el cliente y un servidor backend de autenticación como Remote Authentication Dial-In User Service (RADIUS) o Terminal Access Controller Access Control System Plus (TACACS+) [53].

EAP soporta diferentes mecanismos de autenticación entre los cuales se puede citar los siguientes:

EAP-MD5 PPP-CHAP encapsulado en EAP

EAP-TLS Es un método de encapsular SSL/TLS dentro de los mensajes EAP. La autenticación del usuario se logra mediante mapeo de certificados X.509.

EAP-Tunneled TLS (TTLS) Es una extensión de EAP-TLS que permite utilizar otros mecanismos de autenticación del usuario, como PAP, CHAP o MS-CHAP, en lugar del mapeo de certificados. EAP-TTLS es similar a EAP-TLS ya que establece una conexión TLS entre el cliente y el servidor, y autentica al servidor por medio de su certificado X.509. Sin embargo, para la autenticación del cliente se pueden utilizar otros mecanismos más sencillos de implementar que el mapeo de certificados, una vez que el túnel TLS está establecido.

Protected EAP (PEAP) Es un mecanismo de autenticación EAP propuesto por Microsoft, Cisco y RSA Security, donde en este caso, a diferencia de lo que sucede con EAP-TLS y EAP-TTLS, nuevos mensajes EAP son encapsulados dentro de un túnel TLS. La autenticación PEAP consiste de dos fases: en la primera se establece el túnel TLS y se autentica al servidor mediante su certificado; en la segunda fase se utiliza un segundo mecanismo de autenticación EAP para autenticar al usuario y, dependiendo del mecanismo seleccionado, probablemente se autentica al servidor nuevamente.

Lightweight EAP (LEAP) Es un mecanismo de autenticación propietario del tipo desafío-respuesta desarrollado por Cisco que implementa MS-CHAP v.2 sobre EAP. Como resultado, el cliente y el servidor de autenticación deben tener acceso al hash NT de la contraseña del usuario para completar con éxito la autenticación. Esto es conveniente cuando se utiliza Windows NT y sistemas operativos compatibles como base de datos de usuario para el backend. Dado que MS-CHAP es un protocolo que proporciona autenticación mutua, también lo es LEAP.

EAP-Flexible Authentication via Secure Tunneling (EAP-FAST) EAP-TLS y PEAP son mecanismos de autenticación seguros admitidos por varios dispositivos, pero las organizaciones a menudo encuentran difícil implementarlos debido a los requisitos para los certificados del servidor y el cliente. Cisco desarrolló EAP-FAST en un esfuerzo por proporcionar un método de autenticación flexible y seguro que pueda ajustarse a los entornos de los clientes, ya sea que el cliente tenga o no una implementación de PKI existente.

Un elemento clave del mecanismo de autenticación EAP-FAST es el archivo Protected Access Credential (PAC). El servidor de autenticación genera el PAC, el cual se puede distribuir (aprovisionar) de forma manual o automática. El PAC contiene metadatos y claves secretas, lo que lo hace equivalente a un certificado X.509 y una clave privada.

2.8.2. Control de acceso basado en puerto

Ethernet (IEEE 802.3) a demostrado ser el protocolo de acceso a la red ganador desde principios de la década de los 90, y a fines del siglo XX ser prácticamente el único protocolo de acceso para las redes locales.

Además de la inversión de algunos fabricantes para hacer de Ethernet más popular, una de las principales razones de su éxito fue su relativa simple y barata implementación. También para el usuario conectarse a la red era muy sencillo utilizando algún jack Ethernet disponible cerca de su escritorio, para luego obtener automáticamente la configuración de red IP a través del protocolo Dinamic Host Configuration Protocol (DHCP).

A principios del 2000 emerge el protocolo IEEE 802.11 (WiFi) como alternativa inalámbrica de Ethernet para las organizaciones. Ya para el año 2005 su uso se expande ampliamente, alcanzando también a pequeñas oficinas y hogares. WiFi hace que el acceso a la red sea aún mas sencillo, permitiendo a cualquier usuario conectarse a la red local, aún estando físicamente fuera del edificio donde la red reside, siempre y cuando se encuentre dentro del rango de cobertura de un Access Point (AP).

En la mayoría de los casos, la facilidad en el uso y acceso a la red contradice los principios de seguridad de la información. Mientras que entre los años 1970 y 2000 el esfuerzo estuvo en permitir un acceso sencillo a la red, a partir del 2000 se volvió una necesidad autenticar a los usuarios antes de permitirles una conexión a la LAN.

Es por eso que en 2001, la IEEE introduce el estándar 802.1x para el control de acceso basado en puertos Ethernet, que evoluciona al estándar 802.1x-2004 para incorporar las necesidades de autenticación de las redes WiFi. A medida que la autenticación mediante WEP se volvía obsoleta para redes WiFi, 802.1x fue

adoptado por el estándar 802.11i (WPA/WPA2) como mecanismo primario para la autenticación y gestión de claves.

El método de autenticación utilizado por 802.1x es EAP, que fue originalmente desarrollado para la autenticación en PPP. El estándar 802.1x define los siguientes términos para describir las partes que intervienen en el proceso de autenticación:

Suplicante El sistema cliente que está solicitando la autenticación para acceder a la infraestructura de red, como una computadora de escritorio o equipo portátil.

Autenticador El dispositivo de red o NAS que forma parte del proceso de autenticación y que luego de una autenticación exitosa provee al suplicante del acceso a la infraestructura de red.

Servidor de Autenticación La autoridad de red que centraliza el servicio de autenticación de usuarios para la organización. En el caso de 802.1x, el servidor de autenticación es típicamente un servidor RADIUS como se describe en la sección 2.8.5

Un puerto Ethernet controlado por 802.1x, puede ser del tipo *autorizado* o *no autorizado*, dependiendo de si puede o no enviar y recibir tráfico en la red. Durante el proceso de autenticación sólo se permite enviar y recibir mensajes EAP over LAN (EAPOL). EAPOL es el mecanismo por el cual se encapsulan los mensajes del protocolo EAP en 802.1x, como se observa en la figura 2.22.

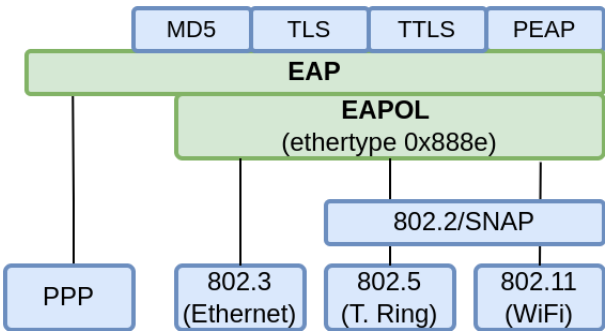

Figura 2.22: Capas del protocolo 802.1x

Los tipos de mensajes utilizados por el protocolo EAPOL son los siguientes:

- 0-EAPOL Packet: los mensajes EAP encapsulados en EAPOL
- 1-EAPOL Start: utilizado por el suplicante para iniciar el diálogo EAP.

- 2-EAPOL Logoff: utilizado por el suplicante para finalizar la sesión EAPOL.
- 3-EAPOL Key: Para el intercambio de claves entre el autenticador y el suplicante
- 4-EAPOL ASF Alert: permite el envío de alertas según el Alerting Standards Forum (ASF) (por lo general, traps SNMP) en puertos que se encuentran en estado *no autorizazdo*

Aunque la fase de autenticación 802.1x termina proporcionando claves de protección a las partes que se autentican, aún pueden decidir intercambiar nuevas claves, y los mensajes EAPOL Key se utilizan exactamente para este propósito. Un ejemplo típico es la protección de datos en 802.11i WPA-PSK, donde no se utiliza la autenticación 802.1x y sólo se realiza el intercambio de claves 802.1x mediante mensajes EAPOL Key.

La figura 2.23 muestra un diálogo EAPOL típico sobre un puerto Ethernet protegido con 802.1x y utilizando el serivicio RADIUS como servidor de autenticación. La cantidad de mensajes RADIUS-Access Challenge/EAP-Request y EAP-Response/RADIUS-Access Request, dependerá del tipo de autenticación EAP seleccionado.

2.8.3. Autenticación en redes WiFi

El estándar 802.11 define dos métodos principales de autenticación para el acceso a la infraestructura WiFi: (1) **autenticación abierta** y (2) **autenticación de clave compartida**. Antes de que un dispositivo se conecte a la red WiFi debe autenticarse y asociarse a un AP mediante alguno de éstos métodos.

La autenticación abierta es tan débil como no tener autenticación alguna. Pero al mismo tiempo, aún luego de una autenticación y asociación abierta exitosa, el usuario no puede utilizar la infraestructura WiFi si la encriptación Wired Equivalent Privacy (WEP) se encuentra habilitada y a su vez desconoce la clave para desencriptar los paquetes.

WEP fue el primer intento de la IEEE para proteger las comunicaciones inalámbricas en las redes 802.11. Utiliza una clave precompartida entre el cliente y el AP para encriptar todo el tráfico de red entre ambos. Las claves pueden ser de 40 o 104 bits de longitud, que se combinan con un vector de inicialización aleatorio de 24 bits para conformar una clave final de 64 o 128 bits respectivamente.

Dada la debilidad del algoritmo de cifrado RC4 utilizado en WEP, la autenticación abierta permite a un atacante autenticarse libremente para luego capturar, modificar y transmitir paquetes en la red. Cuando la autenticación abierta se combina con mecanismos de protección más fuertes, como los definidos por Wifi

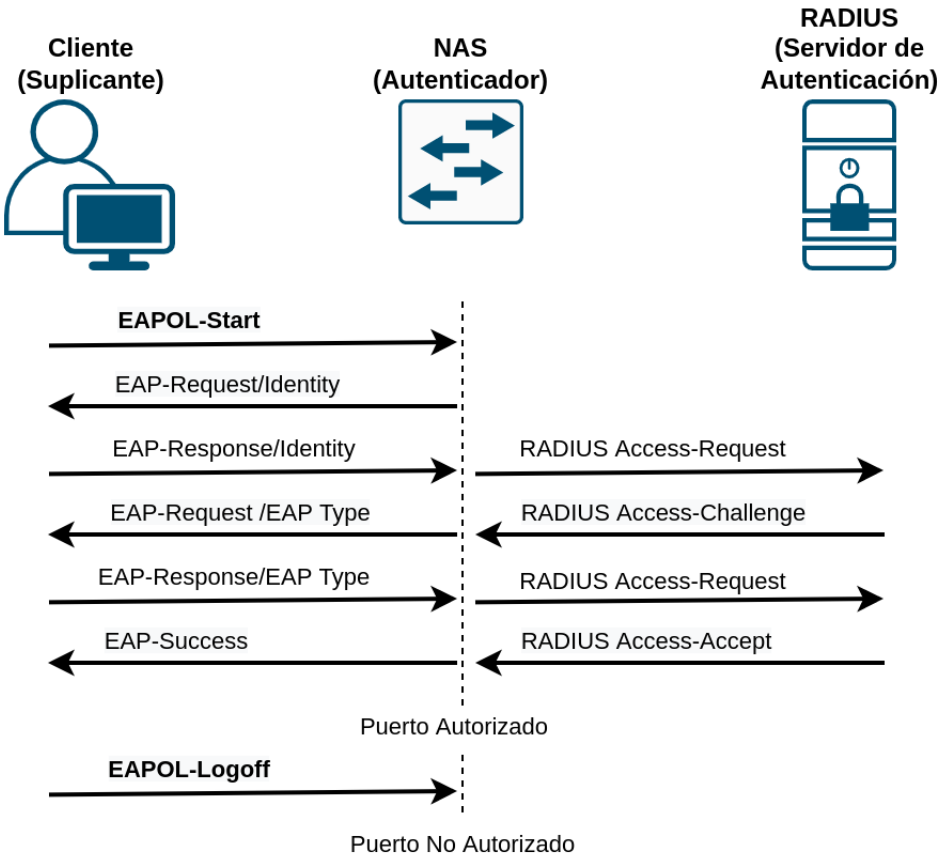

Figura 2.23: Flujo de mensajes 802.1x

Protected Access (WPA) y WPA2, los datos de usuario pueden estar mejor protegidos.

El segundo método de autenticación definido en el estándar 802.11 es el de clave compartida. Se basa en un esquema simple del tipo desafío-respuesta, donde el AP envía al cliente un desafío aleatorio de 128 bits, el cual debe ser devuelto por el cliente encriptado mediante WEP. Al igual que en la autenticación abierta, el uso de WEP hace que la autenticación de clave compartida sea un mecanismo inseguro para las comunicaciones inalámbricas.

Mientras la IEEE trabajaba en un nuevo estándar para solucionar los problemas de seguridad del estándar anterior, los fabricantes comenzaron a utilizar un pre-estándar conocido como WPA. Luego, en 2004 cuando la IEEE lanzó el estándar 802.11i se lo denominó WPA2.

Ambos pueden utilizar 802.1x para la autenticación de usuarios y gestión de claves, pero difieren en la manera de encriptar los datos. WPA introduce el me-

canismo Temporal Key Integrity Protocol (TKIP), mientras que WPA2 utiliza el algoritmo Counter Mode with CBC-MAC Protocol (CCMP) basado en Advanced Encryption Standard (AES).

Tanto TKIP como CCMP-AES, utilizan claves temporales para proteger los datos de usuario, Pairwise Transient Key (PTK), derivadas de una clave común que existe entre el cliente y el AP: Pairwise Master Key (PMK). El proceso de derivación de las claves PTK desde la clave maestra PMK utiliza los mensajes EAPOL-Key y se lo conoce como *saludo de cuatro vías* de WPA.

WPA y WPA2 ofrecen dos modos de operación en términos de autenticación: (1) **modo empresarial** (Enterprise) y (2) **modo clave precompartida** (PSK).

Para aplicaciones de nivel empresarial, el estándar 802.11i abandonó la idea de una autenticación de clave compartida y optó por usar la autenticación basada en 802.1x (EAPOL). En lugar de autenticar al usuario mientras se asocia a la red inalámbrica, los AP configurados con WPA-Enterprise permiten una autenticación abierta y luego solicitan al cliente ya asociado, la utilización del protocolo EAP para autenticar al usuario.

Luego de una autenticación EAP exitosa, el cliente y el AP comparten una clave de sesión común que pueden utilizar como PMK para el saludo de cuatro vías WPA posterior, como se observa en la figura 2.24

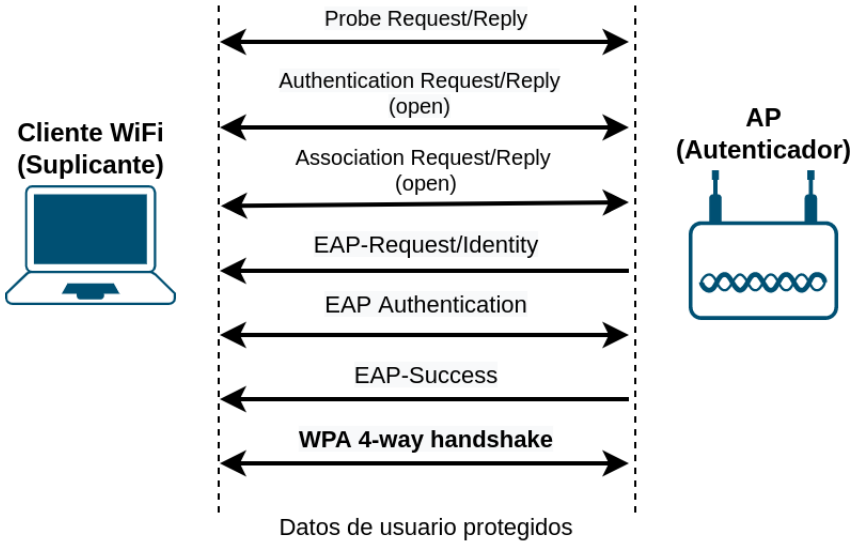

Figura 2.24: Autenticación WPA-Enterprise

Para abordar las necesidades de los usuarios hogareños y de pequeñas oficinas que no están dispuestos a invertir en soluciones de autenticación complejas, el estándar WPA/WPA2 retuvo el concepto WEP de autenticación de clave com-

partida. Este modo de operación, conocido como WPA-PSK, utiliza la clave precompartida como PMK para el saludo de cuatro vías WPA.

Es importante resaltar que este último modo de operación no es estrictamente un método de autenticación de usuario, ya que la clave precompartida es estática y única para todos los usuarios que acceden a la red WiFi.

2.8.4. Autenticación en IPsec/IKE

La arquitectura Internet Protocol security (IPsec), definida en [54], describe un conjunto de protocolos y algoritmos criptográficos para ser utilizados en el envío seguro de paquetes IP. Para su funcionamiento IPsec depende de los protocolos de encapsulación Authentication Header (AH) y Encapsulating Security Payload (ESP) definidos en [55] y [56] respectivamente.

IPsec por si sólo no provee de autenticación de pares y/o usuarios. Para ello se vale del protocolo Internet Key Exchange (IKE) basado en Internet Security Association and Key Management Protocol (ISAKMP) y descripto en [57] para su versión 1, y en [58] para su versión 2.

Ambas versiones de IKE proveen autenticación de las partes y además negocian de manera automática los parámetros de seguridad (Security Association (SA)) para ser utilizados en AH y ESP.

Para la autenticación de usuarios IKEv1 se vale de una implementación no estándar conocida como eXtended Authentication (XAUTH), e IKEv2 incorpora de manera nativa el protocolo EAP para dicho propósito, como se observa en la figura 2.25.

El protocolo IKE funciona sobre el puerto 500 de UDP para construir asociaciones seguras (SA) para IPsec.

Un SA define los parámetros de seguridad que aplican para un flujo unidireccional de datos entre las partes involucradas: protocolo de seguridad y modo de encapsulación (túnel o transporte), autenticación y claves secretas utilizadas para proteger la comunicación.

El SA se identifica de manera unívoca mediante una tupla compuesta por la dirección IP de destino, el protocolo de encapsulación (AH o ESP) y el Security Parameter Index (SPI). El SPI es un número de 32 bits que se genera para identificar al SA y es transportado en las cabeceras de AH y ESP.

En IKEv1 la autenticación de las partes se encuentra en la denominada fase 1 *main mode* de la negociación. Dicha autenticación puede ser mediante una clave precompartida entre las partes (PSK) o mediante certificados de clave pública (RSA). Para identificar a las partes se suele utilizar su dirección IP, su nombre de dominio o un nombre de grupo, en el caso de necesitar autenticar a los usuarios mediante XAUTH. Cuatro métodos de autenticación de usuarios se definen en XAUTH: genérico, RADIUS_CHAP, OTP y Secure ID.

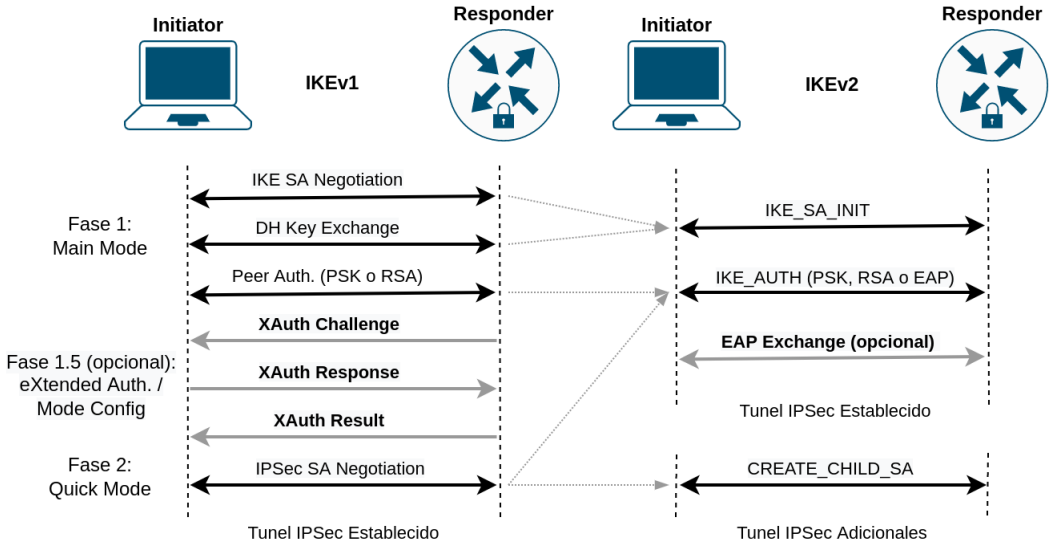

Figura 2.25: Autenticación en IKEv1 e IKEv2

En IKEv2, además de los métodos de autenticación PSK y RSA, se permite autenticar a los usuarios mediante EAP, aprovechando de esta manera los mecanismos de autenticación descriptos en la sección 2.8.1. La autenticación EAP se implementa en IKEv2 como un intercambio IKE_AUTH adicional.

2.8.5. RADIUS

RADIUS, definido en [59], es el estándar más ampliamente aceptado como servicio AAA para el acceso a la infraestructura. Prácticamente, todos los dispositivos de red empresariales lo soportan como servidor de backend para la autenticación centralizada de usuarios.

El funcionamiento consiste en solicitudes de acceso enviadas por los dispositivos de red o NAS al servidor RADIUS, para la autenticación y autorización de los usuarios que intentan acceder a la infraestructura. Si se lo requiere, el servicio de trazabilidad o *accounting* se realiza en una comunicación separada entre el NAS y el servidor RADIUS.

RADIUS es considerado un protocolo sin estados, por lo tanto, no existe el concepto de sesión entre el NAS y el servidor. El protocolo User Datagram Protocol (UDP) se ajusta bien en este caso como protocolo de transporte para los mensajes RADIUS. En la versión actual del protocolo RADIUS se utiliza el puerto UDP 1812 para la autenticación y autorización, y el puerto UDP 1813 para la trazabilidad.

Los mensajes RADIUS se componen de los siguientes campos:

Código Especifica el tipo de mensaje RADIUS:

- *Access-Request*
- *Access-Accept*
- *Access-Reject*
- *Access-Challenge*

Identificador Valor único utilizado para identificar el diálogo entre el cliente (NAS) y el servidor RADIUS.

Longitud Tamaño en bytes del mensaje RADIUS desde el campo Código al campo Atributos.

Autenticador 16 bytes que contienen el calculo hash de todos los campos del mensaje que envía el servidor al cliente, concatenado a una clave precompartida entre ambos. De esta manera, el cliente puede verificar la autenticidad de los mensajes enviados por el servidor. En los mensajes Access-Request enviados por el cliente, el Autenticador se utiliza como valor aleatorio en el proceso de encriptación de la contraseña enviada en el campo Atributos.

Atributos Parámetros pasados entre el cliente y el servidor como parte del proceso de autenticación y autorización. El estándar RADIUS define 63 atributos base, pero los fabricantes suelen definir sus propios atributos no estándar para operar funciones específicas de sus productos.

Dentro de los atributos mas importantes se encuentran los siguientes:

- User-name: El nombre de usuario puede o no incluir el *realm* en la forma de *username@domainname* o *domainname\username*
- User-password: Utilizado en esquemas de autenticación con contraseñas en texto plano (por ejemplo PAP)
- Chap-Password: Utilizado para responder al desafío en una autenticación CHAP
- NAS-IP-Address: Utilizado por el servidor para identificar al NAS en un Access-Resquest
- NAS-Port: La interfaz física o lógica utilizada por el NAS en la comunicación con el servidor

- Service-Type: Puede ser *Login* para una autenticación local con el NAS o *Framed* para utilizar un protocolo de enlace como PPP.
- EAP-Message: Utilizado para encapsular mensajes EAP en RADIUS según lo especificado en [60]

Típicamente, un servidor RADIUS puede servir a uno o varios *realms* o dominios de autoridad. El hecho de que el nombre de usuario indique el realm al que pertenece, permite el uso de servidores proxy para RADIUS. El servidor proxy puede aceptar una solicitud de acceso de un cliente, analizar el realm del nombre de usuario y reenviar la solicitud al servidor responsable de dicho dominio.

Tradicionalmente, un dispositivo de borde o NAS dialogaba con el dispositivo cliente del usuario utilizando un protocolo de capa de enlace como PPP (ver sección 2.8.1). Con el uso de RADIUS como servidor de autenticación, el dispositivo NAS debía transformar los mensajes de autenticación basados en PPP, como PAP y CHAP, en mensajes RADIUS utilizando los atributos apropiados según el mecanismo seleccionado.

