UNIVERSIDAD DE GRANADA

Seguridad: Autenticación SSO (single sign-on)

Ingeniería de Sistemas de Información

José Miguel Aguado Coca

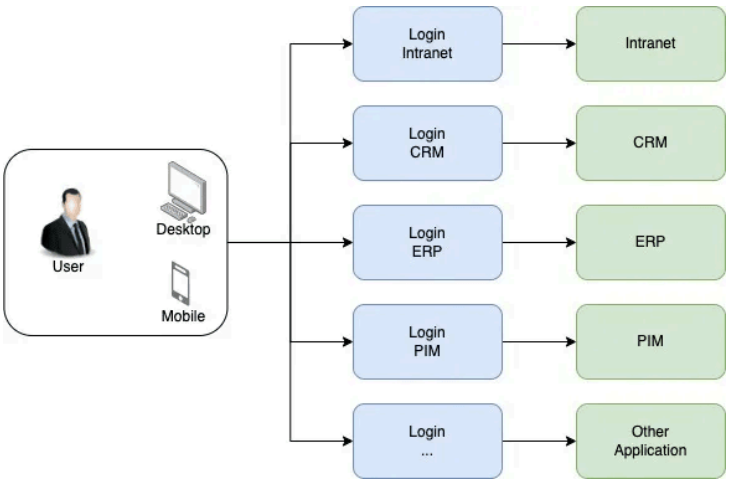
Índice

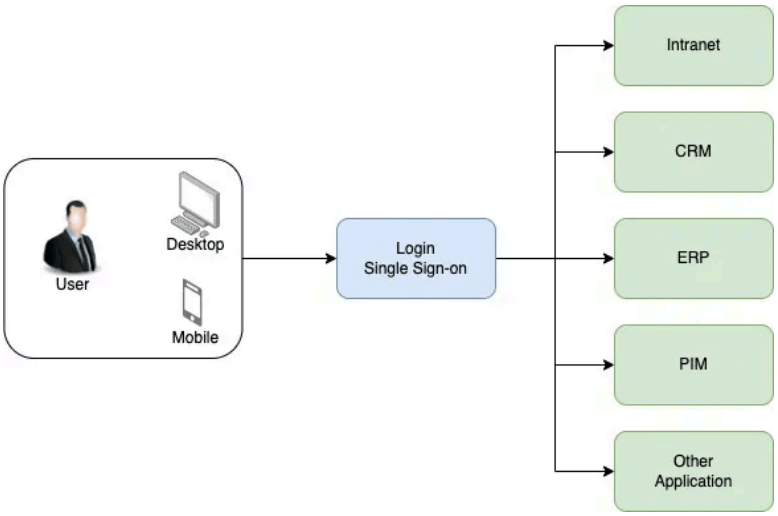
Motivación

En la actualidad, es muy común tener cuentas alojadas en distintos proveedores de servicios, con nombres de usuario distintos pero contraseñas iguales. Esto es un gran riesgo de seguridad al exponer la misma contraseña en distintos sitios, ya que cada uno puede tener distintas vulnerabilidades.
Necesitamos algún método con el cuál introduciendo una sola vez la contraseña podamos iniciar sesión en todos los proveedores de servicios que queramos. Para esto existe SSO.

Introducción

SSO (Single Sign On) es un método de control de accesos el cuál pregunta a un usuario una vez por su login y contraseña, y le permite acceder a todos los recursos y servicios autorizados sin necesidad de volver a iniciar sesión de nuevo. SSO es un componente de la FIM (Gestión de Identidad Federada). Entre SSO y FIM hay una gran diferencia. El primero sólo nos permitirá iniciar sesión en aquellas aplicaciones o servicios encontrados en una sola empresa y el segundo nos permitirá iniciar sesión en cualquier aplicación y cualquier servicio de cualquier empresa. Por lo tanto, FIM proporciona SSO y lo lleva al siguiente nivel permitiendo a los usuarios aplicar su solución SSO fuera de su empresa confiando en terceras partes. SSO como es obvio, no proporciona FIM.