A diferencia de los protocolos de autenticación tradicionales, los mensajes EAP pueden ser reenviados de manera transparente entre el dispositivo cliente y el servidor RADIUS sin necesidad de que sean procesados por el NAS, como se observa en la figura 2.23.

freeRADIUS es un servidor RADIUS de código abierto ampliamente utilizado en las infraestructuras de red de las organizaciones actuales. El nombre del demonio que ejecuta el servidor se conoce como *radiusd*. Corriendo el demonio en modo depuración, *radiusd -X*, se puede observar el diálogo entre el servidor y el cliente de prueba *radtest*, para una autenticación básica de contraseña en texto plano con PAP:

```
$ radtest testing password localhost 0 testing123

Sending Access-Request of id 226 to 127.0.0.1:1812
      User-Name = 'testing'
      User-Password = 'password'
      NAS-IP-Address = radius.domain.com
      NAS-Port = 0

rad_recv : Access-Accept packet from host 127.0.0.1:1812,id=226, length=56
      Framed-IP-Address = 80.84.161.1
      Framed-Protocol = PPP
      Service-Type = Framed-User
      Framed-Compression = Van-Jacobson-TCP-IP
      Framed-IP- Netmask = 255.255.255.255
```

3. Gestión de Identidad y Acceso

En este capítulo se analizan las características, desafíos y beneficios que prestan las diferentes alternativas de gestión de identidad y acceso (IAM), y se propone una solución que se ajuste al organismo del caso de estudio: (TEPM). El capítulo comienza con la descripción de un sistema de IAM, las herramientas disponibles para su implementación y una descripción del entorno de trabajo del TEPM. Luego, se estudia la manera de integrar un IAM al mencionado organismo y se presenta un diseño que se ajuste a las necesidades. Dicho diseño se utiliza como infraestructura tecnológica para la implementación del gestor de autenticación de usuarios del capítulo 4.

3.1. Características de un IAM

En la mayoría de las organizaciones los empleados tienen más permisos y privilegios de los necesarios para realizar sus actividades y además no se conoce con precisión quienes tienen acceso y a que recursos de Information Technology (IT). El acceso a los servicios en la nube y la movilidad han agravado este problema.

La acumulación de privilegios a menudo ocurre cuando los empleados cambian de tareas o sector dentro de la organización y se les asigna nuevos privilegios, manteniendo temporalmente los privilegios anteriores, que luego no son removidos. Revisiones periódicas de los privilegios asignados puede resolver el problema, pero cuando la cantidad de usuarios a verificar es importante, se necesita un sistema que facilite la visualización y gestión.

El mayor porcentaje de ataques a la seguridad de la información provienen de amenazas internas, principalmente de usuarios disconformes o mal intencionados que hacen uso de estos privilegios excesivos. La recomendación en estos casos es implementar en la organización una solución de IAM que permita asegurar que los permisos y privilegios son otorgados de acuerdo a una política de acceso, y que los individuos y servicios se encuentran debidamente autenticados, autorizados y auditados.

Esta solución debe permitir además que el proceso de aprovisionamiento de usuarios y cuentas de usuario sea sencillo y seguro. Por lo tanto, un sistema de

IAM debe resolver dos cuestiones: la **gestión de identidad** y la **gestión de acceso**.

La gestión de identidad se refiere a las necesidades de una organización de administrar (crear, modificar, monitorear, eliminar) las cuentas de usuario, perfiles de usuario y políticas de una infraestructura de IT heterogénea a través de una combinación de roles de usuario y reglas de negocio [61].

La identidad, o identidad digital para ser más precisos, consiste en un identificador único y los respectivos atributos de una persona, grupo, dispositivo o servicio. Por lo que, el alcance de un IAM suele ser mucho más amplio que sólo la gestión de cuentas de usuario, y teóricamente puede incluir cualquier activo identificable de una organización.

Por otro lado, la gestión de acceso se refiere a la aplicación en tiempo real de las políticas de control de acceso, para cada usuario de los recursos de IT, a través de los procesos de autenticación y autorización [62].

En el capítulo 2 se describen los mecanismos de autenticación utilizados con mayor frecuencia en una organización. En relación a los mecanismos de autorización, se pueden citar los siguientes modelos de control de acceso: Mandatory Access Control (MAC), Discretionary Access Control (DAC) y Role Based Access Control (RBAC).

RBAC se destaca como uno de los mecanismos de autorización más utilizados debido a la flexibilidad y escalabilidad que ofrece a la hora de asignar los diferentes permisos de acceso. La filosofía de RBAC es dejar que la semántica del negocio gobierne el control de acceso en lugar de aspectos técnicos. Los roles se configuran para permitir a una persona, grupo, dispositivo o servicio realizar tareas o funciones de acuerdo al puesto o rol que ocupa dentro de la organización.

Para llevar adelante la gestión de identidad y accesos, un sistema de IAM se compone principalmente de los siguientes servicios:

- Servicio de Autenticación: Responsable de identificar y validar la identidad del usuario. Proporciona además características importantes como el inicio de sesión único (SSO), la autenticación multifactor y la administración de sesiones y tokens de acceso.

- Servicio de Autorización: Responsable de verificar si el usuario posee los permisos necesarios para acceder al recurso solicitado. Para ello debe poder corroborar los roles, reglas, atributos, y privilegios que posee dicho usuario.

- Servicio de Directorio: Almacén o repositorio de toda la información referida al sistema de gestión de identidad: usuarios, grupos, roles, privilegios, passwords, etc. Muchas veces se suelen utilizar metadirectorios que permiten consolidar los datos de diferentes fuentes de información y bases de datos.

- Servicio de Administración de Usuarios: Permite el aprovisionamiento de usuarios (creación, modificación y eliminación de cuentas de usuario y privilegios). Una funcionalidad importante de este servicio es la administración de contraseñas que permite sincronizar contraseñas, establecer políticas de contraseña y facilitar a los usuarios la recuperación o reinicio de las mismas a través de un sistema de autoservicio.

En la sección 2.2 se mencionó que el control de acceso se basa en los procesos de autenticación, autorización y trazabilidad, referidos como AAA. Si a ésto se agrega una cuarta 'A' (*Administration*), administración de usuarios, se tienen los cuatro procesos que deben estar presentes en una solución de IAM. En la figura 3.1 se puede ver como se relacionan estos cuatro factores con los servicios presentes en la gestión de identidad y acceso.

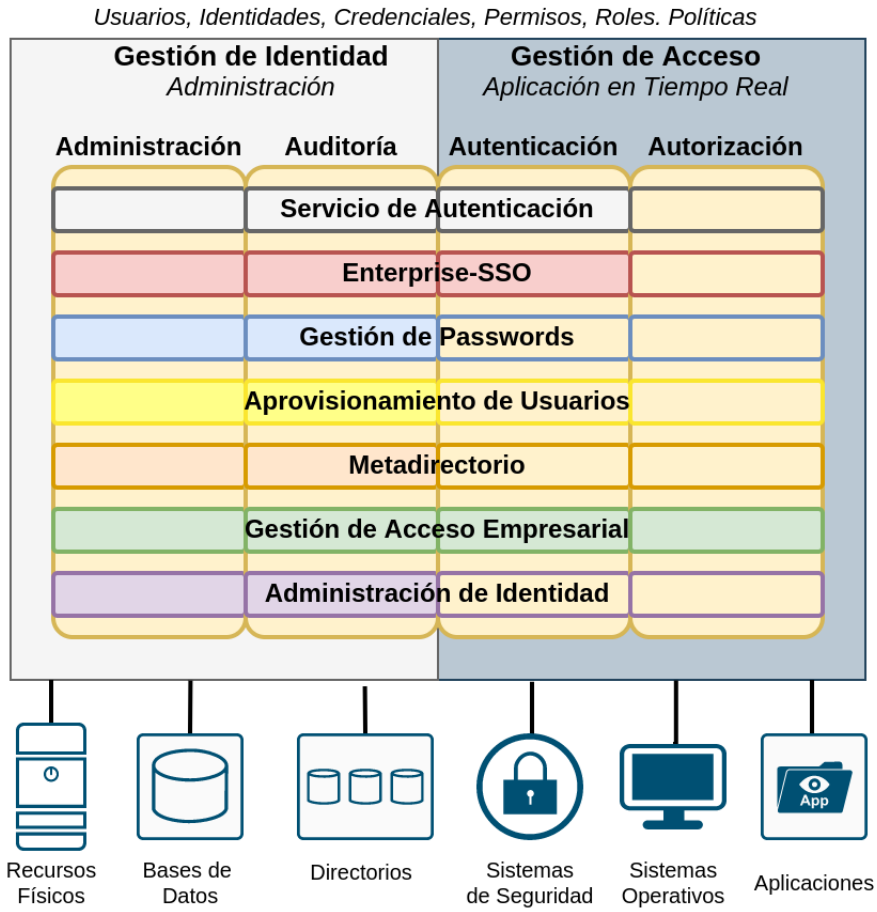

Figura 3.1: Visión general de un sistema de IAM

En resumen, el concepto de IAM combina los procesos de negocio con las políticas y tecnologías necesarias para administrar las identidades digitales y especificar como esas identidades son utilizadas para acceder a los recursos de IT.

La implementación de un sistema IAM implica un gran impacto en la organización, y a su vez significa grandes desafíos para lograrlo [63]. El principal desafío que se presenta es encontrar un balance entre dos requisitos opuestos en la gestión de la seguridad: ser lo suficientemente flexibles para lograr que los usuarios tengan el acceso que necesitan a los recursos que les corresponden (*Security of Enablement*); y ser lo suficientemente robustos para proteger dichos recursos adecuadamente (*Security of Protection*).

Otro de los desafíos que se presentan es la compatibilidad con los diferentes tipos de usuarios. Por ejemplo, en el caso del TEPM existen usuarios internos (empleados del TEPM) y externos (apoderados, jueces de paz, delegados, autoridades de mesa, etc.) cuyos requisitos de acceso son bien diferentes.

Además de lo mencionado, las organizaciones deben resolver como administrar el ciclo de vida de las identidades digitales (aprovisionamiento, mantenimiento y desaprovisionamiento), los diferentes tipos de credenciales para los diferentes tipos de sistemas que se utilizan y, por sobre todo, cumplir con las regulaciones legales que se le imponen.

Una vez implementado un sistema de IAM los beneficios están a la vista:

- Mayor control sobre las identidades y permisos de acceso a los recursos digitales, logrando una reducción de los riesgos internos y externos relacionados a una fuga de datos.
- Asegurar que los permisos y privilegios de los usuarios son otorgados de acuerdo una política de acceso y que se aplica un servicio de AAA apropiado en toda la infraestructura tecnológica de la organización.
- Aumento en la eficiencia del personal de IT y reducción de los costos relacionados a la gestión de las cuentas de usuario.
- Facilidad en la implementación o cambios en las políticas de seguridad de la organización, logrando una aplicación uniforme a todos los dispositivos y plataformas tecnológicas.
- Incremento en la usabilidad de los sistemas y satisfacción de los usuarios debido a la implementación de sistemas SSO, como se describe en la sección 2.4.

3.2. Gestores de Identidad y Acceso

No existe un único producto que realice todo lo que necesita un IAM y que además se ajuste a todas las plataformas y aplicaciones. Una implementación multi-producto es la única forma de cumplir con los requisitos de AAA de una organización moderna.

Los fabricantes y desarrolladores de soluciones IAM persiguen este enfoque multi-producto, distribuyendo una suite de herramientas que los administradores de IT deben luego ajustar, configurar y personalizar de acuerdo a las necesidades concretas de cada organización.

Los sistemas SSO son una parte fundamental de los IAM, y como se describe en la sección 2.4, no existe una solución de ”talla única”. Uno de los objetivos del presente trabajo es encontrar la solución que mejor se ajuste a las necesidades del TEPM y que sirva como base para otras organizaciones de similares características.

Antes de decidir por una solución en particular, es conveniente analizar las alternativas de IAM disponibles en el mercado para luego evaluar los pro y los contra de las diferentes soluciones. Resulta útil también analizar los trabajos realizados por otros autores relacionados a la gestión de identidad y acceso unificada.

Realmente no existen muchas opciones, y son dos las que se destacan según la plataforma elegida: Active Directory para Windows y FreeIPA para Linux/Unix. A continuación se describen ambas soluciones y luego se nombran otros trabajos relacionados a IAM.

3.2.1. Active Directory

Los controladores de dominio en Windows NT 4.0 y sistemas operativos anteriores utilizaban la base de datos SAM, basada en registros, para almacenar información sobre usuarios de dominio, grupos y cuentas de equipos. Esta base de datos sólo se podía modificar en el Primary Domain Controller (PDC). Los demás controladores del dominio, conocidos como Backup Domain Controller (BDC), mantenían una copia de respaldo de sólo lectura en caso de fallas en el PDC.

A partir de Windows 2000, se introdujo un nuevo servicio de directorio revolucionario conocido como Active Directory (AD), con las siguientes características principales:

- Almacenamiento de la información del directorio en archivos de base de datos, en lugar de registros del tipo SAM.
- Directorio tipo jerárquico compatible con LDAP.
- Soporte para los servicios de directorio heredados de Windows NT 4.0.

- Autenticación mediante protocolo Kerberos (más seguro que el antiguo NTLM, ver sección 2.7.2)
- Aplicación de políticas de grupo centralizada.

A diferencia del servicio de directorio en Windows NT 4.0, donde se representan las cuentas de usuarios, grupos y máquinas en forma plana, AD proporciona una estructura lógica jerárquica que consta de dominios y unidades organizativas (Organizational Unit (OU)).

Los dominios utilizan el servicio de DNS para definir el espacio de nombres del AD. Un conjunto de dominios bajo un espacio de nombres contiguo se conoce como árbol (*tree*). A su vez, a un conjunto de árboles, donde el espacio de nombres DNS no es contiguo, se lo denomina bosque (*forest*). En la figura 3.2 se observa la estructura jerárquica del AD descripta con anterioridad.

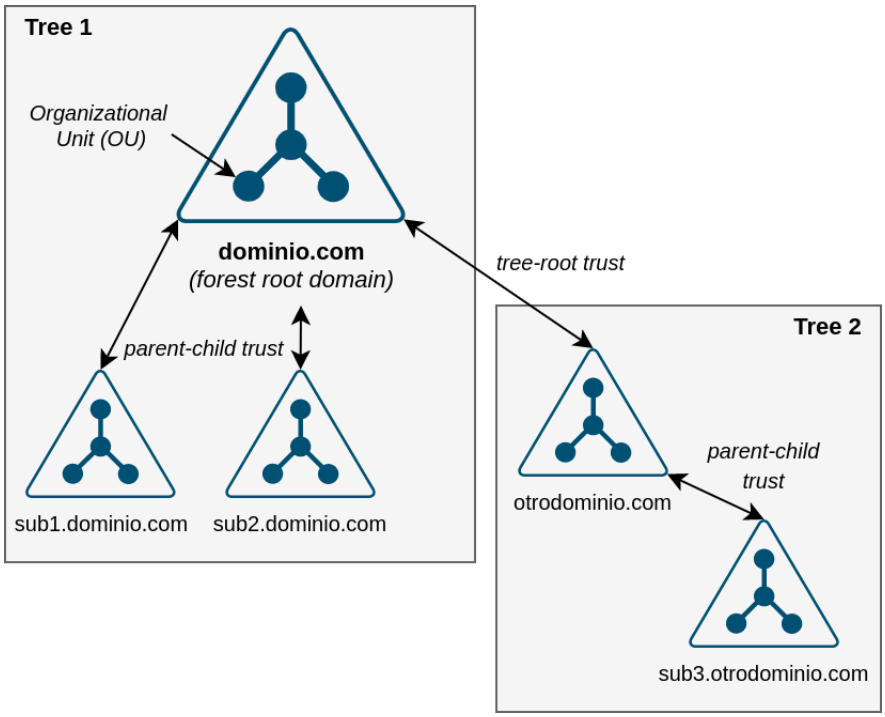

Figura 3.2: Estructura jerárquica de un AD

El primer dominio configurado en la organización, *Forest Root Domain*, proporciona el espacio de nombres para los contenedores de Configuración y Esquema del AD, que se replican en todos los controladores de dominio del bosque. Cada nuevo árbol que se agrega al bosque establece una relación de confianza *tree-root* con el dominio raíz.

Los controladores y miembros de un dominio son también miembros de un sitio o *site*. Los sitios en AD son redes locales altamente conectadas, mediante enlaces rápidos y confiables, definidos por una o más subredes IP.

La definición de sitios en una infraestructura AD redundante es importante porque permite a los clientes descubrir el controlador de dominio óptimo en su red local para lograr una autenticación exitosa. Además, la configuración de sitios y enlaces entre sitios, se utiliza para configurar de una manera más precisa los procesos de réplica entre los controladores de dominio.

Todos los controladores de dominio del bosque, además de compartir información de configuración de los sitios, membresías y esquema para los objetos del AD, comparten un Catálogo Global que facilita la búsqueda de cualquier objeto, junto a un conjunto seleccionado de atributos, en cualquiera de los dominios del bosque.

Las OUs son un medio para que los administradores del domino puedan agrupar recursos (usuarios, grupos, máquinas, etc) y ser gestionados como una sola unidad. Esto permite aplicar políticas de grupo (Group Policy Object (GPO)) de una manera más precisa y también poder delegar el control sobre los recursos del OU a otros administradores del dominio.

Los GPO son una herramienta poderosa que tienen los administradores para definir como se verá y comportará el sistema para un grupo determinado de usuarios. Cada GPO se compone de dos partes: una configuración de usuario y una configuración de computadora. Las políticas aplicadas a la computadora prevalecen sobre las políticas aplicadas al usuario.

Los GPO se pueden asociar a diferentes contenedores de un AD: un sitio, un dominio o una unidad organizativa, y son verificados en ese orden para determinar las políticas aplicadas a una computadora o usuario final.

Independientemente del OU al que pertenece la computadora o el usuario, la autenticación se realiza a través del controlador de dominio para el dominio donde la cuenta reside. El proceso de autenticación en Windows se describe en la sección 2.6.2 que, para el caso de cuentas de dominio, el servicio *Netlogon* es el responsable de contactar con el controlador de dominio para validar las credenciales del usuario según el mecanismo de autenticación elegido (NTLM o Kerberos).

Uno de los aspectos más importantes del AD es su compatibilidad con LDAP. Su estructura jerárquica permite la creación de contenedores de dominio (Domain Component (DC)), unidades organizativas (OU) y objetos, como usuarios y computadoras, identificados mediante el Common Name (CN). El DN, que combina el CN, OU y DC como sufijos, identifica de manera unívoca a un objeto en el dominio.

Existen diferentes métodos utilizados para acceder a la información del AD. Las aplicaciones y clientes pueden usar las interfaces específicas de Windows como Win32 API y Messaging API, la interfaz orientada a objetos Active Directory

Service Interfaces (ADSI), o el propio estándar LDAP.

Los tipos de objetos que pueden ser creados, así como sus atributos o propiedades, están definidos en el esquema del AD mediante clases de objetos. Por defecto, dicho esquema no contiene los atributos necesarios para los usuarios de sistemas Linux/Unix, como el UID o el GID. Sin embargo, en escenarios donde se espera que coexistan ambos sistemas, se pueden agregar tales atributos para lograr la integración esperada. Mas adelante, en la sección 3.4, se describen las alternativas de integración disponibles.

3.2.2. FreeIPA

FreeIPA es un gestor de identidades y control de acceso creado y mantenido por el proyecto Fedora (patrocinado por RedHat) para entornos Linux/Unix. La versión para RedHat se conoce simplemente como IdM (*Identity Management*) o IPA (*Identity, Policy and Audit*)

FreeIPA permite crear almacenes de identidades, políticas de acceso, centralizar la autenticación y controlar los dominios de Kerberos y servicio DNS, todo usando herramientas nativas de Linux. Actualmente, FreeIPA es una de las únicas opciones de software libre disponibles para implementar un IAM en entornos Linux/Unix [64].

Los diferentes protocolos de autenticación y mecanismos de seguridad disponibles en los sistemas Linux, como Kerberos, PAM y sudo, son complejos y difíciles de administrar coherentemente, especialmente cuando se combinan con los distintos almacenes de identidades o servicios de directorio, como NIS y LDAP.

FreeIPA provee una capa de abstracción que unifica todos estos servicios dispares y simplifica la tarea de administrar los usuarios, sistemas y políticas de seguridad. Principalmente, se compone de los siguientes servicios:

Servicio de Directorio Implementado a través del Servidor de Directorio 389, forma parte del núcleo de FreeIPA. Compatible con LDAPv3, soporta una arquitectura multi-master permitiendo desplegar varios servidores FreeIPA para redundancia y balanceo de carga. Gestiona las entidades de todo el dominio y además sirve como back-end de datos para otros componentes del sistema: usuarios, grupos, hosts, servicios, reglas de control de acceso, datos de autenticación Kerberos, nombres de dominio DNS y certificados TLS.

Servidor de Autenticación FreeIPA utiliza el protocolo Kerberos versión 5 como servidor de autenticación y SSO. El servidor KDC se vale del servicio de directorio LDAP como back-end para almacenar los datos de autenticación de usuarios y servicios.

Autoridad de Certificación FreeIPA puede ser configurado para funcionar junto a un servidor de certificados X.509 o CA. Los hosts que se unen al dominio pueden obtener un certificado del CA y usarlo en los servicios donde se requiera autenticación SSL/TLS. Se puede utilizar el servidor para otorgar, renovar y revocar certificados.

Servidor DNS El servicio de nombres de dominio se integra a la arquitectura de FreeIPA para ofrecer actualizaciones automáticas de los registros DNS para los hosts y los servicios como Kerberos, LDAP y CA.

Servidor NTP Servicio esencial para mantener sincronizado los relojes de todos los hosts que participan del dominio.

Tanto los servicios esenciales, LDAP y Kerberos, como los servicios adicionales, CA, DNS y NTP, son coordinados y controlados por el servidor FreeIPA a través de un conjunto de herramientas administrativas vía CLI e interfaz web. En la figura 3.3 se puede observar la arquitectura completa del servidor FreeIPA.

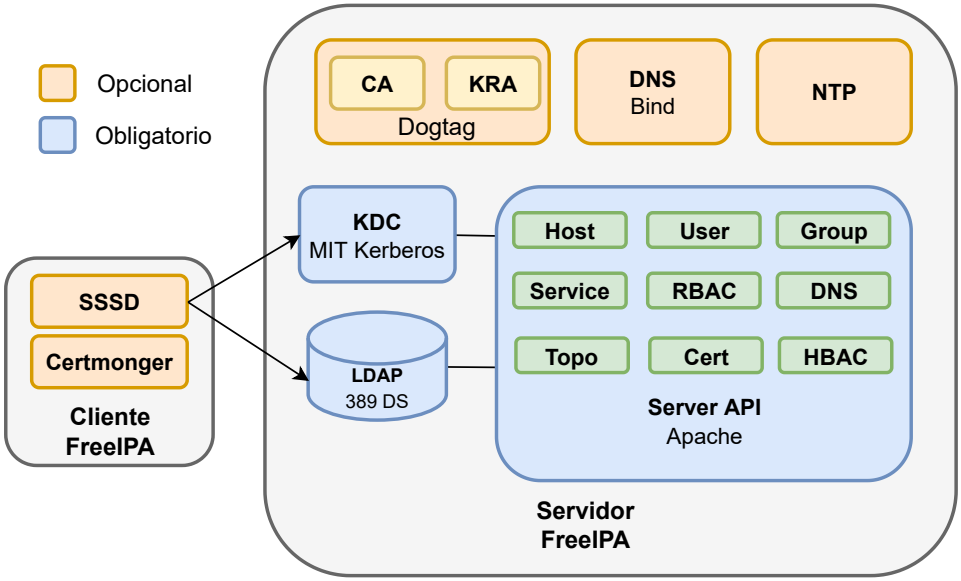

Figura 3.3: Arquitectura de FreeIPA

Es importante destacar que el servidor FreeIPA únicamente se ejecuta sobre sistemas Linux Fedora, Centos o RedHat. No así el cliente, que puede ser instalado sobre otras distribuciones de Linux como Debian, Suse y sus derivados.

El paquete System Security Services Daemon (SSSD) es el núcleo del cliente FreeIPA. La integración con el servidor FreeIPA es completa, ya que forma parte

del mismo proyecto, aunque también se lo puede utilizar junto a otros proveedores de identidad y autenticación como Active Directory, OpenLDAP o Kerberos de manera independiente.

Para su funcionamiento SSSD provee de interfaces NSS y PAM (*responders*), y de un sistema de back-end configurable para conectar con diferentes fuentes de identidad y autenticación. En la figura 3.4 se puede observar la arquitectura del paquete SSSD.