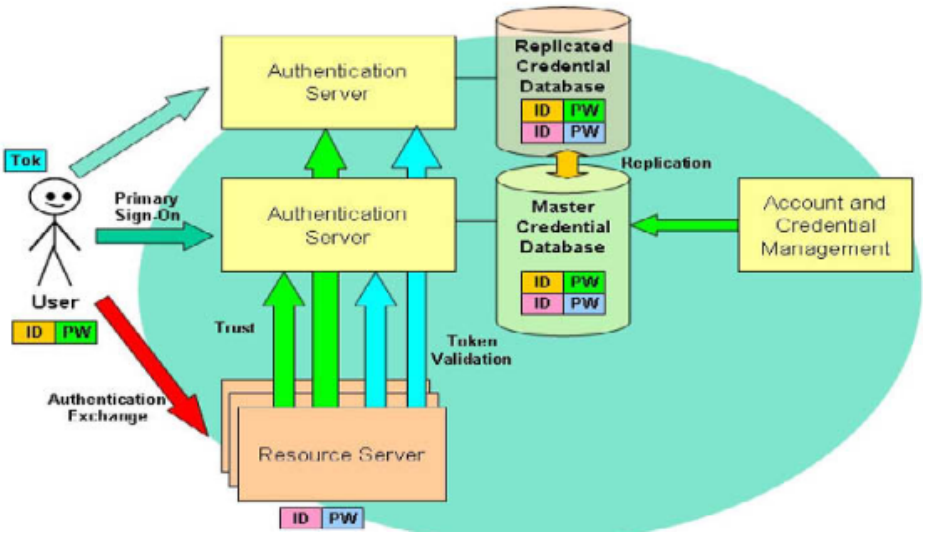




Clasificación Arquitecturas SSO

Antes de adentrarme en detalles más específicos de las posibles arquitecturas, tenemos que distinguir dos grupos, las “Arquitecturas SSO Simples” y las “Arquitecturas SSO Complejas”.

En las arquitecturas simples, el usuario querrá acceder a una serie de recursos por lo que se le solicitará el usuario y la contraseña. Estos son enviados a un servidor de autenticación, el cuál genera un token de sesión para el cliente. El cliente devuelve este token al servidor, y si es válido, se le permite el acceso al recurso. De forma contraria, se le deniega.

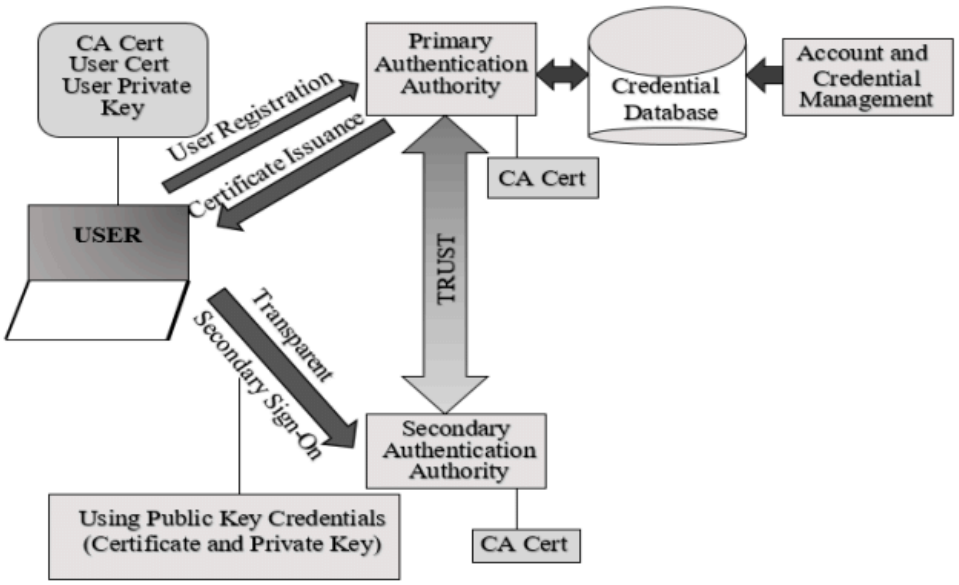


En las arquitecturas complejas el usuario va a querer acceder a distintas plataformas, por lo que va a haber distintas autoridades de autenticación (autoridad del dominio al que quiere acceder el usuario). Estas autoridades serán primarias (dominio al que más accede el usuario) o secundarias (segundo dominio al que menos accede el usuario).
A su vez, las arquitecturas complejas se dividen en las que solo usan un set de credenciales (Arquitectura basada en token y Arquitectura basada en PKI) y las que usan distintas credenciales (Caché en el Cliente y Caché en el Servidor). Voy a profundizar más en cada una de ellas.

SSO con PKI

El usuario se identifica en una Autoridad de Autenticación primaria y recibe un certificado (que contiene una clave pública, una clave privada, e información adicional sobre esta autoridad y el usuario). Cuando el usuario quiera acceder a alguno de los recursos, creará un token en el que incluirá este certificado. El que recibe la solicitud, contacta con la autoridad que generó este certificado, y si la información del token coincide con la de la CA, se le permite el acceso.

- Pros de esta arquitectura: al haber solo un set de credenciales, se simplifica todo para el usuario. El cifrado es asimétrico (clave privada y clave pública), por lo que el nivel de seguridad va a ser muy alto.
- Contras: el entorno de la infraestructura debe ser homogéneo, por lo que se debe usar la misma metodología para todo (protocolos). La validación de los certificados puede llegar a ser compleja y requerir mucho procesamiento en el lado del cliente.

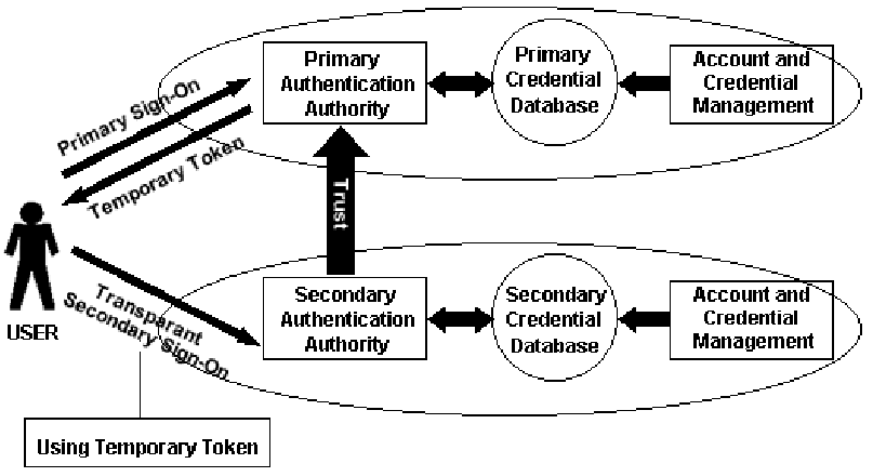


SSO con Token y Protocolo Kerberos

Para explicar esta arquitectura necesito adelantar uno de los protocolos que utiliza SSO (los cuáles explicaré más adelante). Este protocolo es Kerberos. Kerberos es el protocolo estándar usado en la mayoría de plataformas. Funciona con tickets, que son validados entre los dominios con criptografía y transportados con RPC (Remote Call Procedure). Con el protocolo Kerberos los usuarios necesitan autenticarse ante el KDC (Key Distribution Centre) que es el que proporciona el ticket usando una clave secreta.

El usuario se autentica y recibe un token temporal, el cuál guarda en su ordenador y lo usará cuando quiera acceder a un segundo dominio. Este token es validado con métodos de criptografía basada en clave secreta, establecidos entre el primer y el segundo dominio. Así se establece una relación de confianza entre ambos.
En el caso de Kerberos, los usuarios se autentican ante el KDC. Si las credenciales son válidas, el usuario recibe un ticket (token) que permite al usuario solicitar más tickets con los que acceder a los demás recursos del dominio primario y del dominio secundario.