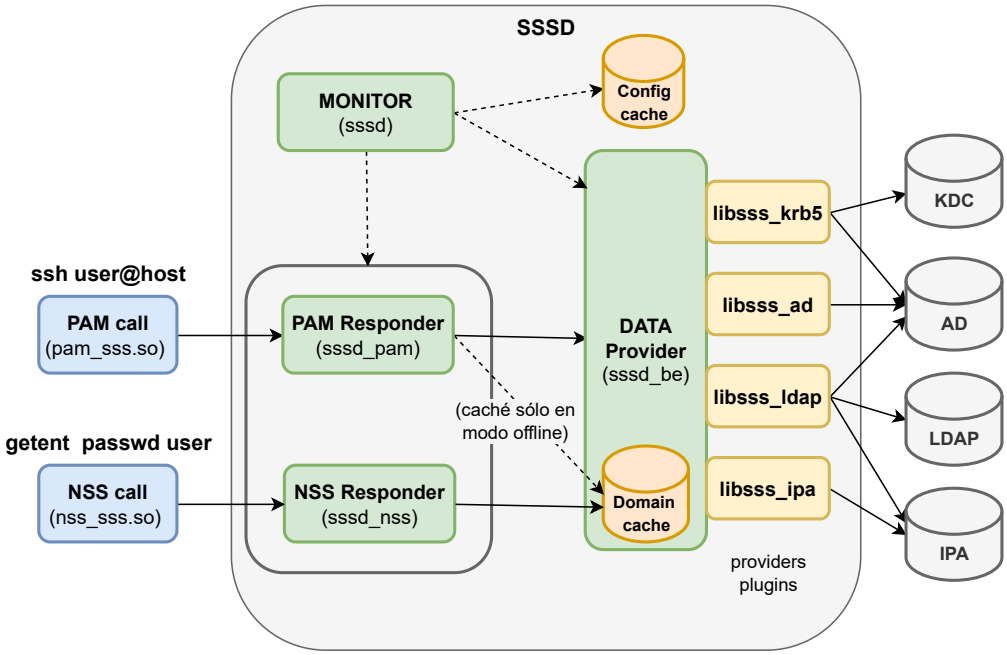

Figura 3.4: Arquitectura de SSSD

3.2.3. Otros IAM

Aparte de Windows AD y FreeIPA, existen otras soluciones y herramientas que acompañan a los mencionados IAM.

Tal es el caso del trabajo realizado por [65], donde se expone el diseño e implementación de un sistema de autenticación de identidades unificado basado en *ApacheDS* y Central Authentication Service (CAS).

ApacheDS es un servidor de directorio extensible e integrable escrito completamente en Java, que ha sido certificado por el Open Group como compatible con LDAPv3. Además del servicio de directorio, ApacheDS incluye un servidor

KDC compatible con Kerberos v5, soporta replicación multi-master y puede ser instalado en diferentes plataformas, como Windows y Linux.

CAS es un solución ESSO multi-lenguaje construida para dar soporte de autenticación y autorización a aplicaciones del tipo web. La solución está compuesta por un servidor CAS, desarrollado con el framework Spring de Java, y los clientes CAS, que pueden integrarse a las aplicaciones web del usuario mediante diferentes plataformas de software: Apache httpd Server (módulo mod_auth_cas), Java (Java CAS Client), PHP (phpCAS), Python (pycas) y Ruby (rubycas-client).

El servidor CAS es responsable de autenticar a los usuarios y otorgar el acceso a los servicios habilitados (clientes CAS). Para ello utiliza el concepto de TGT para crear la sesión SSO y de Service Ticket (ST) para acceder al servicio solicitado; similar en funcionamiento al protocolo Kerberos, pero orientado a entornos web mediante el uso de redirecciones HTTP y cookies de sesión.

Las especificaciones actuales del protocolo CAS se describen en [66]. Además del protocolo CAS, los clientes se pueden comunicar con el servidor vía otros protocolos de autenticación web como SAML, OAuth y OIDC, descriptos en la sección 2.7.9 de este trabajo.

El servidor CAS implementa manejadores para LDAP, bases de datos relacionales y SPNEGO como mecanismos de autenticación en el backend, por lo que se puede integrar con ApacheDS, OpenLDAP, Postgresql, Active Directory o Kerberos.

El trabajo presentado por [1] también propone una manera de diseñar e implementar un servicio de autenticación unificado, basado en un servidor AD y una módulo de software intermedio que permite manejar las peticiones de autenticación desde diferentes tipos de aplicaciones utilizando el protocolo HTTPS como transporte.

En [67] se propone integrar servidores Linux y Solaris al servicio de autenticación de Active Directory mediante la suite de integración Centrify.

El paquete de software Centrify consta de varios componentes que juntos proporcionan una capa de integración entre el entorno AD basado en Microsoft Windows y los sistemas de información que se ejecutan en una variedad de otros sistemas operativos o entornos.

En los últimos años surgieron varias soluciones de IAM en la nube (Identity as a Service (IDaaS)), con el objetivo fundamental de evitar costos y dificultades técnicas derivadas de la implementación y mantenimiento de un servicio hospedado de manera local. Alguno de los proveedores más conocidos que ofrecen este tipo de servicios son: Microsoft Azure Active Directory, Amazon Cognito, Auth0, Cisco Duo y Okta.

Para implementaciones de nube privada se puede nombrar a Keycloak: un producto de software libre que permite la autenticación SSO y la gestión de identidades para aplicaciones web modernas a través de los conocidos protocolos SAML

y OIDC. Además soporta el protocolo LDAP y Kerberos para la integración con los servicios de directorio tradicionales en el backend.

El inconveniente con los servicios de autenticación en la nube es que sólo permiten integrar aplicaciones del tipo web con soporte para los protocolos SAML, OAuth y OIDC. Por lo que, se sigue dependiendo de un IAM local para las aplicaciones nativas o sin el soporte de los mencionados mecanismos de autenticación web.

Más adelante, en la sección 3.4, se discuten las alternativas de IAM planteadas en esta sección pero con un enfoque orientado a solucionar la gestión de autenticación centralizada de usuarios en entornos heterogéneos. Antes de tal discusión, se analiza y describe a continuación el entorno del caso de estudio con el fin de facilitar la selección de la solución pretendida.

3.3. Descripción del entorno de trabajo/caso de estudio

El TEPM, creado por la Constitución Provincial, es uno de los organismos de la Constitución de carácter permanente y tiene como misión fundamental la custodia de los Derechos Políticos, tanto de los ciudadanos como de los partidos; como así también la de los principios constitucionales y legales consagrados en materia electoral. Funciona, en cumplimiento de mandato Constitucional, en el mismo edificio de la Honorable Cámara de Representantes de la Provincia, ubicada en la calle Alberdi 690 de la capital provincial. Desde sus inicios en el año 1958 a la fecha, se remarcan varios hechos históricos de trascendencia electoral:

la creación del Registro de Electores Extranjeros;

en 1965 se sanciona la ley de partidos políticos 277, que fuera hoy día reemplazada por la Ley 4081 XI -7;

la modificación del Art. 110 de la Constitución Provincial, creando el Consejo de la Magistratura;

la reforma de la Constitución Nacional de 1994, que modifica el sistema de elección indirecta por la elección directa (se deja de elegir por medio de colegios electorales, pasándose a elegir en forma directa a nuestros representantes);

se sanciona la Ley 2771 que establece el régimen electoral de lemas provincial y municipal, suprimiéndose posteriormente el provincial, quedando vigente al día de la fecha únicamente el sistema de lemas municipal;

por último en la Ley Electoral -Ley XI N 6-, se estableció la edad de dieciséis años para electores nacionales, siendo éste derecho facultativo e incorpora la *digitalización del proceso electoral* y con ello dispone la *implementación electrónica en la emisión de sufragio y sistema de escrutinio*, aplicándose por primera vez en la provincia un sistema de votación asistido electrónicamente, creación de éste organismo.

Para llevar adelante sus actividades el TEPM cuenta con un sistema informático denominado SGI, donde se desarrollan las mayoría de las tareas operativas y de soporte del organismo:

Operativas:

- Mesa de Entradas
- Padrón de Extranjeros
- Partidos Políticos
- Proceso Electoral
- Notificaciones Electrónicas

Soporte:

- Sistema de Expedientes
- Archivo
- Inventario
- Personal
- Sueldos
- Sistema de Tickets
- Gestión de Usuarios

El SGI también da soporte al sistema de gestión de calidad, que permite al TEPM ofrecer un servicio de calidad según los estándares internacionales. Desde el año 2011 hasta la actualidad el TEPM mantiene la certificación ISO 9001 de Gestión de Calidad.

Además del SGI, se utilizan otros sistemas informáticos como correo electrónico, gestor de contenidos web y archivos compartidos. Todos estos sistemas necesitan sus propias credenciales de acceso (usuario y clave) que el administrador de IT debe gestionar en forma separada para cada sistema. A esto se suman las

credenciales para el acceso a los equipos informáticos (PCs, notebooks, equipos de networking) y los servicios de red (VPN, WiFi, 802.1X). En la sección 3.3.1 y 3.3.2 se describe con mayor detalle los procesos más comunes de autenticación a los servicios y autenticación a la infraestructura respectivamente.

3.3.1. Autenticación a las Aplicaciones y Servicios

El SGI se encuentra desarrollado sobre una plataforma web conformada por:

- Linux, como sistema operativo
- Apache, como servidor web
- Postgresql, como base de datos
- PHP: Hypertext Preprocessor (PHP), como lenguaje principal de desarrollo

La autenticación de usuarios del SGI se realiza a través de la autenticación de usuarios del motor de base de datos Postgresql. Por lo tanto, cada usuario del SGI tiene creado un usuario o *login role* correspondiente en el Postgresql.

El archivo *pg_hba.conf* permite configurar los diferentes métodos de autenticación que soporta el Postgresql: password/MD5, certificado, GSS-API, SSPI, LDAP, RADIUS y PAM, entre otros. La configuración actual del SGI autentica a los usuarios mediante passwords encriptados con MD5.

Para no tener que manipular directamente los usuarios de Postgresql, el SGI tiene su propio módulo de gestión de usuarios que permite administrar las cuentas de usuario a través de la plataforma web.

El proceso de autenticación se inicia en el cliente con un formulario de login HTML mediante el cual el usuario envía sus credenciales al servidor web Apache, que verifica la validez de las mismas utilizando una función PHP de conexión al servidor Postgresql. Para proteger las credenciales que se envían del cliente al servidor web se utiliza una conexión protegida con HTTPS.

Si las credenciales son válidas, se inicia una sesión de PHP y se muestra la página de inicio del SGI junto a los módulos de menú habilitados según el nivel de permisos del usuario.

Los permisos del SGI son administrados con el esquema de roles y privilegios del Postgresql. Los roles permiten agrupar a los usuarios según las tareas o funciones que realiza en el sistema. Un usuario es asignado a uno o más roles para definir el nivel de permisos de éste. En el TEPM se definieron roles por área, por jefes de área y algunos roles especiales.

Cabe aclarar que, además del SGI descripto con anterioridad, el departamento de sistemas del TEPM se encuentra desarrollando un nuevo SGI donde se enmarca el módulo de gestor de usuarios centralizados que se describe en el capítulo 4.

Otro de los sistemas informáticos que respaldan las tareas cotidianas del TEPM es el servidor de correo electrónico. Dicho servidor se encuentra implementado con Zimbra Collaboration Server (ZCS), una suite de colaboración y mensajería completa que incluye email, libreta de direcciones, calendario y lista de tareas.

La arquitectura de Zimbra integra software de código abierto utilizando protocolos estándares de la industria. La siguiente lista muestra los paquetes de terceros más importantes que se incluyen en Zimbra y que han sido probados y configurados para trabajar en conjunto:

- Jetty, como servidor web.
- Postfix, Mail Transfer Agent (MTA) de código abierto que rutea los mensaje al servidor de email correspondiente.
- OpenLDAP, implementación de código abierto de LDAP que almacena la configuración de sistema de Zimbra, la lista de direcciones globales y los proveedores de autenticación. Zimbra puede también trabajar con servicios de directorio LDAP externos como Active Directory.
- MySQL, base de datos donde se almacena el metadato de los mensajes de correo, lista de tareas y calendario.
- Lucene, motor de búsqueda de código abierto para texto
- ClamAV, escáner antivirus
- SpamAssassin, filtro antispam
- Amavisd-new, interfaz entre el MTA y los verificadores de contenido
- James/Sieve, software utilizado para el filtrado de los emails

Zimbra soporta varios mecanismos de autenticación que se controlan con el atributo *zimbraAuthMech*. Por defecto, se encuentra configurado para autenticar con el servicio de LDAP interno, pero es posible también configurar para que autentique con un servidor LDAP externo, con Active Directory o con Kerberos de manera directa. El cliente web de Zimbra además soporta la autenticación con SPNEGO, lo cual permite integrar el servicio de email a un sistema de autenticación SSO.

El servidor de archivos del TEPM está implementado con *Samba*, una suite de programas de código abierto para Linux que implementa el protocolo de archivos compartidos de Windows SMB/CIFS. Samba se encuentra perfectamente integrado a la mayoría de las distribuciones de Linux actuales lo que permite trabajar en entornos multiplataforma.

La autenticación del servidor Samba se configura con la sentencia *security* en el archivo de configuración *smb.conf*. A través de esta sentencia se configuran los diferentes *security modes* que soporta Samba: *share, user, domain, ADS* y *server*. Por defecto, Samba se encuentra configurado en el modo *security = user*, lo que permite verificar el nombre de usuario y clave contra la base de datos local de Samba: *passdb backend = tdbsam*.

El TEPM maneja tres tipos de acceso a los archivos compartidos con Samba: público, por área y por usuario. En el nivel de acceso público todos los usuario del TEPM tienen acceso al recurso compartido. En el nivel de acceso por área los usuarios tienen acceso sólo a los recursos asignados al área que le corresponde. Y el nivel de acceso por usuario permite acceder sólo al recurso asignado a un usuario en particular. La sentencia *valid users* en el archivo de configuración de Samba permite configurar los niveles de acceso descriptos.

La página web del TEPM esta implementada con el administrador de contenidos *Joomla*. Joomla opera sobre una plataforma Linux, Apache, MySQL y PHP (LAMP). A diferencia de los otros servicios descriptos con anterioridad, son pocos los usuarios del TEPM que gestionan los contenidos de la página web. La autenticación de los mismo se realiza con el propio gestor de usuarios incorporado en Joomla, y las credenciales de acceso se guardan en la base de datos MySQL. Para determinar que pueden ver y hacer los usuarios (autorización) Joomla define grupos de acceso a los cuales se les asigna los permisos y niveles de acceso correspondientes. Luego, un usuario es asignado a uno o más de estos grupos para determinar su nivel de acceso final.

Joomla incluye los siguiente grupos predefinidos:

Public Grupo base que no posee ningún tipo de permiso y sólo puede ver los elementos del sitio que estén publicados con el nivel de acceso público.

Guest Grupo hijo de Public (hereda sus permisos y niveles de acceso), utilizado por defecto para los nuevos usuarios y por los visitantes de la página web que no se encuentran registrados.

Registered Grupo hijo de Public que permite login al front-end y ver elementos del sitio configurados con el nivel de acceso para usuarios registrados.

Author Grupo hijo de Registered que permite crear y editar sus propios artículos o contenido en el front-end del sitio.

Editor Grupo hijo de Author que agrega el permiso de poder editar los artículos y contenido de terceros.

Publisher Grupo hijo de Editor que agrega el permiso de poder cambiar el estado de los artículos, por ejemplo publicarlos en el front-end del sitio web.

Manager Grupo hijo de Public que posee todos los permisos, tanto en el front-end como en el back-end, excepto el de acceso a la configuración de los componentes y configuración global de Joomla.

Administrator Grupo hijo de Manager que agrega el permiso de acceso a la configuración de los componentes de Joomla. Lo único que no tiene permitido hacer este grupo es la configuración global de Joomla y la creación o modificación de las cuentas de usuario SuperUser.

SuperUser Grupo hijo de Public que posee todos los permisos sobre el sistema.

Joomla incorpora de manera nativa (sin el agregado de plugins) la posibilidad de configurar autenticación de doble factor mediante el uso de una aplicación compatible con OTP, como Google Authenticator, o el uso de tokens de seguridad por hardware, como Yubikey.

Otra de las opciones que incorpora Joomla es la de autenticación de usuarios mediante LDAP, lo que permite integrar la base de datos de usuarios con otros sistemas compatibles con LDAP.

Todos los servicios descriptos con anterioridad se encuentran implementados sobre el entorno de virtualización *Proxmox VE*. Esta plataforma permite crear máquinas virtuales (Virtual Machine (VM)) donde se instalan y configuran los diferentes servicios y aplicaciones. El soporte físico de las VMs es un conjunto o clúster de servidores que permite gestionar de manera centralizada todo el datacenter.

La gestión centralizada de las VMs se realiza a través de una interfaz web. Por defecto, el acceso a dicha interfaz es a través del usuario root de Linux habilitado en cualquiera de los servidores físicos del cluster.

Si es necesario, Proxmox permite crear a través de su gestor de usuarios, otros usuarios y asignarles diferentes privilegios de acceso gracias a la administración de permisos basado en roles. Toda la configuración de usuarios se almacena en el archivo */etc/pve/user.cfg*, salvo las contraseñas de acceso que dependen del método de autenticación utilizado: PAM, PVE authentication server, LDAP o Active Directory. La autenticación configurada por defecto es mediante PAM. Además del método de autenticación seleccionado, Proxmox admite autenticación de doble factor a través de TOTP o Yubikey OTP.

3.3.2. Autenticación a la Infraestructura

El TEPM utiliza en su mayoría el sistema operativo Linux, más específicamente la distribución Debian y sus derivados como Ubuntu. El uso de computadoras con el sistema operativo Windows se encuentra limitado a unos pocos usuarios

que necesitan trabajar con aplicaciones específicas que se ejecutan sólo en este tipo de sistemas.

La autenticación de los usuarios del TEPM en los sistemas Linux es a través del módulo PAM estándar *pam_ unix.so*. Este módulo realiza llamadas estándares a las librerías del sistema para obtener información de las cuentas y realizar la autenticación. Por defecto, esta información se obtiene de los archivos de sistema */etc/passwd* y */etc/shadow*. Además de la autenticación local estándar, PAM permite a través de los siguientes módulos otros métodos de autenticación:

- pam_ ldap.so
- pam_ krb5.so
- pam_ radius.so
- pam_ sss.so

Como se describe en 2.6.1, los permisos en Linux son administrados a través de los atributos UID y GID que se obtienen luego del proceso de login del usuario. Estos atributos determinan el nivel de acceso a los recursos del sistema según los permisos definidos en el sistema de archivos.

La configuración actual del TEPM requiere que las cuentas de usuario y grupo se gestionen de manera local en cada computadora. Esto dificulta la tarea del administrador al momento de realizar cambios o actualizaciones en un entorno donde la cantidad de computadoras supera el centenar. Para resolver este inconveniente, Linux permite configurar las cuentas de usuario en una base de datos central como NIS o LDAP a través del servicio NSS.

La autenticación de los usuarios del TEPM en los sistemas Windows también se gestiona de manera local en cada computadora. Como se describe en 2.6.2, la información de las cuentas de usuario locales se almacenan en la base datos SAM. Luego del proceso de login el usuario obtiene un token de acceso compuesto por el SID de usuario, los SIDs de grupo a los que pertenece y los derechos de usuario asignados a éste. Las tareas que el usuario puede realizar y los recursos a los cuales puede acceder están determinados por los derechos de usuario y los permisos o ACL de recursos respectivamente.

Windows también permite centralizar las cuentas de usuario mediante la implementación de un servidor o controlador de dominio AD. Si no se dispone en la red de un controlador de dominio se puede implementar un método mixto donde la cuentas de usuario son creadas localmente pero modificadas para autenticar contra un servidor de autenticación Kerberos (KDC) externo.

El acceso remoto a las computadoras Linux se realiza mediante el protocolo SSH. El servicio SSH lo brinda el paquete *openssh*. Este paquete se encuentra configurado para autenticar mediante PAM. Por lo tanto, la autenticación del

usuario cuando accede de manera remota a la computadora se desarrolla de la misma manera que una autenticación local. Sin embargo, es posible configurar otros métodos de autenticación, por ejemplo, mediante claves públicas, OTP, Kerberos y GSS-API.

Con respecto a la infraestructura de red, el TEPM dispone de una variedad de equipos de networking, que incluye switches Cisco y HP, routers/firewalls Mikrotik y puntos de acceso inalámbrico de la marca Engenius y Ubiquiti. La gestión de estos equipos se realiza mediante la interfaz web que presenta cada uno, permitiendo un acceso controlado mediante usuario y clave almacenados en una base de datos local. La mayoría de los equipos de networking añaden además una administración por línea de comando o CLI. Para ello, se accede mediante SSH utilizando las mismas credenciales que el acceso web.

Las computadoras del TEPM se conectan a la red cableada Ethernet (IEEE 802.3) sin ningún tipo de restricción. Si bien los switches instalados soportan el protocolo IEEE 802.1x de control de acceso basado en puerto, el mismo no se encuentra implementado.

Para las computadoras portátiles se utiliza la red inalámbrica WiFi (IEEE 802.11). El acceso a dicha red se realiza a través de una clave pre-compartida o Pre-Shared Key (PSK) utilizando el protocolo de seguridad WPA2-PSK. Al ser una clave compartida entre todos los usuarios inalámbricos la seguridad se encuentra limitada a la protección de esta única clave.

Un método más robusto es utilizar el protocolo WPA2-Enterprise disponible en la mayoría de los puntos de acceso compatibles con IEEE 802.11. Este protocolo permite autenticar los clientes inalámbricos mediante usuario y clave disponibles en un servidor RADIUS, al igual que el protocolo IEEE 802.1x para redes cableadas.

En determinadas ocasiones, se requiere un acceso remoto a la red del TEPM. Para estos casos, se encuentra configurado en el firewall de la red un acceso por VPN utilizando el conjunto de protocolos IPsec/IKE. Las credenciales utilizadas en el acceso de la VPN son administradas localmente en el firewall.

La figura 3.5 resume los procesos de autenticación a los servicios e infraestructura utilizados actualmente en el TEPM. En la misma se puede observar como la autenticación depende de la configuración y repositorio de usuarios propios a cada sistema.

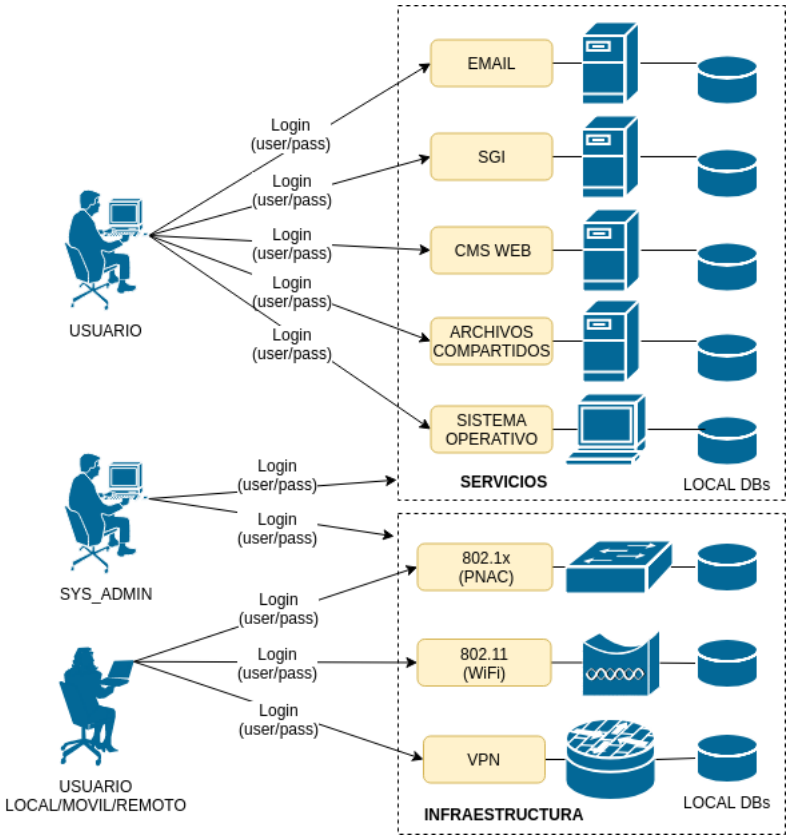

Figura 3.5: Esquema ilustrativo de los proceso de autenticación a los servicios e infraestructura del TEPM

3.4. Utilizando Kerberos en entornos heterogéneos

Resulta difícil definir adecuadamente el término *entorno heterogéneo o mixto* en el ámbito informático. Por lo general, significa que coexisten en un mismo entorno de red sistemas mixtos, esto es, sistemas Windows, Linux, Sun y/o Apple. En la mayoría de los casos se trata de entornos Windows donde se integran equipos de trabajo que no son Windows. En otros casos se trata de entornos no Windows (por ejemplo, Linux) donde se agregan estaciones de trabajo Windows. Este último es el caso del TEPM.

Además del tipo de sistema operativo, la heterogeneidad viene dada también por el mix de aplicaciones y servicios implementados: servicios de directorio, servicios de nombres, mecanismos de control de acceso, aplicaciones de escritorio y/o aplicaciones web. Por ejemplo, un entorno heterogéneo puede usar Kerberos junto con otros mecanismos de autenticación. Inclusive si utiliza sólo Kerberos,

puede combinar diferentes implementaciones: MIT Kerberos, Heimdal Kerberos y Windows Active Directory.

Luego de la descripción realizada en 3.3, se puede afirmar que el TEPM presenta un entorno de red heterogéneo y complejo. Para lograr el objetivo de gestionar de manera centralizada la autenticación de usuarios se debe proponer uno o más métodos o protocolos que compatibilicen con las aplicaciones, servicios e infraestructura de red presentes. Además de una autenticación centralizada, la solución elegida debe ser capaz de permitir al usuario un inicio de sesión único SSO.

Según [65], el núcleo de un sistema de autenticación de identidad unificada incluye principalmente dos partes. Uno es el servicio de directorio; el otro es el sistema de inicio de sesión único. La selección de productos y el diseño de estas dos partes son cruciales para la implementación, funcionamiento y mantenimiento del sistema.

Como se vio en 3.2.1 y 3.2.2, existen alternativas de IAM/SSO tanto para entornos Windows como Linux respectivamente. La elección de uno u otro va a depender del tipo de entorno. Para el caso de entornos mixtos dependerá de la cantidad de dispositivos con Windows y Linux.

Si la mayoría de los sistemas utilizados por la organización son Windows, habría preferencia por el uso de un AD con la posibilidad de integrar de manera parcial/directa sistemas Linux a dicho entorno. Existen varias formas de integrar Linux a un dominio con AD, cada una con sus pro y sus contra [68]

- Plugin de terceros: Requiere la instalación de software de terceros tanto en el cliente como en el AD. El costo está sujeto a la cantidad de sistemas Linux a integrar. La ventaja es que que se puede gestionar todo desde el AD: identidades, autenticación y políticas de acceso. Un buen ejemplo de este método se presenta en el trabajo realizado por [67], donde se utiliza la suite de integración Centrify.

- Legado: Utiliza las librerías estándar de Linux (pam_ krb5, pam_ ldap, nss_ ldap, nslcd) para conectar con los servicios LDAP/Kerberos del servidor AD. Requiere activar y configurar la extensión Identity Management for Unix (IMU) en el AD para el mapeo de los SID de usuarios Windows con los UID de Linux. Esta opción no tiene costos extras pero implica una configuración más compleja y además las políticas de acceso no se pueden administrar de manera centralizada.

- Tradicional: Método bien conocido que se integra mediante el servicio *winbind* de Samba. Además de los protocolos LDAP/Kerberos, utiliza los protocolos nativos del AD. No requiere de la extensión IMU, pero se debe unir el sistema Linux al dominio del AD a través de las herramientas *net join* o

realmd. Unicamente las políticas de *sudo* y *automount* se pueden gestionar desde el AD, el resto se debe hacer de manera local o centralizar a través de herramientas externas como Puppet o Ansible.

- Contemporáneo: Utiliza el paquete SSSD y *realmd* para una integración más flexible y escalable (ver figura 3.6). Posee las mismas ventajas que el método tradicional y además incorpora el soporte a múltiples fuentes de identidad y autenticación, y un sistema de cache para trabajar de manera offline. Como se describe en la sección 3.2.2, SSSD se integra de manera completa a FreeIPA y, llegado el caso, se puede configurar para realizar una integración indirecta con un AD a través de dominios de confianza entre ambos servidores.

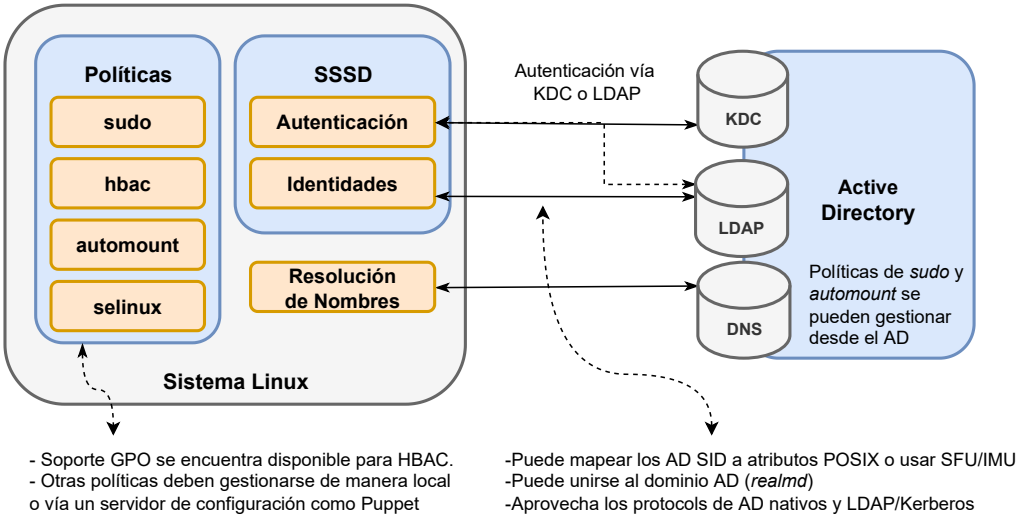

Figura 3.6: Integración directa de un cliente Linux a un servidor AD mediante SSSD

Si la mayoría de los sistemas ejecutan Linux lo más probable es que se utilice una alternativa IAM de código abierto como FreeIPA y se integre de manera parcial/directa los sistemas Windows.

En tal escenario, las opciones de integración son más escasas y se limita a configurar los sistemas Windows (por ejemplo, mediante la utilidad *ksetup*) únicamente para la autenticación contra el servidor KDC estándar. Las identidades y políticas de acceso no se pueden administrar de manera centralizada, y los usuarios se deben gestionar tanto en el servidor FreeIPA, como de manera local en los clientes Windows.

En grandes organizaciones, donde existen una gran cantidad de sistemas con Windows y Linux la opción más acertada es configurar para que trabajen en conjunto un servidor AD y un servidor FreeIPA mediante una relación de confianza entre ambos dominios (integración indirecta). Esta solución utiliza la capacidad de Kerberos para establecer relaciones de confianza entre dominios/realms, habilitando a los usuarios de un dominio autenticar en otro dominio sin la necesidad de replicar su identidad.

Los trabajos y documentos analizados en [67], [1] y [69], proponen la integración de entornos de autenticación heterogéneos a un servidor centralizado con Windows Active Directory. Durante el desarrollo del presente trabajo no se encontró una solución concreta donde la integración se pueda realizar mediante un servidor con Linux y utilizando herramientas de software libre.

Como se mencionó anteriormente, en el caso del TEPM existe un entorno con estaciones de trabajo Linux donde se agregan unas pocas estaciones de trabajo con Windows. Por lo tanto, para gestionar la autenticación centralizada se podría utilizar FreeIPA. Utilizar este tipo de IAM implica instalar y configurar una gran cantidad de paquetes de software además de modificar código fuente de terceros para adaptar a las necesidades de integración con los sistemas del TEPM; por lo que se debe encontrar una solución que se ajuste de manera más precisa y sencilla, aunque ésto implique un desarrollo propio.

De los métodos de integración descriptos con anterioridad, y del estudio y análisis realizado en las secciones previas, se observa que la mayoría de los sistemas y herramientas de IAM utilizan protocolos en común, como LDAP y Kerberos, para lograr la gestión de identidades y autenticación de usuarios.

Teniendo en cuenta esta observación y al análisis hecho sobre las aplicaciones y servicios presentes en el TEPM, se deduce que la forma de obtener autenticación centralizada de usuarios e inicio de sesión único es incorporando Kerberos como componente principal a la infraestructura del TEPM.

Siguiendo esta hipótesis, los detalles relacionados al soporte de Kerberos en un entorno mixto caen en tres áreas básicas:

- Hacer posible que los usuarios utilicen sus credenciales de Kerberos para el login del sistema operativo.
- Habilitar al sistema operativo que guarde en cache dichas credenciales y que participe de un esquema SSO
- Permitir que las aplicaciones hagan uso de Kerberos.

Kerberos, como método de autenticación, esta presente prácticamente en la mayoría de las aplicaciones y servicios. En máquinas Linux se puede usar, por ejemplo, el módulo *pam_ krb5.so* o el módulo *pam_ sss.so* desarrollado, justamente, para el proyecto FreeIPA. Para el funcionamiento de ambos módulos se

debe instalar el paquete *krb5-user* que contiene los programas y librerías básicos para autenticar contra un servidor KDC.

Para sistemas Windows se puede utilizar el cliente de Kerberos (Kerberos For Windows (KFW)) desarrollado por el MIT, o la utilidad *ksetup* incorporada por defecto en las variantes de Windows Server y Profesional. Cual de las dos opciones implementar dependerá de las aplicaciones que se utilizarán:

- Las aplicaciones que implementan el protocolo Kerberos de manera nativa en teoría pueden operar con cualquier implementación de KDC aunque el protocolo en sí no especifica como la aplicación debería interoperar con la implementación de Kerberos local (por ej., donde encontrar el ticket del usuario o las variables de entorno para Kerberos). Afortunadamente este tipo de aplicaciones no son lo habitual.

- Las aplicaciones que soportan solo SSPI (por ej., Microsoft Outlook, Internet Explorer o Windows File Sharing) deben utilizar la impementación nativa de Kerberos para Windows.

- Las aplicaciones que soportan solo GSS-API deben usar KFW.

- Y las aplicaciones que soporten SSPI y GSS-API (como la mayoría de las aplicaciones de terceros) pueden utilizar ya sea la implementación nativa de Kerberos para Windows o KFW.

Para los servicios utilizados en el TEPM, el servidor de correo Zimbra soporta de manera nativa la autenticación con Kerberos a través de la configuración del atributo *zimbraAuthMech*, al igual que el servidor SAMBA a través de la la sentencia *security = ADS*.

El SGI funciona con el servidor web Apache por lo que se puede adaptar para autenticar mediante Kerberos utilizando el módulo *libapache2-mod-auth-kerb* o el *libapache2-mod-auth-gssapi*.

La integración del gestor de contenidos web con Kerberos se logra mediante la configuración de una autenticación LDAP/SASL-GSS-API.

Para finalizar, con respecto a los dispositivos de red, la mayoría soportan el protocolo RADIUS que se puede integrar a Kerberos y así autenticar los usuarios de la red a un servidor KDC.

Dicho todo esto, para lograr la autenticación centralizada de usuarios en un entorno heterogéneo como lo es el TEPM, se debe principalmente incorporar a la infraestructura un servidor de autenticación Kerberos junto a un servidor LDAP que simplifique la gestión de identidades. Cabe recordar que para su funcionamiento el servidor de Kerberos necesita de un servidor de nombres DNS y un servidor de tiempo NTP, servicios que ya se encuentran implementados en el TEPM.

En la sección 3.5 se describe con mayor detalle los componentes necesarios para llevar adelante la arquitectura de autenticación propuesta.

3.5. Propuesta de integración

De las alternativas vistas en 3.4 para lograr la autenticación centralizada de usuarios, la opción elegida es la de una arquitectura independiente. Esta opción permite obtener la autenticación pretendida sin incorporar grandes modificaciones a la infraestructura actual del TEPM y sin la necesidad de implementar un IAM completo como el AD o FreeIPA. Además, a través de las librerías correspondientes, facilita el desarrollo de una interfaz de gestión de usuarios integrada al SGI. Por lo tanto, la arquitectura de autenticación propuesta consta de los siguientes componentes:

Servidor KDC Componente principal de la arquitectura. Proporciona el servicio de autenticación a los usuarios y servicios (*principals*). Disponible en Linux mediante la instalación y configuración del paquete *krb5-kdc*, implementación de referencia del MIT para Kerberos versión 5.

Suplicante de Host Módulo de software presente en las computadoras de usuario que provee las credenciales de acceso al servidor de autenticación Kerberos y almacena en caché el ticket del usuario. En sistemas Linux, se va a utilizar el módulo *pam_ sss.so* disponible mediante la instalación del paquete *sssd* y *sssd-krb5*. Para su funcionamiento es necesario además tener instalado *krb5_ user*, un conjunto de programas y librerías básicos para autenticar MIT Kerberos versión 5. En sistemas Windows se va a utilizar la herramienta *ksetup*, incorporada de manera nativa en las variantes de Windows Server y Profesional, que permite modificar el método de autenticación estándar por el de Kerberos.

Servidor RADIUS Centraliza la autenticación de los dispositivos y servicios de red. Disponible en Linux mediante la instalación y configuración del paquete *freeradius*. Se integra para autenticar con Kerberos a través del módulo *freeradius-krb5*.

Cliente de RADIUS Permite a un NAS (switch, router, AP, etc.) otorgar acceso a la red luego de autenticar a los usuarios utilizando el protocolo RADIUS. Forma parte del sistema operativo de la mayoría de los dispositivos de red. No es necesario instalar un paquete o módulo adicional. Dependiendo del tipo de NAS, la autenticación de usuarios con RADIUS puede estar disponible para el acceso a la red inalámbrica Wi-Fi (WPA2-Enterprise), la red cableada (IEEE 802.1x) y/o el acceso remoto VPN (IPsec/IKE).

Base de Datos de Usuarios Proporciona un repositorio central de usuarios para los servicios y sistemas del TEPM. Para su implementación se va a utilizar un servidor LDAP (*OpenLDAP*) disponible en el paquete de software para Linux *slapd*. Para utilizar dicho servidor como repositorio de usuarios para el KDC y el suplicante de host se deben instalar y configurar los paquetes *krb5-kdc-ldap* y *sssd-ldap* respectivamente.

Servidor DNS Necesario para resolver los nombres de los hosts donde se ejecutan los servicios kerberizados. Como se especificó en la sección 2.7.3, el cliente de Kerberos verifica a través del servicio de DNS la validez de los nombres de dominio antes de otorgar un ticket de servicio TGS.

Servidor NTP Mantiene sincronizado los relojes de todos los dispositivos de la red. Dicha sincronización es crítica para que la autenticación con Kerberos no falle (ver sección 2.7.3).

Gestor de usuarios Debe interactuar con la base de datos de usuarios y el servidor de autenticación KDC para administrar las cuentas de usuario de todos los sistemas y servicios (creación, modificación, eliminación de cuentas y permisos). Para su implementación se debe desarrollar un nuevo módulo en el SGI utilizando las librerías y herramientas necesarias para la comunicación con los servicios mencionados. En el capítulo 4 se describe con mayor detalle el desarrollo de dicho módulo.

En la figura 3.7 se observa la arquitectura de autenticación propuesta con los componentes necesarios para su implementación.

Además de los citados componentes, se deben tener instalados y configurados ciertos módulos y librerías para poder utilizar Kerberos en los servicios y sistemas del TEPM (kerberizar). La mayoría de las aplicaciones disponen de manera nativa la opción de autenticar con Kerberos, y en otras es necesario instalar algún módulo o librería extra. A continuación se citan los servicios del TEPM y la manera de integrarlos a la infraestructura de autenticación con Kerberos:

SGI Para seguir utilizando el motor de base de datos Postgresql como mecanismo de autenticación del SGI pero integrado al Kerberos, se debe modificar el método de autenticación del Postgresql en el archivo de configuración *pg_hba.conf*. En este caso lo conveniente es cambiar el método MD5 actual al método con PAM, ya que el servidor donde se ejecuta el Postgresql debe estar configurado con *pam_sss* y así autenticar con Kerberos. Para desarrollos nuevos, como el gestor de autenticación de usuarios, se puede utilizar directamente el módulo de autenticación *libapache2-mod-auth-kerb* del servidor web Apache para proteger el acceso con Kerberos a los directorios del

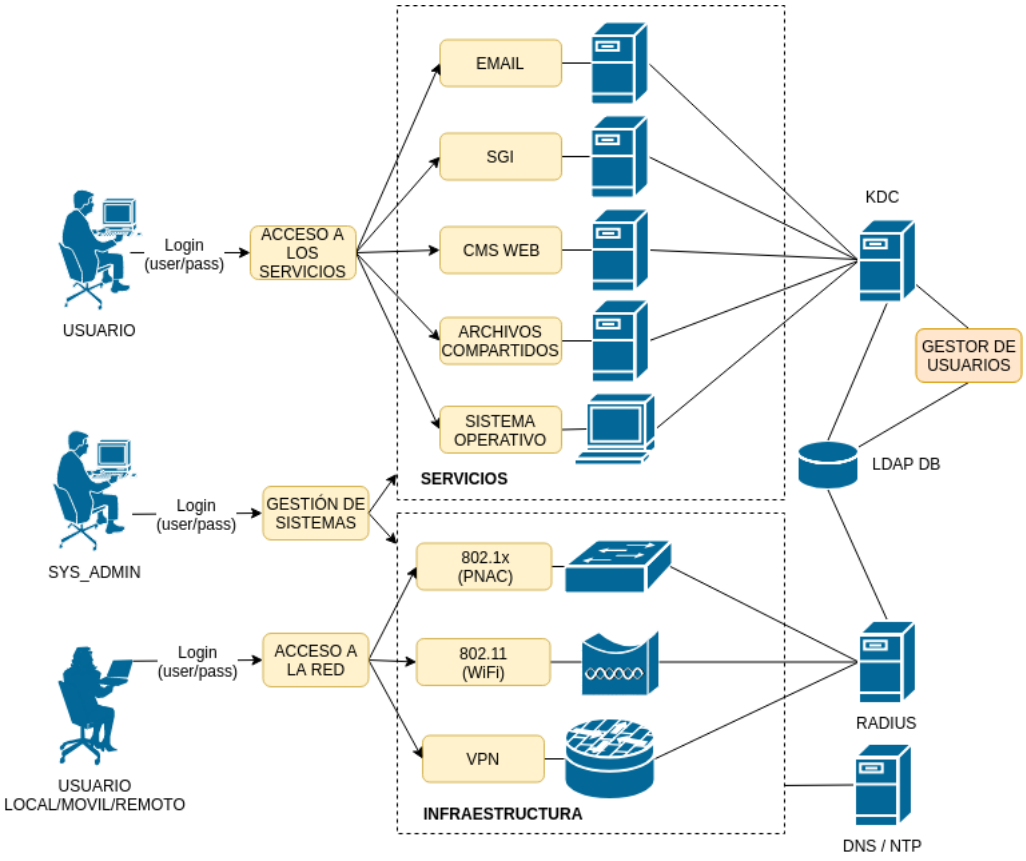

Figura 3.7: Esquema ilustrativo de la arquitectura de autenticación propuesta

sistema. En la sección 4.1 se describe con mayor detalle el esquema de autenticación propuesto para el SGI.

Servidor de Email Para autenticar con Kerberos es necesario modificar el atributo *zimbraAuthMech* a *kerberos5* y definir el dominio de autenticación en el atributo *zimbraAuthKerberos5Realm*. Es importante destacar que esto modifica unicamente el método de autenticación y no la base de datos de usuarios, que continua siendo el servidor LDAP interno del Zimbra. Por lo tanto la cuenta del usuario debe existir en ambos servidores LDAP: el interno del Zimbra y el externo utilizado con Kerberos.

Servidor de Archivos Para que el servidor de archivos Samba autentique con Kerberos se configura en el archivo *smb.conf* las sentencias $security = ADS$ y $realm = KERBEROS\_REALM$. Samba mapea de manera predeterminada los usuarios a autenticar con los usuarios del sistema operativo

donde se ejecuta. Por lo tanto, si se tiene configurado el servidor de archivos para que utilice LDAP como repositorio de usuarios, mediante el paquete *sssd-ldap*, no es necesario mantener una base de datos extra para Samba.

Servidor Web El servidor de contenidos Web Joomla sólo permite la autenticación local o mediante LDAP. Entonces, para integrar Joomla a la arquitectura de autenticación propuesta es necesario que el servidor LDAP utilice Kerberos como *back-end* de autenticación mediante el protocolo SASL: *Pass-Through authentication*. Para ello, es necesario instalar y configurar los paquetes *sasl2-bin* y *libsasl2-modules-gssapi-mit*, y definir el atributo *userPassword: SASLusername@realm* en los usuarios del LDAP.

Acceso remoto vía SSH El acceso remoto a estaciones Linux es mediante un servidor SSH, disponible con la instalación del paquete *OpenSSH*. Para que la autenticación sea con Kerberos es necesario configurar la directiva *GSSAPIAuthentication yes* en el archivo de configuración *sshd_config*.

Antes de continuar con el desarrollo del gestor de autenticación centralizado de usuarios, se debe tener implementada la infraestructura de red y servicios propuesta con anterioridad. En el anexo A se describen los pasos para su implementación sobre un servidor con *Ubuntu Server* como sistema operativo.

4. Gestor de Autenticación de Usuarios

El capítulo a continuación describe el diseño y desarrollo del módulo de software que permite gestionar y verificar el correcto funcionamiento de la infraestructura de autenticación propuesta en el capítulo 3. Tal módulo debe integrarse al SGI del TEPM, por lo que el capítulo comienza con una descripción de la arquitectura actual de dicho sistema. Luego se plantean lo requisitos funcionales del módulo gestor de usuarios y el diseño del mismo. El capítulo finaliza con la implementación final del gestor.

4.1. Arquitectura actual del SGI

Como se describe en 3.3.1, el SGI del TEPM se encuentra desarrollado sobre una arquitectura web cliente-servidor, donde el servidor se implementa a través de una plataforma Linux, Apache, Postgresql y PHP (LAPP). A esta plataforma se añade código de JavaScript para obtener el dinamismo necesario del lado del cliente. En la actualidad conviven dos desarrollos diferentes del sistema, los cuales comparten la misma plataforma LAPP pero con algunas diferencias sustanciales con respecto a su diseño.

El primer desarrollo del SGI sigue un modelo aproximado al modelo Modelo-vista-controlador (MVC) clásico utilizando las tecnologías incluidas en AJAX [70] para lograr una aplicación web enriquecida (RIA). Para la generación de las peticiones AJAX se utiliza la librería PHP Xajax. Dicha librería permite crear funciones AJAX sólo mediante código PHP y sin la necesidad de codificar en JavaScript.

El segundo desarrollo del SGI surge como una necesidad de actualizar y renovar la arquitectura de desarrollo anterior. Las librerías, funciones y versiones PHP del antiguo desarrollo fueron quedando obsoletas ante la continua actualización de las tecnologías web. Para ello, se decide implementar un sistema web siguiendo una arquitectura de diseño REST [71].

La arquitectura REST sigue una serie de principios para la implementación de una API (*back-end* del lado servidor) independiente del tipo de cliente (*front-end*), lo que añade flexibilidad, escalabilidad y seguridad en el diseño del sistema.

En este caso, para que el cambio de arquitectura sea paulatino, se decidió mantener los mismos lenguajes de programación que el sistema anterior: PHP en el back-end y HTML, Cascading Style Sheets (CSS) y JavaScript en el front-end. Se eliminó la librería Xajax, que ya no tenía soporte, y se añadieron las librerías Jquery y Bootstrap de amplio uso y soporte en la actualidad.

Sin embargo, esto no impide que en un futuro se puedan utilizar otras tecnologías como Python y Nodejs en el back-end, o clientes de escritorio/móviles desarrollados en otros lenguajes de programación en el front-end, siempre y cuando se respeten las pautas de diseño impuestas por la API.

Los principios básico de diseño que debe seguir una API REST para lograr las características mencionadas, son los siguientes:

Cliente-servidor Este principio opera sobre el concepto de que el cliente y el servidor deben estar separados entre sí y permitir que evolucionen individualmente.

Sin estado Las API REST no tienen estado, lo que significa que las llamadas se pueden hacer independientemente una de la otra, y cada llamada contiene todos los datos necesarios para completarse con éxito. Esto permite una mayor escalabilidad, ya que el servidor no tiene que mantener, actualizar o comunicar estados de sesión al cliente.

Caché Debido a que una API sin estado puede aumentar la sobrecarga de una solicitud, al manejar grandes cargas de llamadas entrantes y salientes una API REST debe diseñarse para alentar el almacenamiento de datos en caché.

Interfaz uniforme La clave para que el cliente se desacople del servidor es tener una interfaz uniforme que permita la evolución independiente de la aplicación sin tener los servicios, modelos o acciones de la aplicación estrechamente acoplados a la API. Dicha interfaz debe servir para identificar **recursos** (elementos de información) que se acceden de manera unívoca a través de un Uniform Resource Identifier (URI).

Sistema en capas Las API REST tienen diferentes capas que trabajan juntas para crear una arquitectura que ayude a tener una aplicación más escalable y modular.

Code on Demand (Opcional) Permite que código ejecutable o applets se transmitan a través de la API para su uso dentro de la aplicación. A diferencia de SOAP, REST no está limitado a XML, sino que puede devolver XML,

JSON, YAML o cualquier otro formato, según lo que solicite el cliente. Y a diferencia de RPC, los usuarios no están obligados a conocer los nombres de procedimientos o parámetros específicos en un orden específico.

Otra diferencia importante a destacar entre el primer y segundo desarrollo del SGI, es la forma de autenticar a los usuarios y gestionar los permisos de acceso. Como se describe en 3.5 la autenticación del primer SGI se realiza a través del propio motor de base de datos, por lo que se modifica el archivo de configuración *pg_hba.conf* para autenticar mediante PAM. Al configurar PAM para autenticar con Kerberos se pueden utilizar las credenciales de acceso de este último para autenticar con el SGI.

Para mantener la sesión del usuario activa luego de la autenticación se utiliza la función `session_start()` de PHP que almacena todos los datos de sesión en el array superglobal `$_SESSION`. Ya que la autenticación requiere enviar al servidor el nombre de usuario y contraseña de conexión al Postgresql, no es posible implementar un acceso SSO en este caso.