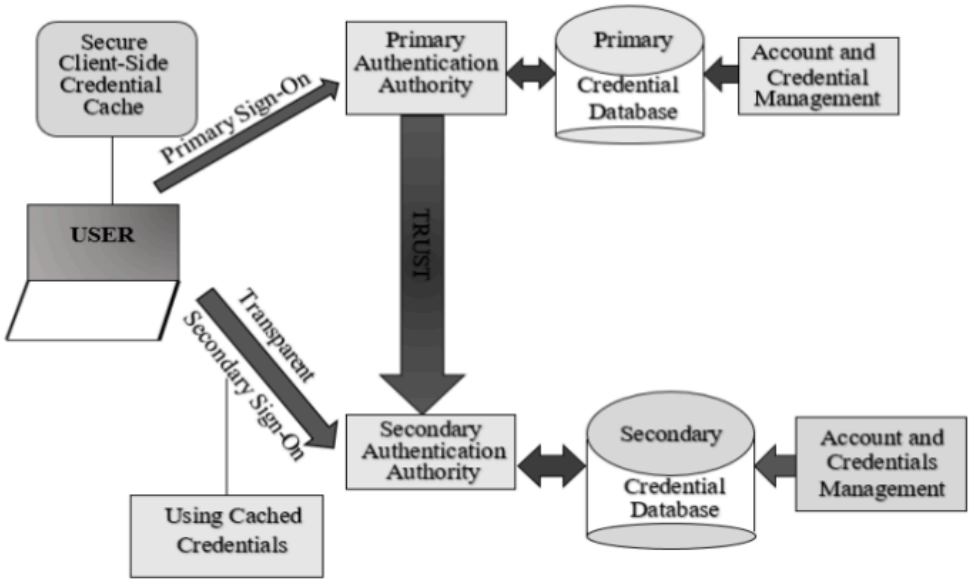
- Pros de esta arquitectura: al igual que en SSO con PKI, al haber solo un set de credenciales, se simplifica todo para el usuario.
- Contras: el entorno de la infraestructura debe ser homogéneo, por lo que se debe usar la misma metodología para todo (protocolos). Al usarse criptografía simétrica, si se ve comprometida la clave para cifrar y descifrar, pueden existir vulnerabilidades en el sistema de SSO.



Secure Client-Side Credential Caching

Esta arquitectura se basa en el cliente, dónde se realiza toda la autenticación. El usuario se autentica una vez en la autoridad primaria y el SSO automáticamente lo autenticará en el resto de aplicaciones. Es muy importante que las credenciales en la caché del cliente sean almacenadas de forma segura, por lo que no es recomendable usar un ordenador que tenga brechas de seguridad, puesto que si nos consiguen atacar, conseguirán acceder a todas las aplicaciones vinculadas. Cada vez que accedemos a una nueva aplicación la caché debe actualizarse.

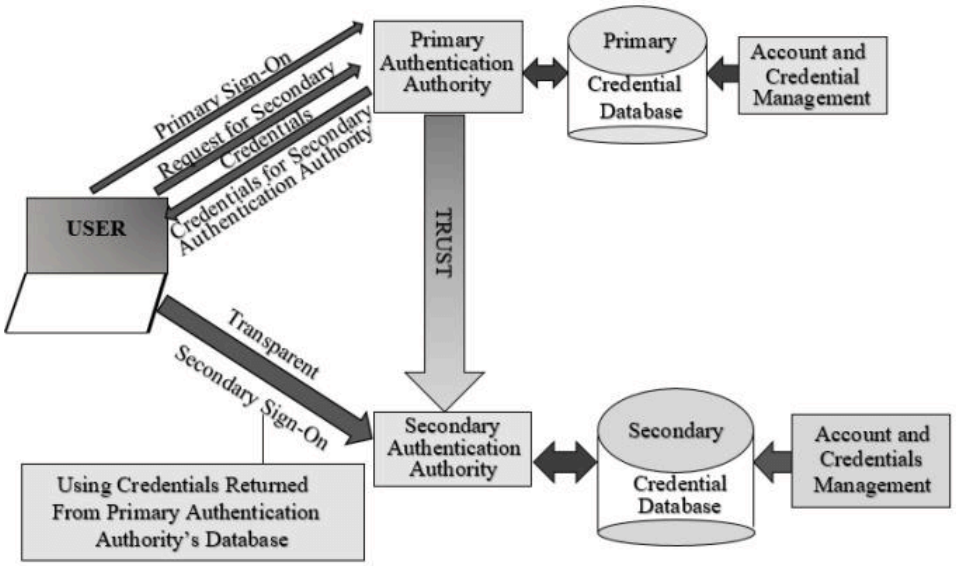
- Pros de esta arquitectura: se pueden usar varias contraseñas, es fácil de implementar ,no es necesario modificar mucho la aplicación y no requiere una infraestructura homogénea, por lo que se pueden usar distintos protocolos.
- Contras: muy poca flexibilidad, al haber varias contraseñas la vida del usuario y del administrador se complica y requiere mucha seguridad en el lado del cliente.



Secure Server-side Credential Caching

Es similar a la solución basada en el cliente, pero se usa una caché localizada en el servidor. La base de datos primaria contiene las credenciales primarias del usuario y la relación con las credenciales secundarias. La base de datos secundaria contiene solo las credenciales secundarias. El usuario inicia sesión en la autoridad primaria junto a una solicitud de las credenciales secundarias, y estas se les devuelve. De forma transparente, el usuario inicia sesión en la autoridad secundaria haciendo uso de estas credenciales secundarias.

- Pros de esta arquitectura: se pueden usar varias contraseñas y no requiere una infraestructura homogénea, por lo que se pueden usar distintos protocolos.
- Contras: requiere un mecanismo de sincronización entre credenciales, la vida del usuario y del administrador se complica por tener que usar varios sets de credenciales y la disponibilidad del servidor debe ser alta.

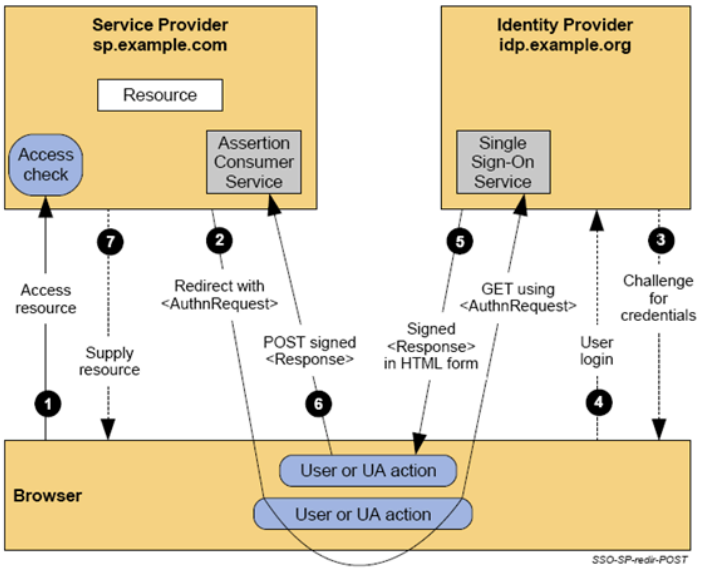


Distintos Protocolos en SSO

Los protocolos más usados en la web son los siguientes descritos a continuación.