El segundo desarrollo del SGI, al implementar una arquitectura REST sin estados, no utiliza el concepto de sesiones. En su lugar, cada petición a la API se encuentra protegida con un token de acceso JWT. JWT es un estándar abierto definido en [44] que permite transmitir información de forma compacta y segura entre las partes como un objeto JavaScript Object Notation (JSON).

En su forma compacta, un JWT se compone de tres partes separadas por puntos: *header.payload.signature*. El header especifica el tipo de token (JWT en este caso) y el algoritmo utilizado en la firma del campo signature. El payload contiene la información requerida por la aplicación y compartida entre las partes. Tanto el header como el payload se codifican en JSON y luego en base64. El campo signature se obtiene al aplicar sobre los dos campos anteriores el algoritmo definido en el header. Con ello se logra el Hashed Message Authentication Code (HMAC) que representa la firma del token.

El cliente obtiene un token JWT a través de una ruta especial de la API denominada */api/login*. Para poder ejecutar esta ruta el cliente se debe autenticar. Es aquí donde se configura el servidor web Apache con el módulo *libapache2-mod-auth-kerb* que habilita la autenticación con Kerberos. La configuración para acceder a la ruta */api/login* protegida con Kerberos se muestra a continuación:

```
Directory "/var/www/sistema/api/login/">
        # Enable Kerberos authentication using mod_auth_kerb
        AuthType Kerberos
        AuthName "Sistema2"
        KrbAuthRealm ELECTORALMISIONES.GOV.AR
        Krb5KeyTab "/etc/apache2/krb5.keytab"
        KrbMethodNegotiate on
        KrbSaveCredentials on
        KrbMethodK5Passwd on
```

```
    KrbVerifyKDC on
    KrbServiceName HTTP
    Require valid-user
</Directory>
```

La opción *KrbMethodNegotiate* permite al cliente autenticar con el servidor web mediante el protocolo SPNEGO y así lograr de manera efectiva un acceso SSO utilizando los tickets de Kerberos. La opción *KrbMethodK5Passwd* habilita la autenticación HTTP básica en caso de que el cliente no posea un ticket de Kerberos. Si bien las credenciales de acceso en una autenticación básica no están encriptadas, toda la comunicación cliente servidor se encuentra protegida debido al uso de HTTPS.

Una vez autenticado el cliente recibe el token correspondiente para ser utilizado en las siguientes peticiones a la API. El token posee un tiempo de expiración, el cual una vez vencido se deberá solicitar un nuevo token para poder seguir operando. Además del tiempo de expiración, el token utilizado por el SGI se compone de otros atributos que se pueden visualizar en el siguiente extracto de código PHP de la ruta */api/login*:

```php
$iat = time();
$exp = $iat + LIFETIME * 60;
$user = $_SERVER['REDIRECT_REMOTE_USER']

$headerArray = array(
  "alg" => "HS256",
  "typ" => "JWT"
);

$payloadArray = array(
  "iss" => "electoralmisiones.gov.ar",
  "iat" => $iat,
  "exp" => $exp,
  "user" => $user);

$header = base64_encode(json_encode($headerArray));
$payload = base64_encode(json_encode($payloadArray));
$signature = hash_hmac("SHA256", $header.".".$payload, SECRET_KEY, true);
$token = $header.".".$payload.".".$signature;
```

Otro de los atributos importantes del token es el nombre del usuario que se ha autenticado y que se almacena en la variable `$_SERVER['REDIRECT_REMOTE_USER']`.

A partir de dicho nombre se puede determinar para cada solicitud que se realiza a la API si el usuario tiene el permiso de ejecutar o no la ruta correspondiente. En la figura 4.1 se puede apreciar el diagrama de flujo general de la API y del front-end diseñados para el SGI.

En el desarrollo del gestor de usuarios se utiliza la arquitectura REST del segundo desarrollo del SGI. Para ello, el primer paso es definir los requisitos del gestor y entidades necesarias para cumplir con dichos requisitos. Esto permite

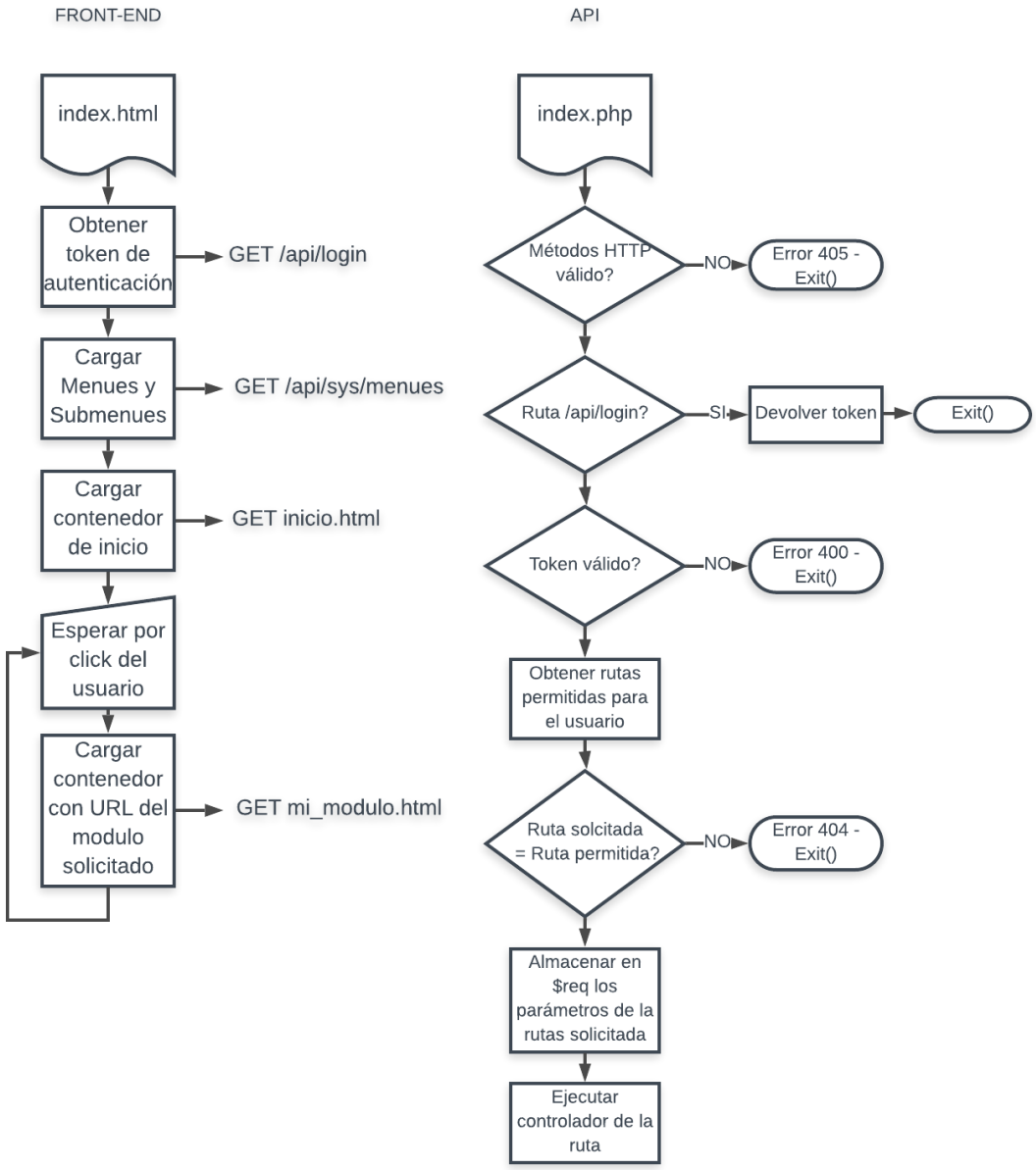

Figura 4.1: Diagrama de flujo general de la API y el front-end del SGI

elaborar los *endpoints* o rutas de la API que serán consumidas por el front-end para interactuar con los datos.

4.2. Requisitos del gestor

El gestor de autenticación debe ser un módulo de software integrado al SGI que permita al administrador del sistema gestionar de manera centralizada el alta, baja y modificación de usuarios de los servicios y aplicaciones utilizadas en el TEPM, según la infraestructura de autenticación propuesta en 3.5. A continuación se detallan los requisitos necesarios para cumplir con dicho objetivo:

- Registrar toda la información referida a los usuarios: nombre de usuario, contraseña de acceso, permisos, sector y grupos a los que pertenece, información de contacto, últimos accesos, etc.
- Permitir al administrador crear, eliminar y modificar los datos de usuario.
- Permitir al administrador listar, buscar y filtrar los datos de usuario.
- Permitir al usuario modificar y recuperar su contraseña de acceso.

Al crear un usuario el gestor debe manipular la base de datos del SGI como también las bases de datos de los servidores LDAP, KDC y Zimbra. La administración de permisos se realiza mediante *roles*. Un rol es un conjunto de menús que el usuario puede ver en el front-end y un conjunto de rutas que el usuario puede ejecutar en la API

A continuación se detallan las entidades y los atributos correspondientes para llevar adelante la implementación del gestor de usuarios:

- Usuario
    - Nombre de usuario
    - Nombre Completo
    - Matricula
    - Dirección
    - Teléfono
    - Email
    - Contraseña
    - Fecha expiración
    - Política de contraseña
    - Grupos LDAP/RADIUS
    - Roles

    - Sector
- Grupo
    - Nombre de grupo
    - Miembros
- Política de contraseña
    - Nombre
    - Tiempo de vida
    - Longitud mínima
    - Número mínimo de clases de caracteres
    - Número máximo de fallas antes de bloquear
    - Intervalo de reset del conteo de fallas
    - Tiempo de bloqueo
- Rol
    - Nombre
    - Descripción
    - Rutas incluidas
    - Menús incluidos
- Ruta
    - URL
    - Método
    - Nombre archivo
    - Nombre función
    - Descripción
- Item de Menú
    - Nombre
    - URL
    - Menú padre
    - Orden

4.3. Diseño del gestor

Respetando la arquitectura REST se debe primero diseñar las rutas de la API que permitan manipular las entidades definidas en 4.2, para luego pasar a diseñar el front-end de la aplicación.

Según las buenas prácticas de REST, las rutas deben hacer referencia a objetos, sustantivos, y no verbos. Las operaciones sobre dichos objetos son identificadas a través de los métodos del protocolo HTTP. Para obtener las clásicas operaciones de manipulación de datos Create, Read, Update and Delete (CRUD) se utilizan los métodos POST, GET, PUT y DELETE respectivamente. Así, por ejemplo la ruta */api/usuarios* mediante el método GET, devolvería un listado de todos los usuarios en formato JSON. Si se desea obtener la representación de un usuario en particular basta con llamar a la ruta anterior con el identificador de dicho usuario: */api/usuarios/*{*id_usuario*}.

Siguiendo con el mismo ejemplo, a continuación se detallan las rutas de la API necesarias para la entidad «Usuario»:

/api/usuarios Manipula los datos de usuario referidos al control de acceso.

GET */api/usuarios* : Obtiene un listado de todos los usuarios

GET */api/usuarios/*{*id_usuario*} : Obtiene el usuario *id_usuario*

POST */api/usuarios* : Crea un nuevo usuario

PUT */api/usuarios/*{*id_usuario*} : Modifica los datos del usuario *id_usuario*

DELETE */api/usuarios/*{*id_usuario*} : Elimina el usuario *id_usuario*

La representación de la entidad «Usuario» en formato JSON es la siguiente:

```
usuarios: {
        "id" : "integer",
        "username": "string",
        "nombre": "string",
        "apellido": "string",
        "matricula": "integer",
        "sexo": "string",
        "email": "string",
        "telefono": "string",
        "password": "string",
        "fecha_expiracion": "date",
        "fecha_expiracion_pass": "date",
        "cambio_pass": "boolean",
        "id_sector": "integer",
        "sector": "string",
        "externo": "boolean",
        "links": [
                {
                "href": "/api/usuarios/ldap?username=<username>",
                "rel": "ldap",
```

```
                "type" : "GET"
                },
                {
                "href": "/api/usuarios/krb?username=<username>",
                "rel": "krb",
                "type" : "GET"
                },
                {
                "href": "/api/usuarios/<id>/roles",
                "rel": "roles",
                "type" : "GET"
                }
        ]
}
```

Además de la entidad «Usuario» se definen las siguientes entidades en formato JSON necesarias para completar el diseño del gestor de usuarios:

La representación de la entidad «ldap» en formato JSON es la siguiente:

```
"ldap": {
        "uid": "integer",
        "gid": "integer",
        "cn": "string",
        "ou": "string",
        "gecos": "string",
        "home": "string",
        "loginShell": "string",
        "grupos": [
                {
                "id": "integer",
                "grupo": "string"
                },
                {
                "id": "integer",
                "grupo": "string"
                },
                ...
        ],
        "radius": [
                "string1", "string2", ...
        ]
}
```

La representación de la entidad «Krb» en formato JSON es la siguiente:

```
"krb": {
        "principal": "string",
        "expire": "date",
        "last_pwd_change": "date",
        "pwexpire": "date",
        "maxlife": "time",
        "maxrenewlife": "time",
        "mod_date": "date",
        "mod_name": "string",
        "last_success": "date",
        "last_failure": "date",
        "failures": "integer",
        "kvno": "integer",
        "policy": "string"
}
```

La representación de la entidad «Rol» en formato JSON es la siguiente:

```
"roles":        {
        "id" : "integer",
        "nombre": "string",
        "descripcion": "string",
        "links":        [
                {
                "href": "/api/sys/roles/<id>/rutas",
                "rel": "rutas",
                "type" : "GET"
                },
                {
                "href": "/api/sys/roles/<id>/menues",
                "rel": "menues",
                "type" : "GET"
                }
        ]
}
```

La representación de la entidad «Ruta» en formato JSON es la siguiente:

```
"rutas":        {
        "id" : "integer",
        "url": "string",
        "nombre_funcion": "string",
        "nombre_archivo": "string",
        "metodo": "string",
        "resumen": "string"
}
```

La representación de la entidad «Menu» en formato JSON es la siguiente:

```
"menus": {
        "id" : "integer",
        "nombre": "string",
        "url": "string",
        "target": "string",
        "submenus":[]
}
```

4.4. Implementación

Salvo las entidades «Ldap» y «Krb» que se encuentran implementadas en el servidor LDAP y KDC respectivamente, el resto de las entidades se implementan en la base de datos Postgres del SGI según el diagrama entidad-relación de la figura 4.2.

Para cada una de las rutas definidas en 4.3 se desarrollaron los controladores y modelos necesarios en el back-end. El enrutador de la API es el encargado de verificar si la ruta solicitada existe y si el usuario que la solicita posee los permisos o roles necesarios para poder ejecutarla. Luego, llama al controlador de la ruta que a su vez invoca al modelo correspondiente para obtener los datos de la

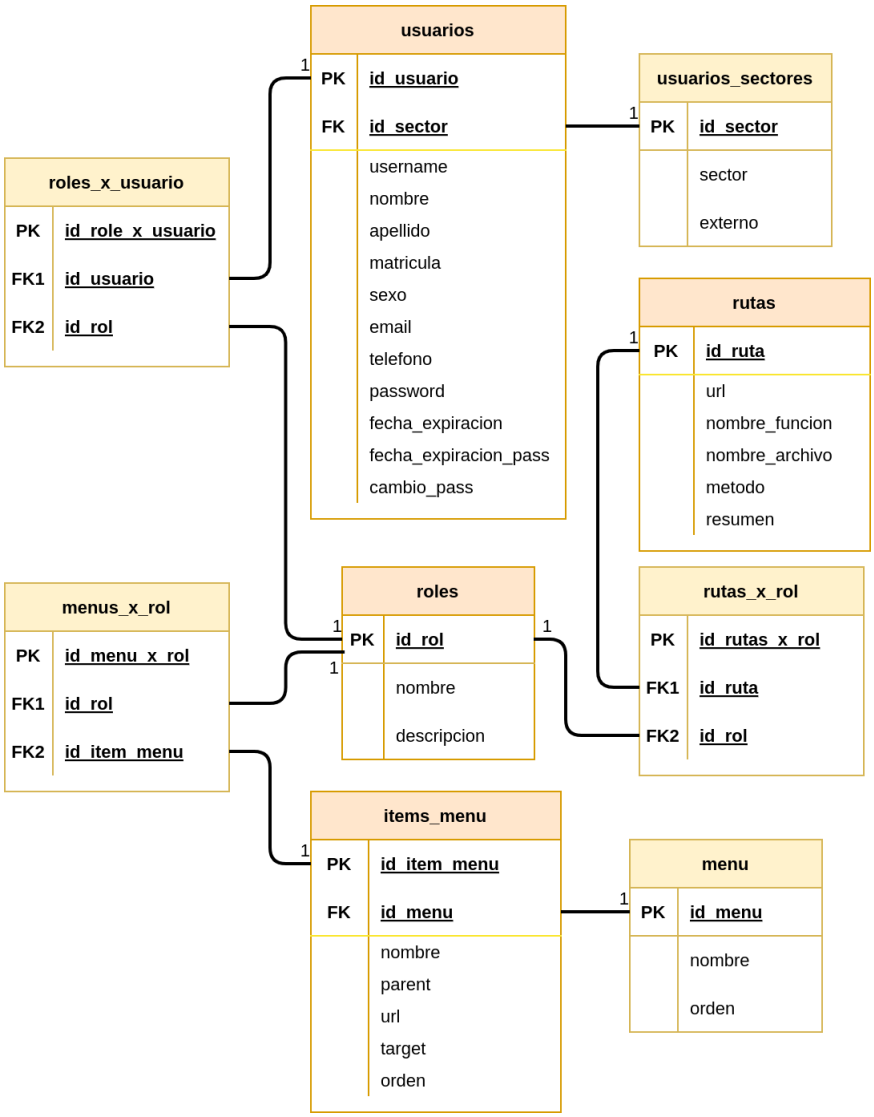

Figura 4.2: Diagrama entidad-relación para el Gestor de Usuarios

respuesta. Los controladores y modelos desarrollados para el gestor de usuarios son los siguientes:

/api/usuarios/controller/ Directorio con los archivos controladores para el Gestor de Usuarios

- UsuariosController.php: Implementa las funciones de control para las rutas de */api/usuarios/*

- UsuariosLdapController.php: Implementa las funciones de control para las rutas de */api/usuarios/ldap*
- UsuariosKrbController.php: Implementa las funciones de control para las rutas de */api/usuarios/krb*

/api/usuarios/model/ Directorio con los archivos modelo para el Gestor de Usuarios

- UsuariosModel.php: Implementa las funciones del modelo para las rutas de */api/usuarios/*
- UsuariosLdapModel.php: Implementa las funciones del modelo para las rutas de */api/usuarios/ldap*
- UsuariosKrbModel.php: Implementa las funciones del modelo para las rutas de */api/usuarios/krb*

Los siguientes controladores y modelos complementan al gestor para el manejo de permisos: roles, rutas, menús y sectores de los usuario en el SGI

- /api/roles/controller/RolesController.php
- /api/roles/model/RolesModel.php
- /api/rutas/controller/RutasController.php
- /api/rutas/model/RutasModel.php
- /api/menus/controller/MenusController.php
- /api/menus/model/MenusModel.php
- /api/sectores/controller/SectoresController.php
- /api/sectores/model/SectoresModel.php

En el front-end se implementa una pantalla o vista principal donde se listan todos los usuarios del sistema. Esta vista permite filtrar el contenido por nombre de usuario, tipo, sector y cantidad de registros a mostrar por página. Además tiene los controles necesarios para crear un nuevo usuario y ver, editar o eliminar un usuario existente. En la figura 4.3 se aprecia la vista principal del Gestor de Usuarios.

Los íconos *lupa* y *lápiz* permiten acceder a las vistas de *Ver Usuario* y *Editar Usuario* respectivamente. Haciendo click en el botón *Nuevo* se accede al formulario de creación de un nuevo usuario del sistema.

En la figura 4.4 se observa como la pantalla para ver las propiedades de un usuario se divide en un panel de Datos principales y tres paneles desplegables con

Figura 4.3: Vista principal del Gestor de Usuarios

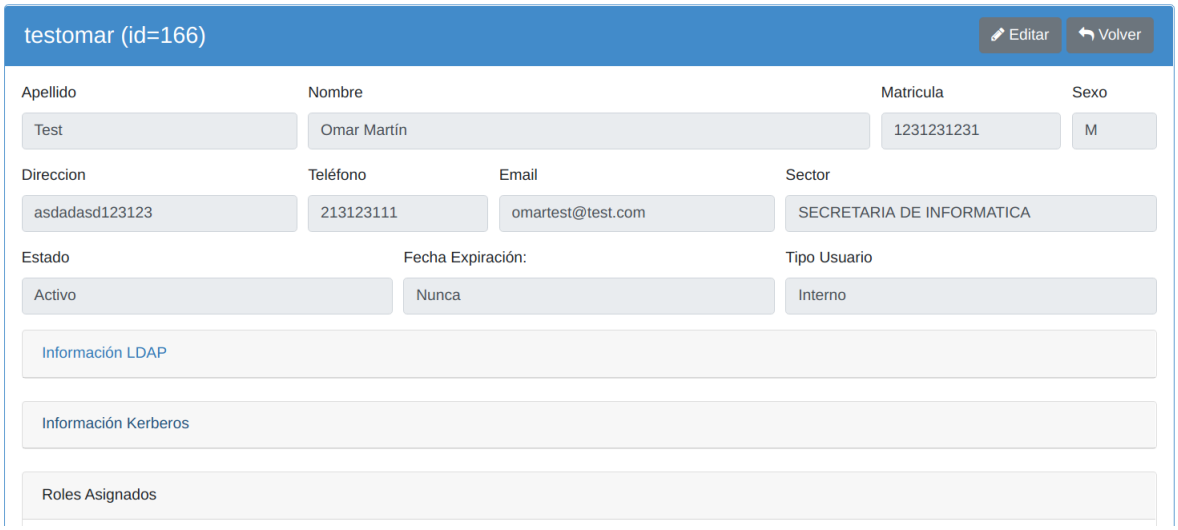

Figura 4.4: Vista Ver Usuario

Información de LDAP, Información de Kerberos y Roles Asignados. En 4.5 y 4.6 se visualizan dichos paneles.

En las figuras 4.7 y 4.8 se aprecia como se organiza el formulario para crear y editar un usuario. Ambas pantallas comparten la misma estructura de formulario donde además se visualizan los paneles de Selección de Datos LDAP y Roles, según figuras 4.9 y 4.10.

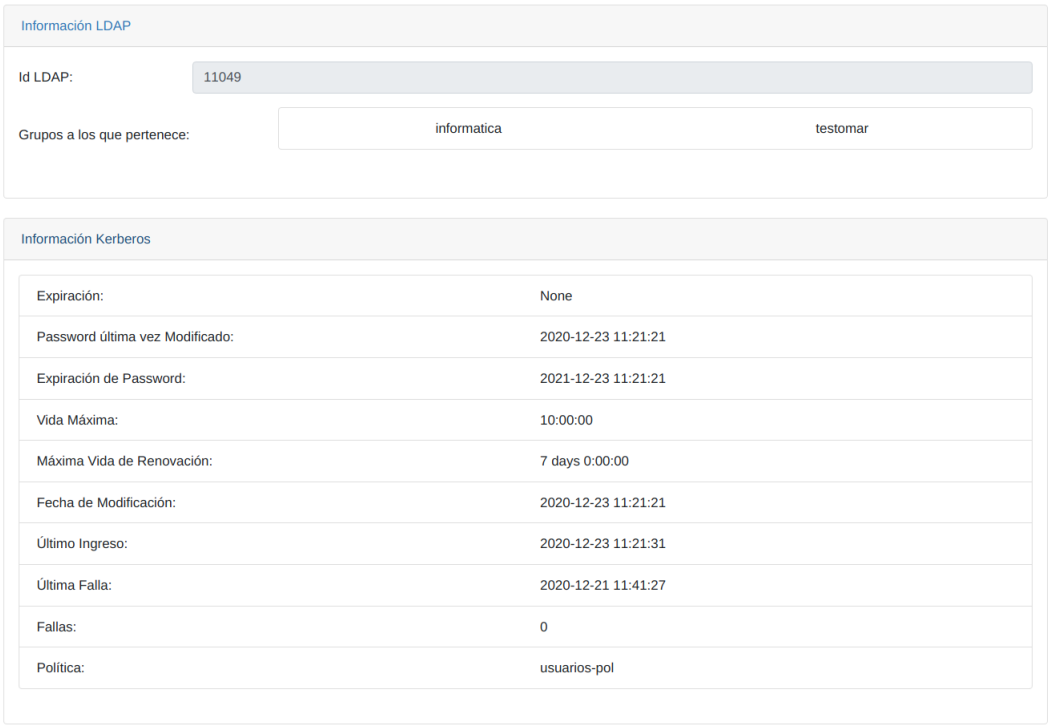

Figura 4.5: Paneles desplegables Información LDAP e Información Kerberos

Figura 4.6: Panel desplegable Roles Asignados

Los archivos donde se encuentran implementadas las vistas anteriormente citadas se organizan de la siguiente manera:

/front-end/usuarios/ Directorio con los archivos del front-end para el Gestor de

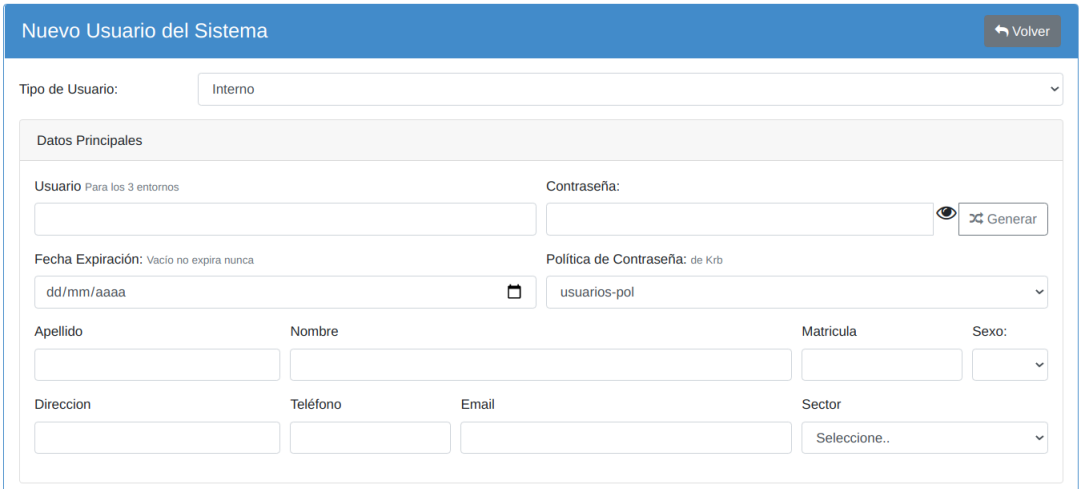

Figura 4.7: Vista Nuevo Usuario

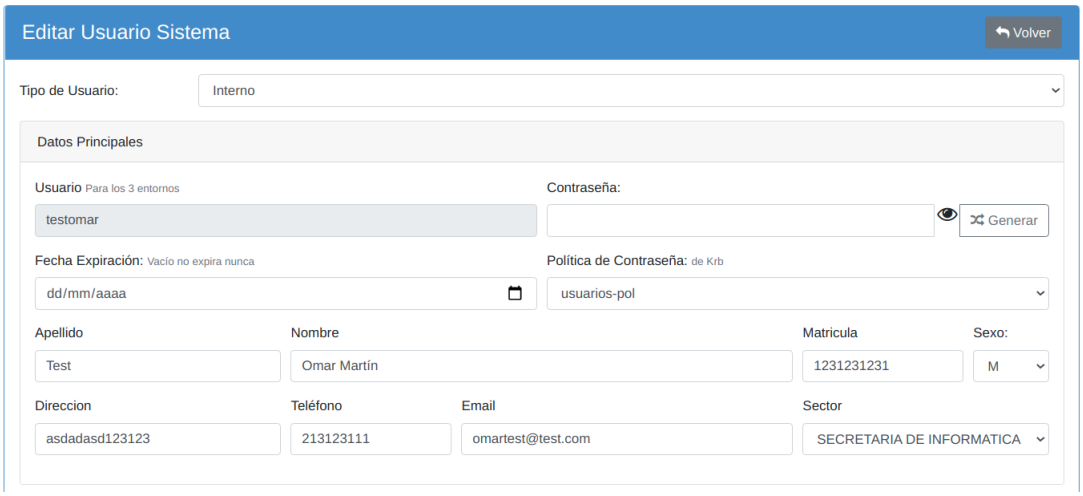

Figura 4.8: Vista Editar Usuario

Usuarios

- usuarios_index.html: Implementa la vista principal de usuarios, según 4.3.
- usuarios_get.html: Implementa la vista Ver Usuario y los paneles desplegables Información de LDAP, Información de Kerberos y Roles Asignados, según 4.4, 4.5 y 4.6.

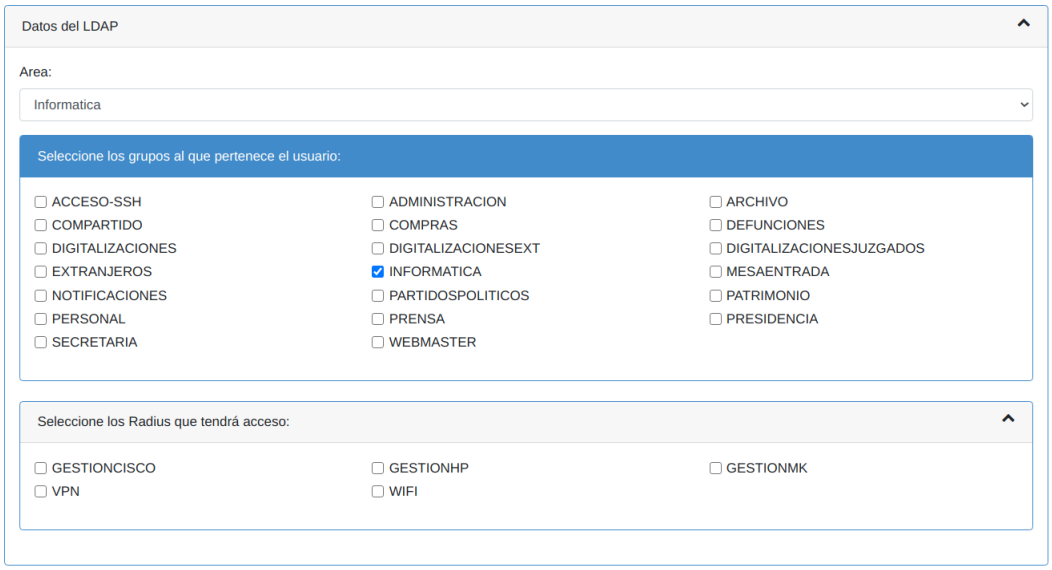

Figura 4.9: Panel de Selección Datos LDAP

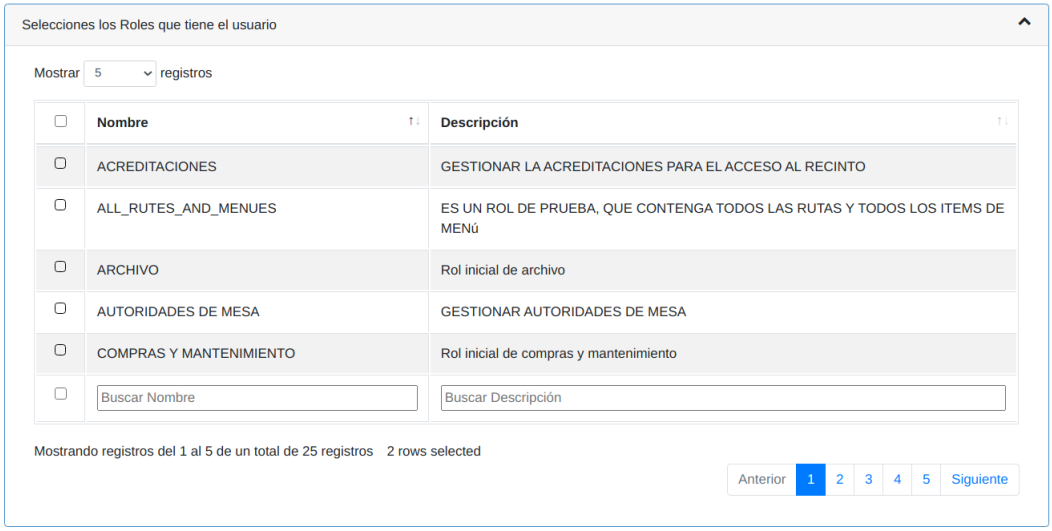

Figura 4.10: Panel de Selección Roles

- usuarios_add.html: Implementa las vistas Nuevo, Editar Usuario y los paneles de Selección de Datos LDAP y Roles, según 4.7, 4.8, 4.9 y 4.10.

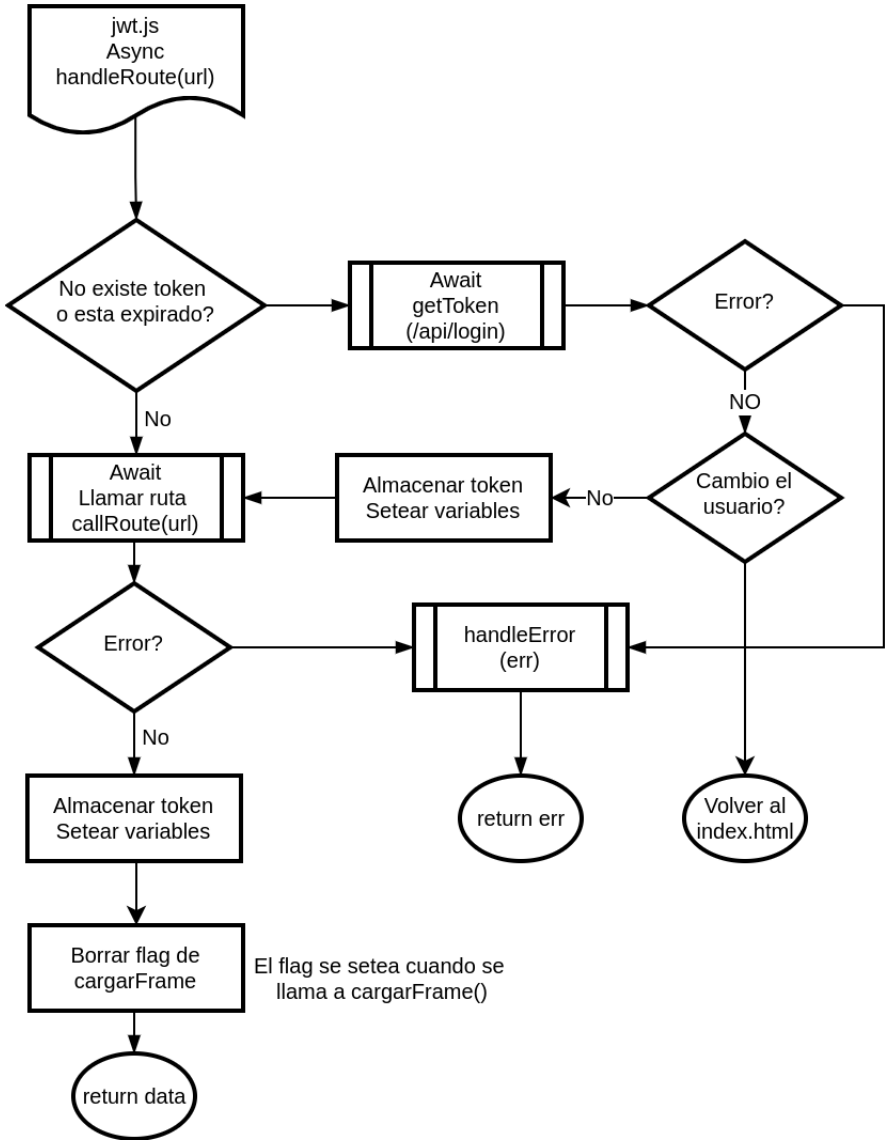

Figura 4.11: Diagrama de flujo para la función *handleRoute()*

A continuación se muestra un extracto del código HTML y javascript del archivo *usuarios_index.html*.

```
<div class="col-lg-12">
    <div class="card card-primary">
        <div class="card-header">
            <div>
                <div class="pull-right">
                    <a class="btn btn-secondary" href='#' onclick="cargarFrame
                        ('./usuarios/usuarios_add.html')">
```

```
                        <i id="btn_nuevo" class="fa fa-plus"></i>
                        Nuevo
                    </a>
                    <a class="btn btn-secondary d-none">
                        <i id="btn_imprimir" class="fa fa-print"></i>
                        Imprimir
                    </a>
                </div>
                <h3>Usuarios</h3>
            </div>
        </div>
        <div class="card-body" id="div_tabla_usuarios">
            <!-- Cargo la tabla con JQUERY -->
        </div>
    </div>
</div>

<script type="text/javascript">
    function handleUsuarios(data){
      params_tabla_usuarios = new Object();
      params_tabla_usuarios.data = data.usuarios;
      $.get('usuarios/tabla_usuarios.html', function(data){
        $('#div_tabla_usuarios').html(data);
      });
    }

    $(document).ready(function() {
      handleRoute(server + "usuarios").then(handleUsuarios);
    });
</script>
```

La función *handleRoute()* es la función encargada de llamar a la ruta de usuarios correspondiente en la API y además verificar la validez del token necesario para solicitar dicha ruta.

En *handleRoute()* se encuentra la lógica necesaria para el manejo del token y lo errores que pudieran suceder en la llamada a una ruta. Es utilizada como una función envoltorio (*wrapper*) para todas las llamadas del front-end a la API como se muestra en la figura 4.11.

5. Resultados y Conclusiones

En este capítulo se describen los resultados obtenidos luego de implementar el gestor de autenticación de usuarios desarrollado en el capítulo 4. Para finalizar, se elaboran las conclusiones del trabajo realizado y se detallan las líneas de trabajo a futuro.

5.1. Pruebas y Resultados

El sistema de gestión de calidad del TEPM establece, en el procedimiento de Desarrollo de Sistemas Informáticos, las pautas a seguir por los desarrolladores para la producción de cualquier nuevo módulo de software o modificación de los módulos existentes.

El proceso de desarrollo de sistemas informáticos culmina con la etapa de prueba y validación, mediante la cual se constata la verificación del cumplimiento de los requisitos de diseño y la inexistencia de fallas funcionales o vulnerabilidades de seguridad graves, dejando un registro de las actividades realizadas en el Registro de Prueba y Validación correspondiente.

El gestor de usuarios desarrollado en el capítulo 4 no es la excepción a esta regla y por ello, antes de poder utilizarlo en producción, debió pasar por dicha etapa. A continuación se describen algunas de las pruebas realizadas según el Registro de Prueba y Validación:

- Se prueba primero el back-end para corroborar que las respuestas obtenidas de la API se corresponden con lo visto en la sección 4.3.
- Luego, se accede al front-end del gestor de usuario para crear un usuario de prueba completando los datos solicitados en el formulario de alta según figuras 4.7, 4.9 y 4.10.
- Se prueba que el ingreso de la contraseña temporal sea de acuerdo a la política de contraseña definida.

- Se verifica que los datos cargados en el formulario de alta anterior se reflejen correctamente en las bases de datos correspondientes: Postgresql, LDAP y Kerberos.
- Se realiza el login a una computadora con sistema Linux utilizando el usuario y contraseña temporal anterior. Se verifica que se cree el ticket de kerberos correspondiente al usuario autenticado (TGT).
- El primer ingreso al nuevo SGI debe solicitar el cambio de la contraseña temporal. Este cambio debe respetar la política de contraseñas definida.
- Una vez modificada la contraseña temporal, se ingresa nuevamente al SGI probando que la autenticación se realice mediante SPNEGO y no requiera de un nuevo login (SSO). Verificar que se crea el ticket del servicio HTTP correspondiente (TGS).
- De la misma manera, se prueban los demás servicios kerberizados y configurados con SPNEGO: archivos (samba), webmail (zimbra) y SSH.
- Los servicios que no permiten SSO pero utilizan kerberos como backend de autenticación deben aceptar la contraseña configurada para el usuario de prueba: acceso al viejo SGI, gestión web del Proxmox, servicios que utilizan RADIUS, como la VPN y 802.1x
- Crear el usuario de prueba en un sistema Windows y repetir las pruebas de autenticación anteriores verificando el correcto funcionamiento de Kerberos sobre esta plataforma.
- Se prueba la visualización y edición del usuario según lo visto en las figuras 4.4 y 4.8. Probar que la configuración de una nueva contraseña obliga al usuario a cambiar nuevamente la contraseña temporal.

Luego de varias etapas de prueba y corrección de errores, el gestor de usuarios fue puesto en producción, previa configuración de la infraestructura vista en la sección 3.5. Además, se inició el proceso de migración de los usuarios existentes y la configuración de las computadoras para autenticar con la nueva infraestructura centralizada.

Los resultados obtenidos luego de la implementación del gestor de usuarios fueron muy satisfactorios. En primer lugar, se logró reducir significativamente la cantidad de trabajo necesario para gestionar los usuarios y sus cuentas de acceso. Además, la gestión centralizada permitió disminuir la posibilidad de errores en la configuración y mantenimiento de dichas cuentas.

A modo de ejemplo, se cita la simple tarea de actualizar las contraseñas de un usuario debido a la expiración anual establecida en la política de contraseñas.

Previo a la implementación del gestor centralizado de usuarios, dicha tarea se podía programar para obligar al usuario a realizar el cambio luego de la expiración, pero sólo en determinadas aplicaciones que lo soportaban. Para las demás aplicaciones el cambio debía hacerse de manera manual, involucrando al personal de IT en el proceso.

Con la implementación del gestor centralizado de usuarios esta tarea se vuelve trivial, ya que la misma política se pude aplicar de manera homogénea a todas las aplicaciones y la tarea de actualización la realiza el propio usuario desde un solo lugar: el SGI. El personal de IT se debe ocupar ahora sólo de capacitar correctamente al usuario en el uso de esta nueva funcionalidad del SGI.

Para cuantificar el costo operativo del sector de IT antes y después de la implementación del gestor, se realizaron pruebas para medir el tiempo incurrido en dos de las tareas más demandadas: la creación de cuentas de usuario y el reset de contraseñas. En la tabla 5.1 se muestran los valores obtenidos de dichas pruebas y en la tabla 5.2 se presentan los valores medios calculados.

Para la medición de los tiempos en las mencionadas tareas se consideró que una cuenta de usuario estándar incluye: una cuenta para el SGI, una casilla de email, acceso a los archivos compartidos y un usuario para la computadora de escritorio. En el caso del gestor centralizado, la cuenta del usuario creada queda disponible para ser utilizada en cualquier computadora del dominio del TEPM.

Otro de los aspectos que permitió evaluar el costo operativo del sector de IT fue el sistema de tickets del TEPM. Dicho sistema se utiliza para solicitar ayuda al área de soporte técnico en cuestiones tales como: fallas de hardware, fallas de conectividad, fallas de impresión, falta de tinta o tóner, errores de acceso y errores en la funcionalidad de algún programa o aplicación.

Luego de una revisión e identificación de los tickets generados por los usuarios durante los períodos 2019, 2020 y 2021, se obtuvieron todos los tickets relacionados a *Errores de acceso con usuario y clave*. Si se tiene en cuenta que el sistema unificado de autenticación se comenzó a utilizar a principios de 2021, se puede determinar la cantidad de solicitudes de este tipo realizadas antes y después de la implementación.

Debido a que en los años 2019 y 2021 hubieron elecciones provinciales, se consideraron dichos períodos para la comparación. Como se observa en la tabla 5.2, el porcentaje de tickets relacionados a errores de acceso disminuyó un 22,2 % luego de la implementación del sistema de autenticación SSO, en relación al total de tickets realizados al sector de soporte técnico.

Es importante destacar que la cantidad de solicitudes de servicio hechas a través del sistema de tickets es sólo una parte del total de solicitudes hechas al sector de IT, ya que el mayor porcentaje de las solicitudes son realizadas vía

	Antes del Gestor		**Después del Gestor**	
Test #	Tarea 1	Tarea 2	Tarea 1	Tarea 2
1	47 min	28 min	12 min	5 min
2	42 min	26 min	14 min	4 min
3	45 min	29 min	11 min	4 min
4	45 min	28 min	13 min	5 min
5	50 min	31 min	12 min	4 min
6	48 min	27 min	15 min	5 min
7	44 min	30 min	14 min	6 min
8	42 min	28 min	11 min	4 min
9	46 min	28 min	12 min	5 min
10	47 min	27 min	11 min	4 min

Tarea 1: Creación de una cuenta de usuario estándar
Tarea 2: Reset de contraseña de un usuario

Tabla 5.1: Tiempos obtenidos en las tareas 1 y 2, antes y después del gestor centralizado de usuarios

telefónica y, lamentablemente, no quedaron registros de las mismas.

Según un estudio realizado en [72], donde se estima que un usuario típico realiza 16 llamadas a la mesa de ayuda por año, podemos calcular que la cantidad de solicitudes registradas en el sistema de tickets representa aproximadamente el 16 % del total de solicitudes hechas al sector de IT.

Desde el punto de vista de los usuarios, el hecho de tener una sola cuenta para todos los sistemas y un acceso SSO, se ve incrementada su productividad a la hora de trabajar con los diferentes servicios ofrecidos por el TEPM, ya que no requieren lidiar con constantes solicitudes de login.

Esto se refleja en la encuesta realizada a los usuarios (anexo B), donde el 71,43 % respondió que su desempeño laboral mejoró luego de un año de uso del sistema SSO, como se observa en la figura 5.1.

Previo al gestor de usuarios, la movilidad era prácticamente nula ya que la cuenta de acceso de un usuario existía únicamente en la computadora ubicada en su escritorio de trabajo. Hoy, debido a la gestión centralizada de las cuentas, el usuario puede utilizar cualquier computadora configurada para autenticar en el dominio del TEPM, trasladando todo su perfil de acceso de manera automática.

	Antes del Gestor	**Después del Gestor**
Usuarios que realizan 2 a 5 logins por día:	$65,71\,\%$	$28,57\,\%$
Usuarios que realizan 1 login por día:	$20\,\%$	$65,71\,\%$
Usuarios con problemas de acceso durante el login:	$57,14\,\%$	$14,29\,\%$
Usuarios que solicitaron reset de contraseña alguna vez:	$85,71\,\%$	$37,14\,\%$
Cantidad promedio de contraseñas por usuario:	3	1
Nivel de satisfacción del usuario:	Medio Bajo	Medio Alto
Nivel de confianza del usuario:	Medio Bajo	Alto
Movilidad del usuario:	Nula	En todo el dominio
Tiempo medio de creación de cuenta de usuario:	$45,6\ min$	$12,5\ min$
Tiempo medio de reset de contraseña:	$28,2\ min$	$4,6\ min$
Tickets relacionados a errores de acceso:	$31,8\,\%$	$9,6\,\%$

Tabla 5.2: Resultados obtenidos luego de la implementación del Gestor de Usuarios

En la tabla 5.2 se resumen todos los resultados obtenidos luego de implementar el gestor centralizado de usuarios, comparados con los valores sostenidos antes de la implementación.

Dichos resultados se sustentan con la encuesta realizada a los usuarios luego de un año de uso del sistema SSO, los registros del sistema de tickets referidos a errores de acceso y las mediciones obtenidas de las pruebas de creación de cuentas de usuario y reset de contraseñas.

Antes del gestor centralizado de usuarios, la cantidad de logins realizados dependía de la cantidad de sistemas y aplicaciones necesarias para llevar adelante las tareas diarias.

Como se observa en la tabla 5.2, el usuario promedio debía realizar entre 2 a 5 logins diarios. Esta cantidad se veía multiplicada por la cantidad de veces que el usuario bloqueaba la pantalla o se bloqueaba de manera automática por cuestiones de seguridad referidas a las políticas de usuario desatendido.

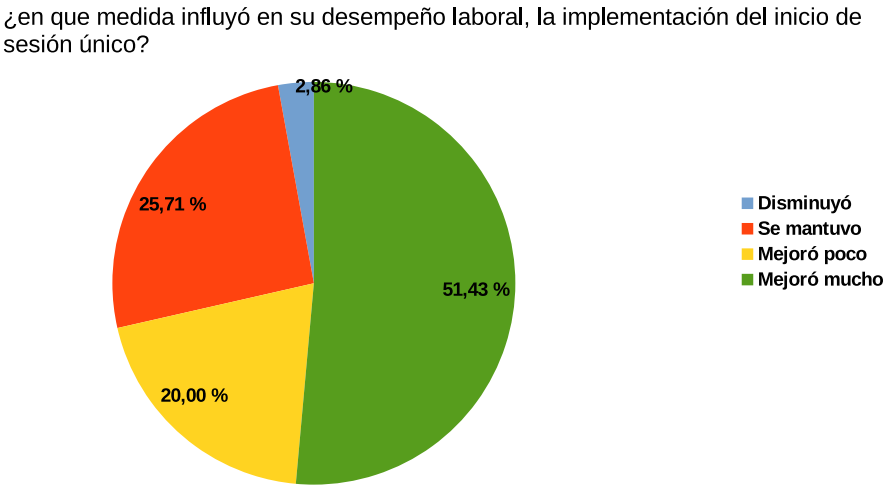

Figura 5.1: Influencia del sistema SSO en el desempeño laboral luego de su implementación

Luego de la implementación del gestor de usuarios, esta cantidad se vio reducida a un sólo login debido a la integración de las aplicaciones de uso diario al sistema de SSO.