SAML

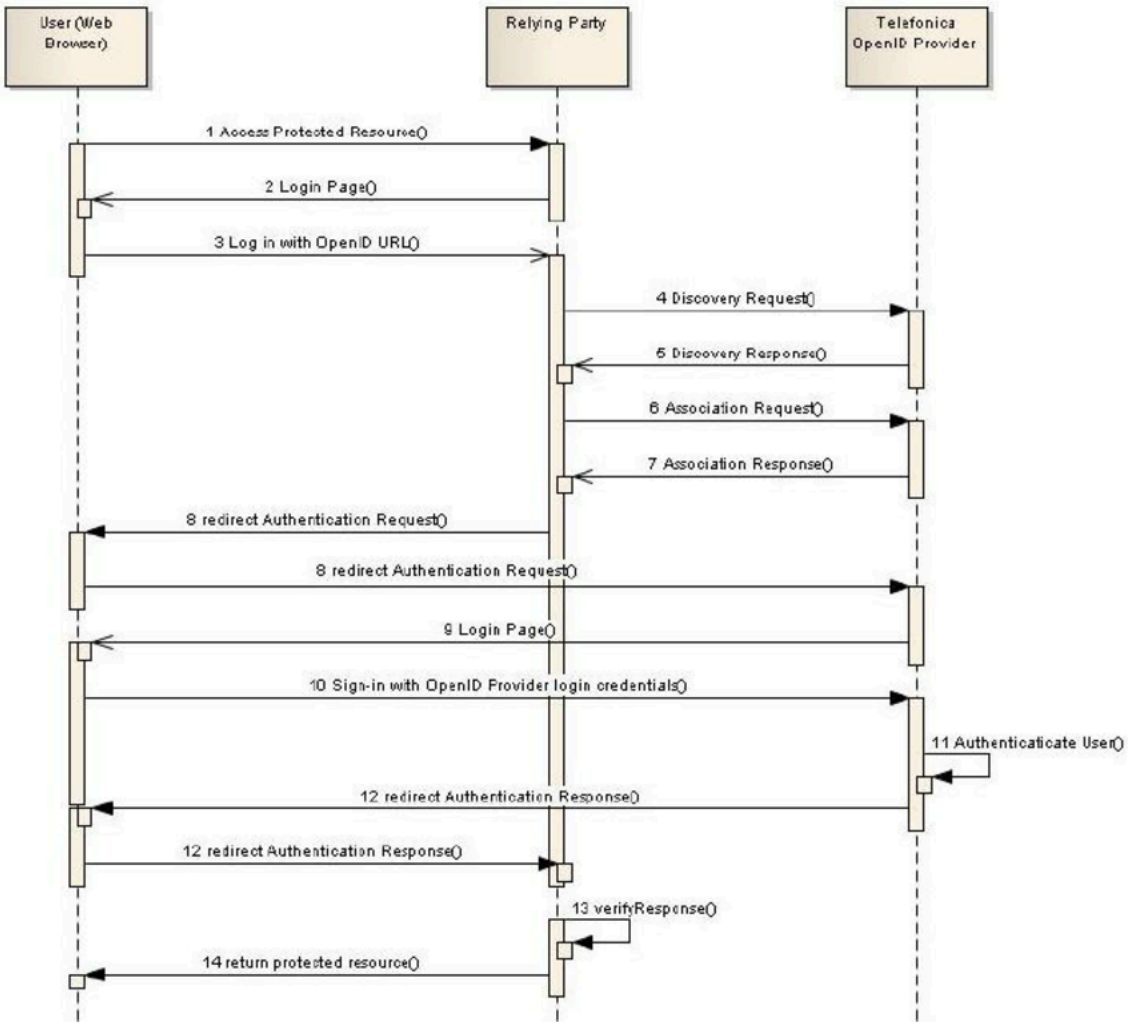
SAML es un protocolo basado en XML el cuál se utiliza para comunicar las identidades de usuarios en distintos sitios. Es decir, se permite la comunicación entre dominios con distinta metodología de autenticación. Las entidades que participan en SAML son el usuario, el Identity Provider y el Service Provider. El IdP es el encargado de almacenar y gestionar la identidad de los usuarios. Contiene una lista con detalles sobre esto, como grupos roles… El service provider es el que ofrece el recurso que quiere el usuario.



1) El usuario intenta acceder a un recurso del Service Provider.
2) El usuario es redirigido al Identity Provider con un REQUEST de autenticación
3) El IdP solicita al usuario sus credenciales.
4) El usuario inicia sesión.
5) El usuario obtiene una respuesta firmada en formato HTML RESPONSE.
6) El RESPONSE se le manda al Assertion Consumer Service, encargado de procesar la respuesta SAML y autenticar al usuario.
7) Se le proporciona el recurso al usuario.