Para el cálculo del nivel de satisfacción y confianza de la tabla 5.2 se asignaron valores numéricos (1, 2 y 3) a las variables cuantitativas utilizadas en la encuesta realizada a los usuarios. De esta manera se pudo calcular el promedio y volver a asignar un valor cualitativo al resultado obtenido según la siguiente escala:

- Entre 1 y 1,4 el valor es *Bajo*
- Entre 1,5 y 1,9 el valor es *Medio Bajo*
- Entre 2 y 2,4 el valor es *Medio Alto*
- Entre 2,5 y 3 el valor es *Alto*

Si bien el costo de despliegue del gestor de usuarios fue alto, la reducción de costos relacionados a la carga operativa del sector de IT y el incremento sustancial de la seguridad y productividad del usuario, compensan ampliamente dicho costo.

Otro aspecto importante que permitió implementar la gestión centralizada de usuarios, es la aplicación de una política de acceso estandarizada y homogénea a toda la infraestructura del TEPM. A modo de ejemplo, en la tabla 5.3 se presentan posibles perfiles de acceso que se pueden implementar en una organización en función de los sistemas utilizados.

Mediante el gestor centralizado de usuarios se pudieron aplicar perfiles de acceso predefinidos por tipo de usuario siguiendo una matriz similar a la presentada en la tabla 5.3. En el caso de permisos especiales se debió solicitar por separado, previa autorización del responsable del sector y aprobación del administrador de IT.

	1 - Estándar	**2 - Soporte Técnico**	**3 - Desarrollador**	**4 - Administrador**
Sistema de Gestión Interno (SGI):	Rol por sector	Rol por sector (informática)	Rol por sector (informática) + Rol Desarrollador	Rol Administrador (Gestión de Usuarios)
Permisos PC (Grupos LDAP):	Archivos de Usuario, Sector y Compartido	Ídem perfil 1 + Acceso SSH, Sudo y USB	Ídem perfil 2 + Repositorios de Software	Ídem perfil 3 + Radius (Gestión de equipos de red y acceso VPN)
Servicio de Email (Zimbra):	Casilla estándar (2GB)	Casilla estándar (2GB)	Casilla estándar (5GB)	Casilla estándar (10GB) + Cuenta Administrador
Servidor de Contenidos WWW:	Invitado (Rol Publisher para Sector de Prensa)	Invitado	Manager	Administrator
Base de Datos (Postgresql):	N/A	N/A	Rol Develop (DB Test)	Rol DB-Admin

Tabla 5.3: Matriz de Perfiles de Acceso por tipo de Usuario

También, durante el desarrollo del trabajo se presentó una manera novedosa de integrar Kerberos como sistema SSO a aplicaciones web del tipo API REST, tal como se describe en la sección 4.1.

La forma de utilizar el módulo de Apache *libapache2-mod-auth-kerb* con la opción *KrbMethodNegotiate* para proteger la ruta */api/login*, permite al cliente autenticar con el servidor web mediante el protocolo SPNEGO y obtener el token JWT correspondiente para ser utilizado en las demás rutas de la API. Así se logra de manera efectiva un acceso SSO a las aplicaciones del tipo API REST mediante tickets de Kerberos.

Cabe aclarar, que dicho mecanismo de acceso SSO no solo aplica a un servidor web con Apache, sino que puede ser implementado en otros servidores web que soporten autenticación SPNEGO con Kerberos, como es el caso del servidor Nginx.

Para finalizar, se destaca que si bien el gestor de usuarios mencionado con anterioridad fue desarrollado para un escenario específico, quitando alguna particularidades referidas a los permisos y roles del SGI, puede ser adaptado fácilmente para ser utilizado en otras organizaciones, ya que la infraestructura de integración propuesta en la sección 3.5 hace uso de componentes y servicios estándares de la industria.

5.2. Conclusiones

En este trabajo final de maestría se presentó una solución para la gestión de autenticación centralizada de usuarios en entornos de red heterogéneos como el existente en el TEPM, caso de estudio del mencionado trabajo.

Además, como parte de dicha solución, se presentó una manera novedosa de integrar Kerberos como sistema SSO a aplicaciones web del tipo API REST.

En efecto, la implementación del gestor de autenticación de usuarios desarrollado en el capítulo 4, propició los siguientes beneficios:

- Mayor control sobre las identidades y permisos de acceso a los recursos digitales, logrando una reducción de los riesgos internos y externos relacionados a una fuga de datos.
- Asegurar que los permisos y privilegios de los usuarios son otorgados de acuerdo una política de acceso y que se aplica un servicio de AAA apropiado en toda la infraestructura tecnológica de la organización.
- Aumento en la eficiencia del personal de IT y reducción de los costos relacionados a la gestión de las cuentas de usuario.
- Facilidad en la implementación o cambios en las políticas de seguridad de la organización, logrando una aplicación uniforme a todos los dispositivos y plataformas tecnológicas.
- Incremento en la usabilidad de los sistemas y satisfacción de los usuarios debido a la implementación de sistemas SSO, como se describe en la sección 2.4.

Para alcanzar los objetivos planteados, en primer lugar se debió estudiar e investigar los mecanismos y protocolos de autenticación de usuarios existentes (estado del arte), tal como lo expuesto en el capítulo 2.

El marco teórico elaborado en dicho capítulo aportó los conocimientos necesarios para analizar y comprender los mecanismos de autenticación utilizados en el caso de estudio.

Mediante la comparación de diferentes sistemas de IAM e identificación de componentes y herramientas necesarias para su implementación, se logró presentar en el capítulo 3, una solución para integrar al TEPM un gestor de usuarios centralizados que abarque la mayoría de los sistemas involucrados en sus operaciones diarias.

Para poner a prueba la solución propuesta en el mencionado capítulo, se debió desarrollar e implementar un módulo de software que permitiera gestionar las

cuentas de usuario de manera centralizada, interactuando con los componentes principales de la arquitectura de autenticación.

Una descripción detallada de las características y etapas de desarrollo de dicho módulo de software se expusieron en el capítulo 4.

Finalmente, las pruebas y resultados obtenidos en la sección 5.1 confirmaron el correcto funcionamiento del gestor centralizado de usuarios y el aporte como herramienta de gestión a otras organizaciones de similares características al TEPM; cuyos requerimientos impliquen un sistema de IAM adaptable a sus necesidades específicas y con soporte para integrar aplicaciones del tipo API REST.

5.3. Líneas de trabajo a futuro

El gestor de autenticación de usuarios y la infraestructura propuesta en el presente trabajo constituyen las bases para desarrollar un sistema de IAM con mayores funcionalidades y características. Algunas de las características que aportarían valor y resulta conveniente trabajar a futuro, se detallan a continuación:

- Incorporación de un mecanismo de 2FA para mitigar o reducir el riesgo que implica una única credencial para un sistema SSO.
- Implementación de un módulo de auditoría, capaz de recolectar información referida a los procesos de autenticación y acceso de los usuarios a los sistemas de IT.
- Mejorar la integración con los servicios que requieran su propia base de datos de usuario (por ejemplo Zimbra). En este sentido, sería conveniente automatizar los procesos de gestión de usuarios utilizando las herramientas y APIs proporcionadas por dichos servicios.
- Investigar la manera de incorporar a la gestión de usuarios, las reglas de control de acceso a los hosts y políticas de *sudo*.
- Debido a la criticidad del servicio de autenticación es conveniente estudiar la manera de implementar una infraestructura de alta disponibilidad capaz de soportar la caída de al menos uno de los elementos críticos que componen la arquitectura de autenticación propuesta y que el servicio no se resienta.
- Trabajar sobre un módulo PAM que permita la autenticación 802.1x previo al proceso de login del sistema operativo. Actualmente no existe dicha funcionalidad en los sistemas Linux, lo que limita la posibilidad de integrar la autenticación de red al resto de los servicios SSO.

- Teniendo en cuenta que la infraestructura de autenticación propuesta integra un servidor Kerberos, se puede investigar y analizar la viabilidad de implementación del protocolo de autenticación EAP presentado en [73], denominado EAP-Kerberos, para traspasos en redes inalámbricas de baja latencia. Dicho protocolo basado en Kerberos, podría ser una solución superadora frente a otras variantes del protocolo EAP.

Además de las cuestiones técnicas mencionadas con anterioridad, sería conveniente estudiar la aplicación del sistema de autenticación propuesto en otros organismos de similares características al TEPM.

Siguiendo con el análisis realizado en el presente trabajo, resulta interesante diseñar una guía o framework para implementar un sistema de autenticación unificado en diferentes entornos o escenarios según: cantidad de sistemas Linux/-Windows, tipos de servicios y mecanismos de autenticación presentes, cantidad de dispositivos de red, cantidad de usuarios, etc. Dicho framework podría, por ejemplo, definir que sistema de autenticación y método de integración utilizar según las características de cada caso.

Otra línea de trabajo se presenta en la evaluación del desempeño y seguridad del método de autenticación SSO presentado para aplicaciones API REST. Se podría realizar una comparación contra otros métodos de autenticación web similares, tales como el presentado en [74], donde se describen las bondades de un algoritmo de alta seguridad basado en Kerberos para servicios de almacenamiento en la nube tipo REST.

A. Implementación de la Infraestructura

A continuación se detallan los pasos para la implementación de la infraestructura de red y servicios propuesta en la sección 3.5 sobre un servidor con *Ubuntu Server* como sistema operativo.

1. Instalar y configurar los siguientes paquetes para la implementación del servidor KDC:

   ```
   kdc:~$ sudo apt install krb5-kdc krb5-admin-server
   ```

   Durante la instalación se le solicita que defina el *realm*, el nombre del servidor KDC y el servidor de administración para Kerberos. El archivo de configuración */etc/krb5.conf* debe quedar configurado de la siguiente manera:

   ```
   [libdefaults]
   default_realm = ELECTORALMISIONES.GOV.AR

   [realms]
   ELECTORALMISIONES.GOV.AR = {
   kdc = kdc.electoralmisiones.gov.ar
   admin_server = kdc.electoralmisiones.gov.ar
   default_domain = electoralmisiones.gov.ar
   }

   [domain_realm]
   .electoralmisiones.gov.ar = ELECTORALMISIONES.GOV.AR
   electoralmisiones.gov.ar = ELECTORALMISIONES.GOV.AR
   ```

   Luego de la instalación, se crea la base de datos para el *realm* definido y un usuario administrador del KDC:

```
kdc:~$ sudo krb5_newrealm
kdc:~$ sudo kadmin.local
Auth. as principal root/admin@ELECTORALMISIONES.GOV.AR with password.
kadmin.local: addprinc soporte/admin
Enter password for principal "soporte/admin@ELECTORALMISIONES.GOV.AR":
Re-enter password for principal "soporte/admin@ELECTORALMISIONES.GOV.AR":
Principal "soporte/admin@ELECTORALMISIONES.GOV.AR" created.
kadmin.local: quit
```

Agregar la siguiente línea al archivo */etc/krb5kdc/kadm5.acl* para que el usuario creado tenga los permisos de administrador:

```
soporte/admin@ELECTORALMISIONES.GOV.AR *
```

Para finalizar la configuración del KDC, se pueden agegar al *realm* los usuarios (*principals*) necesarios con la utilidad *kadmin*. Para facilitar la tarea del administrador del sistema, en lugar de la utilidad *kadmin*, se utiliza el gestor de autenticación de usuarios desarrollado en el capítulo 4.

Para los servicios también es necesario agregar un *principal* de servicio y un archivo *keytab* para autenticar con el KDC sin necesidad de una contraseña. Por ejemplo, para el servicio de acceso remoto a un *host* (SSH, telnet, rlogin, etc.) se agrega el siguiente *principal* y su *keytab* correspondiente:

```
pc1:~# kinit soporte
Password for soporte@ELECTORALMISIONES.GOV.AR:
pc1:~$ kadmin soporte
Auth. as principal soporte/admin@ELECTORALMISIONES.GOV.AR with password.
Password for soporte/admin@ELECTORALMISIONES.GOV.AR:
kadmin: addprinc -randkey host/pc1.electoralmisiones.gov.ar
Princ. "host/pc1.electoralmisiones.gov.ar@ELECTORALMISIONES.GOV.AR" created.
kadmin: ktadd -k /etc/krb5.keytab  host/pc1.electoralmisiones.gov.ar
Entry for principal host/pc1.electoralmisiones.gov.ar with kvno 3, encryption
type aes256-cts-hmac-sha1-96 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: quit
```

El *principal* de servicio *host* se debe crear para todas las PCs y servidores que necesiten acceso remoto. Además, se pueden crear los siguientes *principals* para el resto de los servicios que se desean kerberizar:

- HTTP/web-server.electoralmisiones.gov.ar
- cifs/samba-server.electoralmisiones.gov.ar
- smtp/mail-server.electoralmisiones.gov.ar
- imap/mail-server.electoralmisiones.gov.ar
- nfs/nfs-server.electoralmisiones.gov.ar

2. Instalar y configurar los siguientes paquetes para la implementación del servidor LDAP:

```
kdc:~$ sudo apt install slapd ldap-utils
```

Durante la instalación se define la credencial de acceso del usario administrador del Directory Information Tree (DIT) creado (*rootDN*). El sufijo o DN base del DIT se configura según el dominio al que pertence el servidor, en este caso es `dn: dc=electoralmisiones,dc=gov,dc=ar`. Se puede cambiar el sufijo luego de la instalación, y antes de almacenar cualquier tipo de datos, con el siguiente comando:

```
kdc:~$ sudo dpkg-reconfigure slapd
```

Resta ahora crear las unidades organizativas (OU), grupos y usuarios del sistema. Una forma de agregar contenido a la base de datos del LDAP es a través de archivos LDIF. Por ejemplo, el siguiente archivo *add_content.ldif* define las OU «Usuarios» y «Grupos», el grupo «informatica» dentro de la OU «Grupos», y el usuario «user1» perteneciente al grupo «informatica» y dentro de la OU «Usuarios»:

```
dn: ou=Usuarios,dc=electoralmisiones,dc=gov,dc=ar
objectClass: organizationalUnit
ou: Usuarios

dn: ou=Grupos,dc=electoralmisiones,dc=gov,dc=ar
objectClass: organizationalUnit
ou: Grupos

dn: cn=informatica,ou=Grupos,dc=electoralmisiones,dc=gov,dc=ar
objectClass: posixGroup
cn: informatica
gidNumber: 5000

dn: uid=usuario1,ou=Usuarios,dc=electoralmisiones,dc=gov,dc=ar
uid: usuario1
cn: usuario1
objectClass: account
objectClass: posixAccount
objectClass: top
loginShell: /bin/bash
uidNumber: 10000
gidNumber: 5000
homeDirectory: /home/usuario1
gecos: Usuario1
```

Luego, agregamos dichas entradas al LDAP con el comando:

```
kdc:~$ sudo ldapadd -x -D cn=admin,dc=electoralmisiones,dc=gov,dc=ar \
       -W -f add_content.ldif
```

Para facilitar la tarea del administrador del sistema, en lugar de los archivos LDIF, se utiliza el gestor de autenticación de usuarios que se desarrolla con mayor detalle en el capítulo 4.

3. Instalar y configurar los siguientes paquetes para la implementación del servidor RADIUS:

```
pc1:~$ sudo apt install freeradius freeradius-krb5
```

4. Instalar y configurar el suplicante de host para máquinas con Linux:

```
pc1:~$ sudo apt install sssd krb5-user
```

El SSSD durante su instalación instala las librerías y configura las opciones necesarias para los módulos del PAM y NSS. Luego de la instalación se configura el archivo */etc/sssd/sssd.conf* con las siguientes opciones:

```
[sssd]
config_file_version = 2
services = nss,pam
domains = electoralmisiones.gov.ar

[nss]
filter_users = root
filter_groups = root

[pam]

[domain/electoralmisiones.gov.ar]
auth_provider = krb5
krb5_server = kdc.electoralmisiones.gov.ar
krb5_realm = ELECTORALMISIONES.GOV.AR
cache_credentials = true

access_provider = simple
chpass_provider = krb5

id_provider = ldap
ldap_uri = ldap://kdc.electoralmisiones.gov.ar
ldap_search_base = dc=electoralmisiones,dc=gov,dc=ar
sudo_provider = none
```

Por último, es necesario configurar el módulo *pam_mkhomedir.so* para crear automáticamente el directorio *home* del usuario cuando se autentica a la PC. Para ello se agrega la siguiente sentencia en el archivo */etc/pam.d/common-account*:

```
session required pam_mkhomedir.so skel=/etc/skel/ umask=0022
```

5. Configurar el suplicante de host para máquinas con Windows:

6. Configurar el navegador web para un acceso SSO con SPNEGO a los servicios web compatibles: La mayoría de los navegadores web importantes como Mozilla Firefox, Internet Explorer, Chrome y Safari soportan la autenticación con SPNEGO. Cada uno tiene su forma de configurarlo. A continuación se muestra, a modo de ejemplo, los pasos para la configuración de SPNEGO en Mozilla Firefox:

   - Navegar a la Uniform Resource Locator (URL) *about:config*.
   - Hacer click para aceptar el mensaje de advertencia sobre las consecencias de modificar el comportamiento del navegador.
   - Definir las siguientes opciones de configuración:

     ```
     network.negotiate-auth.delegation-uris: *electoralmisiones.gov.ar
     network.negotiate-auth.trusted-uris: *electoralmisiones.gov.ar
     ```

7. Configurar los servicios para autenticar con Kerberos (se supone que ya estan instalados y funcionando en la infraestructura del TEPM pero con los métodos de autenticación descriptos en 3.3.1):

   Zimbra Para habilitar la autenticación con Kerberos en el servidor de email se ejecutan los siguientes comandos de provisionamiento del Zimbra:

   ```
   zimbra@mail:~$ zmprov md electoralmisiones.gov.ar \
   zimbraAuthMech kerberos5
   zimbra@mail:~$ zmprov md electoralmisiones.gov.ar \
   zimbraAuthKerberos5Realm ELECTORALMISIONES.GOV.AR
   ```

   El acceso a las cuentas de correo electrónico se realiza principalmente a través del cliente web de Zimbra. Para permitir un acceso SSO se habilita la opción de autenticación con SPNEGO de Zimbra:

   ```
   zimbra@mail:~$ zmprov mcf zimbraSpnegoAuthEnabled TRUE
   zimbra@mail:~$ zmprov mcf zimbraSpnegoAuthErrorURL '?ignoreLoginURL=1'
   zimbra@mail:~$ zmprov mcf zimbraSpnegoAuthRealm ELECTORALMISIONES.GOV.AR
   ```

   Como se describe en 1, es necesario crear un archivo *keytab* para el servicio *HTTP/mail.electoralmisiones.gov.ar* y copiarlo en */opt/zimbra/data/mailboxd/spnego/jetty.keytab* del servidor Zimbra.

   Cuando la autenticación con SPNEGO falla, el usuario es redirigido a al página de login por defecto de Zimbra. Para permitir que el usuario se autentique de todas formas con las credenciales de Kerberos, es necesario configurar el archivo */opt/zimbra/jetty/etc/krb5.ini.in* (no se utiliza el archivo por defecto */etc/krb5.conf* cuando SPNEGO se encuentra habilitado en Zimbra):

```
[realms]
%%zimbraSpnegoAuthRealm%% = {
  kdc = kdc.electoralmisiones.gov.ar
  admin_server = kdc.electoralmisiones.gov.ar
  default_domain = %%zimbraSpnegoAuthRealm%%
}
```

Reiniciar el servicio para que los cambios tengan efecto:

```
zimbra@mail:~$ zmcontrol restart
```

Samba Para autenticar los usuarios del servidor de archivos con Kerberos se modifica el archivo de configuración de Samba */etc/samba/smb.conf*:

```
security = ADS
realm = ELECTORALMISIONES.GOV.AR
encrypt passwords = yes
kerberos method = secrets and keytab
```

Como se describe en 1, es necesario crear un archivo *keytab* para el servicio *cifs/samba.electoralmisiones.gov.ar* y copiarlo en */etc/krb5.keytab* Luego se reinicia el servicio para que los cambios tengan efecto:

```
soporte@samba:~$ sudo service smbd restart
```

SGI

8. Configurar la infraestructura para autenticar con Kerberos (se supone que ya se encuentra instalada y funcionando en el TEPM pero con los métodos de autenticación descriptos en 3.3.2):

   Para el acceso por SSH a una máquina de usuario o servidor se habilita la autenticación GSS-API en el archivo de configuración */etc/ssh/sshd_config* del servidor *openssh-server* y luego se reinicia el servicio:

   ```
   GSSAPIAuthentication yes
   ```

   ```
   pc1:~$ sudo service sshd restart
   ```

   Es necesario además tener creado el *principal* y *keytab* correspondiente según lo detallado en 1.

B. Encuesta sobre los sistemas de autenticación

Para poder evaluar los resultados de la implementación del sistema de inicio de sesión único y el gestor centralizado de usuarios, se realizó una encuesta a los usuarios del TEPM luego de un año de uso del nuevo sistema de autenticación.

En dicha encuesta se les pidió que respondan a diferentes cuestiones relacionadas al antes y después de la implementación del mencionado sistema. En total respondió un 83 % de los usuarios activos del TEPM y que estuvieron en el proceso de cambio del sistema de autenticación.

En las siguientes figuras se muestra el resumen de las respuestas obtenidas en la encuesta. En el capítulo 5 se realiza la evaluación correspondiente para obtener los resultados y conclusiones finales del trabajo.