OpenID

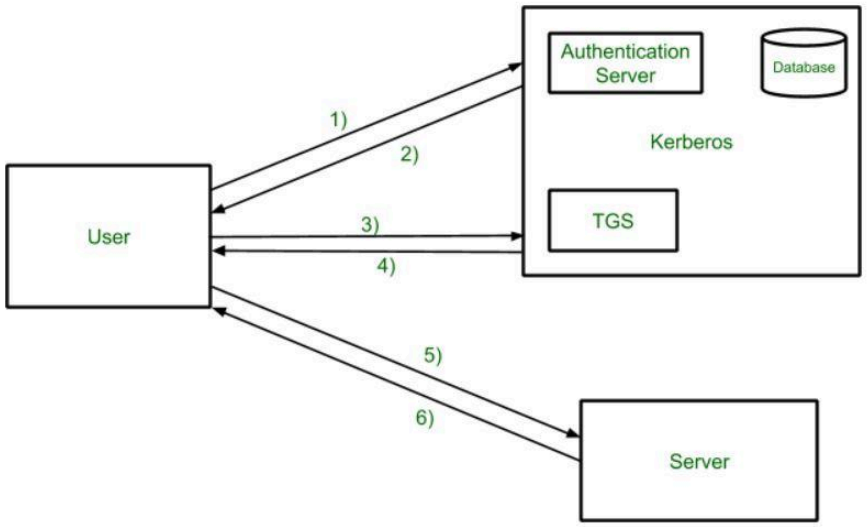
OpenID es otra de las soluciones SSO que proporciona un mecanismo de autenticación al usuario de forma descentralizada. Aquí tenemos cuatro importantes términos, el del usuario, el del Identity Provider, el de Relying Party y el Identifier o OpenID. El RP es el sitio que quiere verificar el usuario. El identifier es la url elegida por el usuario final para nombrar a su identidad. Por ejemplo, si me llamo josemi, http://josemi.myopenid.com. El IdP en este contexto se encarga de registrar el Identifier y autenticar.
Una vez conocemos estos conceptos, podemos ver su diagrama de secuencia para comprender mejor su funcionamiento.



1) El usuario desde su navegador quiere acceder a un recurso protegido encontrado en el Relying Party.
2) El RP nos proporciona el formulario de inicio de sesión.
3) El usuario inicia sesión con su OpenID URL.
4) El RP manda un DISCOVERY para descubrir dónde se encuentra el Idp.
5) El IdP le responde al DISCOVERY.
6) El RP establece una asociación de confianza con el IdP.
7) El IdP devuelve la confianza.
8) Como el RP ya conoce el IdP, la solicitud de autenticación del usuario puede ser redirigida al IdP.
9) El IdP manda un formulario de inicio de sesión al usuario.
10) El usuario inicia sesión con las credenciales de OpenID.
11) El IdP autentica al usuario comprobando que sus credenciales son correctas.
12) Se redirige la respuesta del IdP al usuario, y el usuario se las devuelve al RP.
13) Si todo ha ido bien al verificar el RP la respuesta del IdP, se le devuelve al usuario el recurso protegido al que quería acceder.

Kerberos

Aunque ya he hablado anteriormente de Kerberos, quiero profundizar algo más explicando su diagrama de flujo.



En Kerberos debemos conocer los siguientes términos, usuario, base de datos y Ticket Granting Server. La base de datos está conectada al servidor de autenticación, dónde se verifica el correcto acceso de los usuarios. El TGS proporciona los tickets para el servidor.

1) El usuario inicia sesión y solicita un ticket.
2) El servidor de autenticación comprueba que el acceso es correcto mirando en su base de datos, y proporciona al usuario el ticket y una clave de sesión. Esto se devuelve encriptado por la contraseña del usuario.
3) El usuario desencripta el mensaje usando su contraseña, y le manda el ticket al TGS. El ticket contiene información sobre nombres de usuario y network addresses.
4) El TGS desencripta el ticket enviado por el usuario y verifica que el ticket haya sido generado por el servidor de autenticación, y se le envía al usuario un autenticador.
5) El usuario manda el ticket y el autenticador al servidor.
6) El servidor comprueba que todo esté correcto y le da acceso al usuario. El usuario ya tendrá acceso siempre.

Beneficios, Problemas y Desafíos SSO

Los mayores beneficios que podemos obtener con SSO son:

- Mayor productividad para el usuario: el usuario no deberá memorizar contraseñas, solo iniciar sesión 1 vez y disfrutar de todos los servicios disponibles.
- Mayor productividad para el desarrollador: al implementar SSO los desarrolladores no tendrán que preocuparse por el tema de la autenticación en los distintos frameworks.
- Facilidad de uso: permitirá a los administradores administrar los usuarios de forma más sencilla.
- Seguridad: como es lógico, el implementar SSO va a ser que el sistema sea más seguro.

Los mayores problemas que podemos obtener con SSO son:

- Escalabilidad: la implementación de SSO se puede volver una tarea complicada.
- Seguridad en otro contexto: aunque el implementar aumenta la seguridad, si el usuario deja sesión abierta en su ordenador, y un

atacante consigue acceder a este, podrá acceder a todos los servicios disponibles a través de su sesión abierta.

Los mayores desafíos con SSO son:

- Ataques de phishing: es un reto el hacer que se diferencie entre los sitios webs reales y los falsos en los que iniciar sesión haciendo uso de SSO.
- Problemas de seguridad: muchos usuarios dudan sobre proporcionar al servidor su nombre de usuario y contraseña, y que esta información sensible sea almacenada en algún lugar.
- Resistencia al cambio: muchos usuarios no quieren adaptarse a estos sistemas SSO y prefieren seguir usando las mecánicas de los navegadores.
- Desconocimiento del proceso: la mayoría de usuarios no conocen cómo funciona esto, por lo que deciden abandonar el plan de usar SSO.

Referencias Bibliográficas

Bazaz, T., & Khalique, A. (2016). A Review on Single Sign on Enabling Technologies and Protocols. *International Journal Of Computer Applications*, *151*(11), 18-25. https://doi.org/10.5120/ijca2016911938

Patil, Anita & Pandit, Rakesh. (2013). Analysis of Single Sign on for Multiple Web Applications. https://www.researchgate.net/publication/369269685_Analysis_of_Single_Sign_on_for_Multiple_Web_Applications

PPT - Single Sign-On PowerPoint Presentation, free download - ID:507407. (2012, junio 29). SlideServe. https://www.slideserve.com/jersey/single-sign-on

Facchini, C. (2023, julio 31). *Architecture for Single Sign-on (SSO) and identity provider*. Medium. https://medium.com/@corrado.facchini/single-sign-on-sso-46ea458aec30

De Clercq, J. (2002). Single Sign-On Architectures. En *Infrastructure Security* (pp. 40–58). Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/3-540-45831-X_4

(S/f). Amazon.com. Recuperado el 15 de mayo de 2024, de https://aws.amazon.com/es/what-is/sso/

SakshiBhakhra Follow, S. (2019, julio 11). *Kerberos*. GeeksforGeeks. https://www.geeksforgeeks.org/kerberos/

Gomez Marmol, Felix & Kuhnen, M.Q. & Martinez Perez, Gregorio. (2011). Enhancing OpenID through a Reputation Framework. 1-18. 10.1007/978-3-642-23496-5_1. https://www.researchgate.net/publication/221108184_Enhancing_OpenID_through_a_Reputation_Framework

SAML2 IdP overview 1.1. (s/f). Eclipse.org. Recuperado el 15 de mayo de 2024, de https://wiki.eclipse.org/SAML2_IdP_Overview_1.1

N. Shaikh, K. Kasat and S. Jadhav, "Secured Authentication by Single Sign On (SSO): A Big Picture," *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, 2022, pp. 951-955, doi: 10.1109/ICCCIS56430.2022.10037708. https://ieeexplore.ieee.org/document/10037708