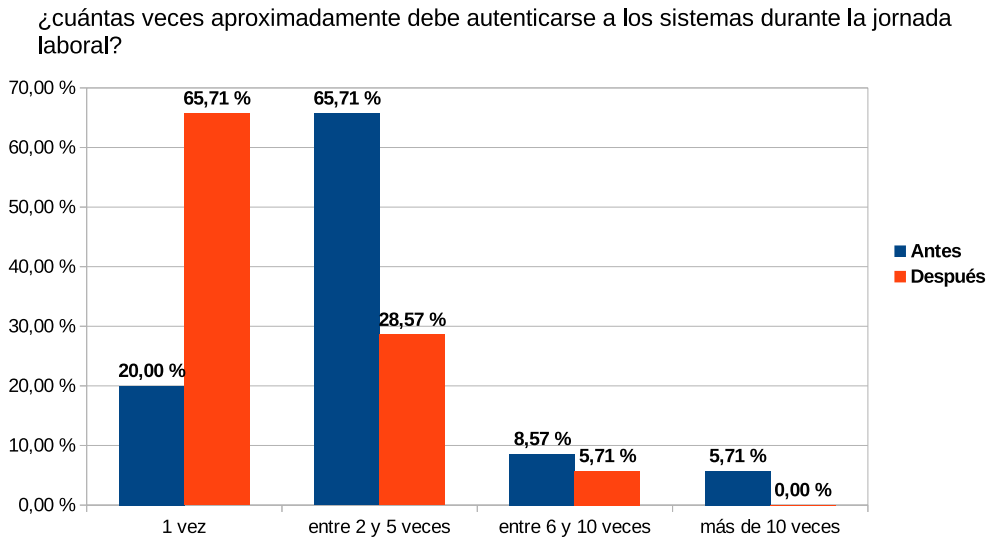

Figura B.1: Cantidad de logins durante la jornada laboral

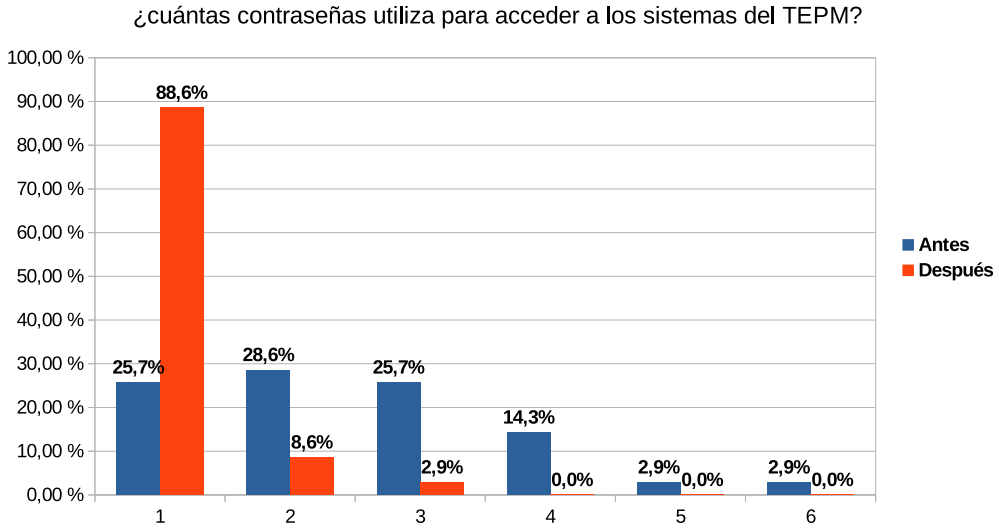

Figura B.2: Cantidad de contraseñas utilizadas

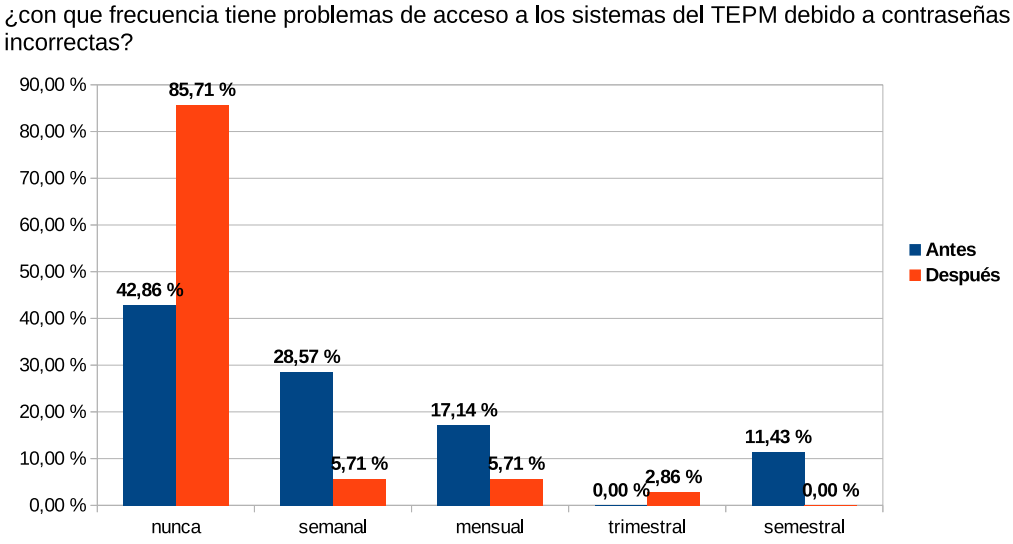

Figura B.3: Frecuencia de problemas relacionados al login

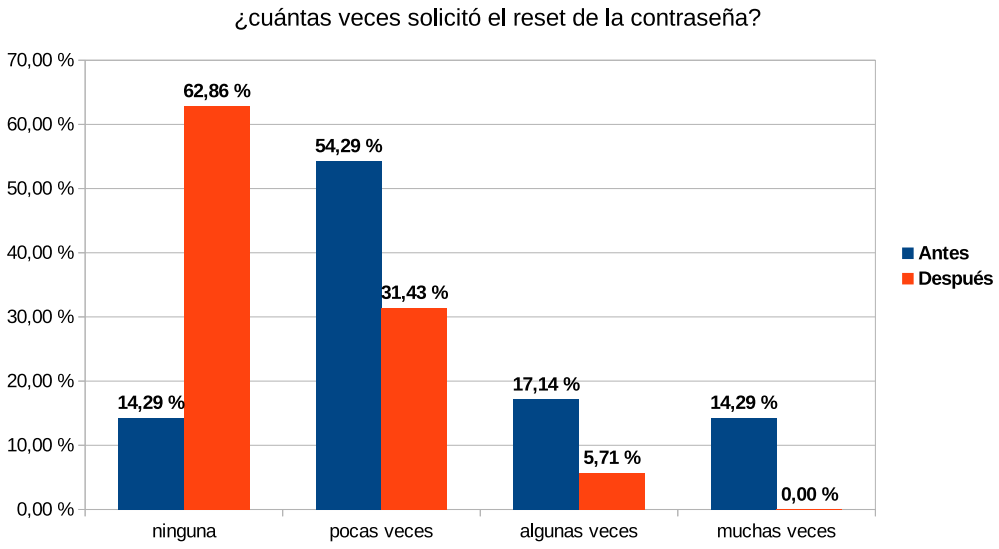

Figura B.4: Cantidad de veces que se solicitó un reset de la contraseña

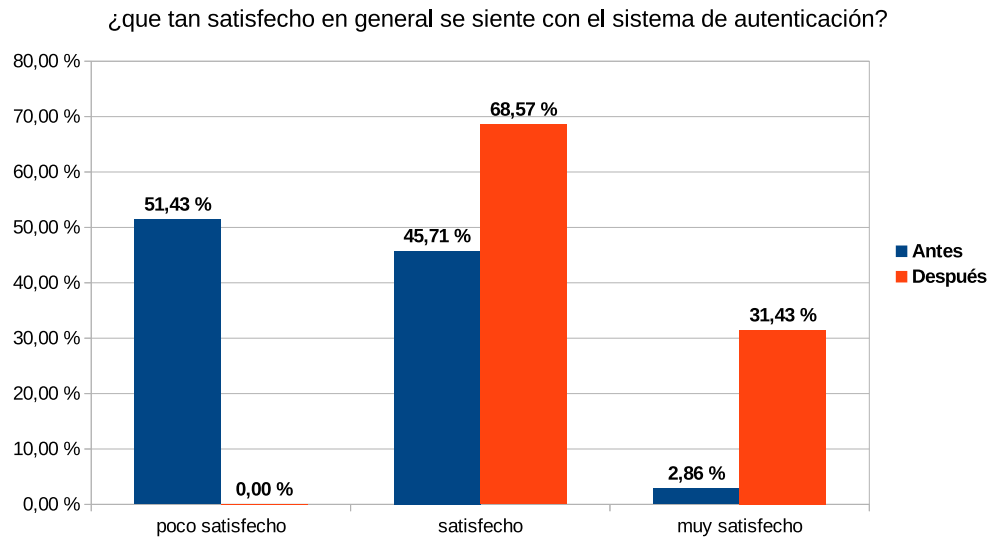

Figura B.5: Nivel de satisfacción del usuario en relación al sistema de autenticación

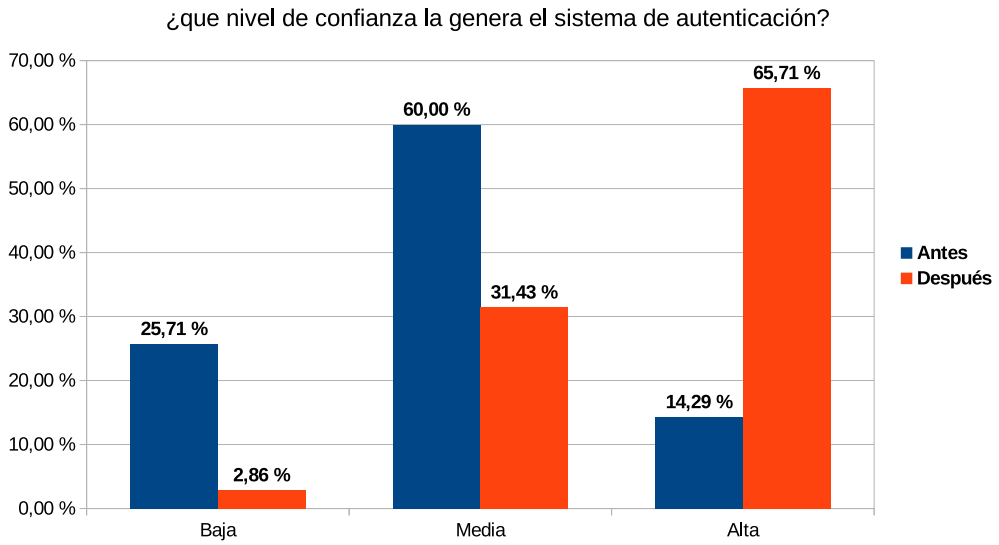

Figura B.6: Nivel de confianza del usuario en relación al sistema de autenticación

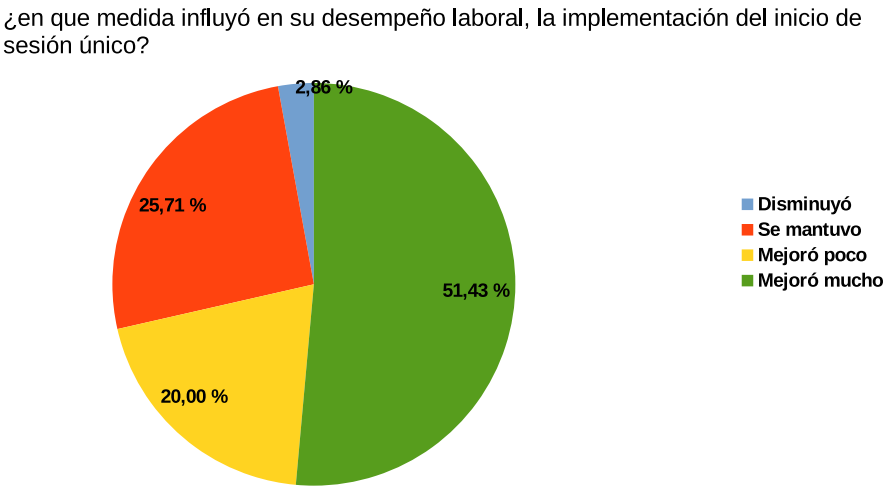

Figura B.7: Influencia del sistema SSO en el desempeño laboral luego de su implementación

C. Acrónimos y siglas

A

AAA Authenticaction, Authorization and Accounting

ACL Access Control List

ACE Access Control Entry

AD Active Directory

ADSI Active Directory Service Interfaces

AES Advanced Encryption Standard

AH Authentication Header

AJAX Asynchronous JavaScript And XML

ASN.1 Abstract Syntax Notation One

ASP Authentication Service Provider

AP Access Point

API Application Programming Interface

AS Authentication Server

ASF Alerting Standards Forum

B

BER Basic Encoding Rules

BDC Backup Domain Controller

B2B Business to Business

B2C Business to Customers

C

CA Certification Authority

CAS Central Authentication Service

CCMP Counter Mode with CBC-MAC Protocol

CHAP Challenge-Handshake Authentication Protocol

CIFS Common Internet File System

CLI Comnand Line Interface

CN Common Name

CRUD Create, Read, Update and Delete

CSS Cascading Style Sheets

D

DAC Discretionary Access Control

DC Domain Component

DCE Distributed Computing Environment

DES Data Encryption Standard

DHCP Dinamic Host Configuration Protocol

DIT Directory Information Tree

DN Distinguished Name

DNS Domain Name System

DOM Document Object Model

DSA Digital Signature Algorithm

E

EAP Extensible Authentication Protocol

EAPOL EAP over LAN

EAP-FAST EAP-Flexible Authentication via Secure Tunneling

EGID Effective Group ID

ESP Encapsulating Security Payload

ESSO Enterprise SSO

EUID Effective User ID

F

FQDN Fully Qualified Domain Name

G

GID Group ID

GINA Graphical Identification and Authentication

GPO Group Policy Object

GSS-API Generic Security Services API

H

HMAC Hashed Message Authentication Code

HOTP HMAC-Based One-Time Password

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

HTTPS Hyper Text Transfer Protocol Secure

I

IAM Identity and Access Management

IEEE Institute of Electrical and Electronic Engineers

IETF Internet Engineering Task Force

ID Identification

IDaaS Identity as a Service

IdP Identity Provider

IKE Internet Key Exchange

IMAP Internet Message Access Protocol

IMU Identity Management for Unix

IP Internet Protocol

IPC Inter-Process Communication

IPsec Internet Protocol security

ISDN Integrated Service Digital Network

ISAKMP Internet Security Association and Key Management Protocol

IT Information Technology

J

JSON JavaScript Object Notation

JWT JSON Web Token

K

KDC Key Distribution Center

KFW Kerberos For Windows

L

LAMP Linux, Apache, MySQL y PHP

LAPP Linux, Apache, Postgresql y PHP

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

LDIF Lightweight Directory Interchange Format

LEAP Lightweight EAP

LCP Link Control Protocol

LSA Local Security Authority

M

MAC Mandatory Access Control

MD4 Message-Digest Algorithm 4

MD5 Message-Digest Algorithm 5

M2M Machine-to-Machine

MIT Massachusetts Institute of Technology

MS-CHAP Microsoft Software CHAP

MTA Mail Transfer Agent

MVC Modelo-vista-controlador

N

NAS Network Access Server

NCP Network Control Protocol

NFS Network File System

NIS Network Information System

NTLM New Technology LAN Manager

NTP Network Time Protocol

NSS Name Services Switch

O

OASIS Organization for the Advancement of Structured Information Standards

OIDC OpenID Connect

OS Operating System

OSI Open Systems Interconnection

OTP One-Time Password

OU Organizational Unit

P

PAC Protected Access Credential

PAM Pluggable Authentication Modules

PAP Password Authentication Protocol

PEAP Protected EAP

PDC Primary Domain Controller

PGP Pretty Good Privacy

PHP PHP: Hypertext Preprocessor

PKI Public Key Infrastructure

PIN Personal Identification Number

PMK Pairwise Master Key

PTK Pairwise Transient Key

POP3 Post Office Protocol 3

PPP Point to Point Protocol

PSK Pre-Shared Key

Q

QoP Quality of Protection

R

RBAC Role Based Access Control

RADIUS Remote Authentication Dial-In User Service

REST REpresentational State Transfer

RDP Remote Desktop Protocol

RFC Request For Comment

RGID Real Group ID

RIA Rich Internet Application

RID Relative Identifier

RPC Remote Procedure Call

RSA Rivest, Shamir and Adleman

RTT Round Trip Time

RUID Real User ID

S

SA Security Association

SASL Simple Authentication and Security Layer

SAM Security Account Manager

SAML Security Assertion Markup Language

SGI Sistema de Gestión Interno

SID Security Identifier

SLIP Serial Line Internet Protocol

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

SP Service Provider

SPI Security Parameter Index

SPNEGO Simple and Protected GSSAPI Negotiation Mechanism

SOAP Simple Object Access Protocol

SSH Secure SHell

SSL Secure Socket Layer

SSO Single Sign-On

SSP Security Support Provider

SSPI Security Support Provider Interface

SSSD System Security Services Daemon

ST Service Ticket

T

TACACS+ Terminal Access Controller Access Control System Plus

TEPM Tribunal Electoral de la Provincia de Misiones

TCP Transport Control Protocol

TGS Ticket Granting Service

TGT Ticket Granting Ticket

TKIP Temporal Key Integrity Protocol

TLS Transport Layer Security

TTLS Tunneled TLS

TOTP Time-Based One-Time Password

2FA Two Factor Authentication

U

UDP User Datagram Protocol

UID User ID

URI Uniform Resource Identifier

URL Uniform Resource Locator

V

VM Virtual Machine

VPN Virtual Private Network

W

WEP Wired Equivalent Privacy

WPA Wifi Protected Access

WPA2 Wifi Protected Access 2

X

XAUTH eXtended Authentication

XML eXtensible Markup Language

XRI eXtensible Resource Identifier

Z

ZCS Zimbra Collaboration Server

Bibliografía

[1] H. Wang and C. Gong, "Design and implementation of unified identity authentication service based on AD," in *Computational Intelligence and Communication Networks (CICN), 2016 8th International Conference on*. IEEE, 2016, pp. 394–398.

[2] C. Herley and P. Van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Security and Privacy*, vol. 10, no. 1, pp. 28–36, 2012.

[3] B. M. David, A. C. A. Nascimento, and R. Tonicelli, "A Framework for Secure Single Sign-On," *Components*, pp. 52–58, 2008.

[4] A. Adams and M. A. Sasse, "Users are not the enemy," *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[5] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, "The Kerberos Network Authentication Service (V5)," RFC 4120 (Proposed Standard), RFC Editor, Tech. Rep. 4120, Jul. 2005.

[6] A. I. Ivanova and S. Vodanovich, "Single Sign-On Taxonomy," in *2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, vol. 1, 2017, pp. 151–155.

[7] ISO/IEC 27000:2018(E), "Information technology - Security techniques - Information security management systems - Overview and vocabulary," International Organization for Standardization, Standard, Feb. 2018.

[8] D. Todorov, *Mechanics of user identification and authentication: Fundamentals of identity management*. Auerbach Publications, 2007.

[9] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2021–2040, 2003.

[10] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, vol. 58, no. 7, 2015.

[11] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proceedings - IEEE Symposium on Security and Privacy*. IEEE, may 2012, pp. 553–567.

[12] C. Shen, T. Yu, H. Xu, G. Yang, and X. Guan, "User practice in password security: An empirical study of real-life passwords in the wild," *Computers and Security*, vol. 61, pp. 130–141, 2016.

[13] R. Shay, L. F. Cranor, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, and N. Christin, "Can long passwords be secure and usable?" in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, 2014.

[14] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "Hotp: An hmac-based one-time password algorithm," Internet Requests for Comments, RFC Editor, RFC 4226, December 2005.

[15] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "Totp: Time-based one-time password algorithm," Internet Requests for Comments, RFC Editor, RFC 6238, May 2011.

[16] L. A. Hart, "Single Sign-On: Holy Grail of Holy Crap!" *Global Information Assurance Certification Paper*, vol. GSEC v.1.4, 2003.

[17] A. Pashalidis and C. J. Mitchell, "A taxonomy of single sign-on systems," in *Australasian conference on information security and privacy*, vol. 2727. Springer, 2003, pp. 249–264.

[18] J. De Clercq, "Single Sign-On Architectures," in *Infrastructure Security*, vol. 2437. Springer, 2002, pp. 40–58.

[19] V. Radha and D. H. Reddy, "A Survey on Single Sign-On Techniques," *Procedia Technology*, vol. 4, pp. 134–139, 2012.

[20] A. H. Al-Ghushami, N. H. Zakaria, N. Katuk, and A. Mohammed, "Analysis of Single Sign-On Protocols from the Perspective of Architecture Deployment, Security and Usability," 2015.

[21] K. Geisshirt, *Pluggable Authentication Modules*. Birmingham, UK: Packt Publishing Ltd., 2007.

[22] D. Simpson, S. Mandalika, and A. M. Gorzelany. (2021) Security Principals - Microsoft Docs. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/security-principals

[23] J. Gerend, K. Downie, and E. Ross. (2021) Credentials Processes in Windows Authentication — Microsoft Docs. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/security/windows-authentication/credentials-processes-in-windows-authentication

[24] X. Yang, "The Use of One-Time Password and RADIUS Authentication in a GSS-API Architecture," Master Thesis, Royal Institute of Technology (KTH), 2006.

[25] J. Linn, "Generic Security Service Application Program Interface Version 2, Update 1," RFC 2743 (Proposed Standard), RFC Editor, Tech. Rep. 2743, Jan. 2000.

[26] L. Zhu, K. Jaganathan, and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," RFC 4121 (Proposed Standard), RFC Editor, Tech. Rep. 4121, Jul. 2005.

[27] L. Zhu, P. Leach, K. Jaganathan, and W. Ingersoll, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism," RFC 4178 (Proposed Standard), RFC Editor, Tech. Rep. 4178, Oct. 2005.

[28] "The ActiveX Core Technology Reference - Chapter 11: NTLM," 1999. [Online]. Available: http://www.opengroup.org/onlinepubs/009899899/NCH1222X.HTM#ntlm

[29] E. Glass. (2006) The NTLM Authentication Protocol and Security Support Provider. [Online]. Available: http://davenport.sourceforge.net/ntlm.html

[30] A. Ashcraft, K. Sharkey, and D. Coulter. (2022) Microsoft NTLM. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/secauthn/microsoft-ntlm

[31] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, dec 1978. [Online]. Available: http://portal.acm.org/citation.cfm?doid=359657.359659

[32] J. G. Myers, "Simple Authentication and Security Layer (SASL)," RFC 2222, Tech. Rep. 2222, oct 1997.

[33] K. Zeilenga and A. Melnikov, "Simple Authentication and Security Layer (SASL)," RFC 4422, RFC Editor, Tech. Rep. 4422, jun 2006.

[34] A. Melnikov, "The Kerberos V5 ("GSSAPI") Simple Authentication and Security Layer (SASL) Mechanism," RFC 4752, RFC Editor, Tech. Rep. 4752, nov 2006.

[35] K. Zeilenga, "The PLAIN Simple Authentication and Security Layer (SASL) Mechanism," RFC 4616, RFC Editor, Tech. Rep. 4616, aug 2006.

[36] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP): The Protocol," RFC 4511, RFC Editor, Tech. Rep. 4511, jun 2006.

[37] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Protocol Architecture," RFC 4251, RFC Editor, RFC 4251, jan 2006.

[38] J. A. Salowey, V. Welch, J. Hutzelman, and J. Galbraith, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol," RFC 4462, RFC Editor, Tech. Rep. 4462, may 2006.

[39] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Authentication Protocol," RFC 4252, RFC Editor, RFC 4252, jan 2006.

[40] H. Nielsen, R. T. Fielding, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.0," RFC 1945, Tech. Rep. 1945, may 1996.

[41] H. Nielsen, J. Mogul, L. M. Masinter, R. T. Fielding, J. Gettys, P. J. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, Tech. Rep. 2616, jun 1999.

[42] R. T. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication," RFC 7235, Tech. Rep. 7235, jun 2014.

[43] M. Jones and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage," RFC 6750, Tech. Rep. 6750, oct 2012.

[44] M. Jones, J. Bradley, and N. Sakimura, "Json web token (jwt)," RFC 7519, RFC 7519, May 2015.

[45] P. J. Franks, P. Hallam-Baker, L. C. Stewart, J. L. Hostetler, S. Lawrence, P. J. Leach, and A. Luotonen, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, Tech. Rep. 2617, jun 1999.

[46] R. Shekh-Yusef, D. Ahrens, and S. Bremer, "HTTP Digest Access Authentication," RFC 7616, Tech. Rep. 7616, sep 2015.

[47] K. Jaganathan, L. Zhu, and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows," RFC 4559, Tech. Rep. 4559, jun 2006.

[48] S. Farrell, P. E. Hoffman, and M. Thomas, "HTTP Origin-Bound Authentication (HOBA)," RFC 7486, Tech. Rep. 7486, mar 2015.

[49] K. D. L. a. E. Lewis, "Web Single Sign-On Authentication using SAML," vol. 2, pp. 41–48, 2009. [Online]. Available: http://arxiv.org/abs/0909.2368

[50] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Tech. Rep. 6749, oct 2012.

[51] V. Bertocci and A. Chiarelli, "OAuth2 and OpenID Connect : The Professional Guide - Beta," pp. 1–109, 2020.

[52] V. Beltran, "Characterization of web single sign-on protocols," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 24–30, 2016.

[53] J. Vollbrecht, J. D. Carlson, L. Blunk, D. B. D. Aboba, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748, Tech. Rep. 3748, jun 2004.

[54] K. Seo and S. Kent, "Security Architecture for the Internet Protocol," RFC 4301, Tech. Rep. 4301, dec 2005.

[55] S. Kent, "IP Authentication Header," RFC 4302, Tech. Rep. 4302, dec 2005.

[56] ——, "IP Encapsulating Security Payload (ESP)," RFC 4303, Tech. Rep. 4303, dec 2005.

[57] D. Carrel and D. Harkins, "The Internet Key Exchange (IKE)," RFC 2409, Tech. Rep. 2409, nov 1998.

[58] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," RFC 4306, Tech. Rep. 4306, dec 2005.

[59] A. Rubens, C. Rigney, S. Willens, and W. A. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, Tech. Rep. 2865, jun 2000.

[60] P. R. Calhoun and D. B. D. Aboba, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)," RFC 3579, Tech. Rep. 3579, sep 2003.

[61] M. A. Thakur and R. Gaikwad, "User identity and access management trends in it infrastructure- An overview," in *2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015*, 2015.

[62] R. J. Witty, A. Allan, J. Enck, and R. Wagner, "Identity and Access Management Defined," *Research Study SPA-21-3430, Gartner*, 2003.

[63] O. Toelen, "Identity and access management," Master Thesis, Eindhoven University of Technology, 2008.

[64] E. D. Lackey, *Fedora 17 FreeIPA: Identity/Policy Management*. Red Hat, 2012.

[65] Z. Y. Wu, W. P. Huang, and L. Yu, "Design and Implementation of Unified Identity Authentication System Based on LDAP in Digital Campus," *Advanced Materials Research*, vol. 912-914, pp. 1213–1217, 2014.

[66] D. Mazurek, "CAS Protocol 3.0 Specification," 2017. [Online]. Available: https://apereo.github.io/cas/6.5.x/protocol/CAS-Protocol-Specification.html

[67] S.-m. Pietilä, "Integration of a heterogeneous authentication environment to Active Directory," Master Thesis, JAMK University of Applied Sciences, 2015.

[68] D. Pal, "Integrating Linux systems with Active Directory," Security Camp at Boston University, Tech. Rep., 2015.

[69] F. Delehaye. (2019) Windows Integration Guide Red Hat Enterprise Linux 7. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/windows_integration_guide/index

[70] J. J. Garrett *et al.*, "Ajax: A new approach to web applications," Adaptative Path, San Francisco, CA, USA, Tech. Rep., 2005. [Online]. Available: https://www.scriptol.fr/ajax/ajax_adaptive_path.pdf

[71] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine, 2000.

[72] M. Lang, "Identity and Access Management : The Promise and the Payoff," Netegrity, Tech. Rep., 2003. [Online]. Available: https://silo.tips/download/identity-and-access-management-the-promise-and-the-payoff

[73] S. Zrelli, N. Okabe, and Y. Shinoda, "EAP-kerberos: A low latency EAP authentication method for faster handoffs in wireless access networks," *IEICE Transactions on Information and Systems*, vol. E95-D, no. 2, pp. 490–502, 2012.

[74] Y. Yang, H. Li, X. Cheng, X. Yang, and Y. Huo, "A High Security Signature Algorithm Based on Kerberos for REST-style Cloud Storage Service," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE, 2020, pp. 0176–0182.