Universidad de Buenos Aires
Facultades de Ciencias Económicas, Cs Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Tema
“Seguridad Informática: Seguridad de servicios expuestos”

Autor:
Christian Narváez Montero

Tutor:
Dr. Pedro Hecht

Año de Presentación
2024

Año de Cohorte
2018

Declaración Jurada de origen de los contenidos

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

Resumen

Este trabajo final se centra en brindar algunas alternativas de solución a una problemática de seguridad actual, en donde diversas vulnerabilidades pueden llegar a afectar a múltiples servicios, internos o externos de una organización. Estas soluciones se basan en la utilización de protocolos de autorización como OAuth y OAuth 2.0, así como en la implementación de API Gateways, para garantizar una gestión eficaz y segura de las identidades y el acceso a recursos. Estos protocolos permiten a los usuarios conceder acceso a sus datos sin necesidad de compartir sus credenciales, ofreciendo así una capa adicional de seguridad y comodidad. Sin embargo, su implementación correcta, es vital para evitar vulnerabilidades que podrían ser explotadas por atacantes.

Además, se examinará el papel de los API Gateways, que actúan como intermediarios entre los clientes y los servicios backend. Estos gateways no solo facilitan la gestión del tráfico de API, sino que también integran medidas de seguridad que protegen las aplicaciones contra amenazas comunes. La implementación adecuada de un API Gateway, puede mitigar riesgos asociados con el acceso no autorizado y la sobrecarga de servicios.

Para contextualizar estas tecnologías dentro del entorno de la seguridad de la información, se explorará el Open Web Application Security Project (OWASP) y su listado de las diez vulnerabilidades más críticas, conocido como OWASP Top Ten. Este recurso es fundamental para entender las amenazas que enfrentan las aplicaciones web modernas y proporciona un marco para evaluar y mejorar la seguridad de las mismas.

Índice

INTRODUCCIÓN 1
CAPÍTULO 1. Control De Acceso - Seguridad Informática 3
1.1 Métodos De Autenticación 3
1.2 OAuth 4
1.2.1 Tipos De Roles En OAuth 4
1.2.2 Flujo OAuth 5
1.2.3 Endpoints 6
1.2.4 Scope 7
1.2.5 Inconvenientes De OAuth 1.0 7
1.3 OAuth 2.0 7
1.3.1 Tipos De Clientes 8
1.3.2 Token E Identificadores De OAuth 2.0 8
1.3.3 Procesos De Autorización De Oauth2 9
1.3.3.1 Authorization Code 9
1.3.3.2 Implicit 14
1.3.3.3 Password Credentials 15
1.3.3.4 Client Credentials 17
1.3.4 Registro de la Aplicación 18
1.3.5 Errores de Respuesta 18
1.3.6 Problemas en el protocolo OAuth - OAuth2 19
1.3.7 Consideraciones de Seguridad 19
1.4 OpenId Connect 20
1.4.1 ID Token 21
1.4.2 Userinfo 23
1.4.3 Scopes 23
1.4.4 Claims 23
1.5 Autenticación mediante Certificado 23
1.5.1 Proceso de autenticación 24
CAPÍTULO 2. Estructura - Metodología De Implementación 25
2.1 Definición y concepto de Api Gateway 25
2.2 Topología de Api Gateway 26
2.2.1 Topología Básica 26
2.2.2 Topología con funciones de seguridad 26
2.2.3 Topología con balanceo de carga y alta disponibilidad 27
2.2.4 Topología con agregación de servicios 28
2.2.5 Topología con microservicios internos 28
2.3 Funcionalidades de Api Gateway 29
2.3.1 Autenticación y Autorización 29
2.3.2 Control de Acceso 30
2.3.3 Monitoreo y Registro 31
2.3.4 Seguridad en el despliegue y configuraciones de un Api Gateway 31
2.4 Casos de Estudio 33
2.4.1 Caso de Estudio: Aplicación Financiera 33
CAPÍTULO 3. Estándares, Riesgos, Tendencias y Desafíos 36
3.1 Estándares 36
3.1.1 OWASP 37
3.1.1.1 Top 10 Año 2021 37
3.2 Riesgos 44

3.2.1 Análisis de vulnerabilidades..... 45
3.3 Impacto Comercial..... 46
3.4 Tendencias y desafíos..... 46
4. Conclusiones 47
5. Bibliografía..... 49

Índice de Imagines

Figura 1: Flujo de OAuth 5
Figura 2: Authorization Code 10
Figura 3: Implicit 14
Figura 4: Password Credentials 16
Figura 5: Client Credentials 17
Figura 6: Topología Básica 26
Figura 7: Topología con funciones de seguridad 27
Figura 8: Topología con balanceo de carga y alta disponibilidad 28
Figura 9: Topología con agregación de servicios 28
Figura 10: Topología con microservicios internos 29
Figura 11: Ejemplo de proceso de Transferencia de Fondos. 35

PROLOGO

En la era actual, o también llamado era digital, la seguridad de la información es un aspecto crítico para cualquier organización. Los avances tecnológicos han permitido el desarrollo de servicios innovadores que facilitan la gestión y el acceso a la información, pero también han incrementado los riesgos asociados a su exposición. La ciberseguridad se ha convertido en un pilar fundamental para proteger estos servicios y, por ende, la integridad de las organizaciones.

A continuación, se presenta un análisis sobre la importancia de implementar medidas de seguridad robustas para los servicios expuestos, tanto internos como externos. Se examina la falsa percepción de seguridad en los servicios y la necesidad de adoptar un enfoque integral para mitigar los crecientes ataques cibernéticos. Este análisis subraya la relevancia de encontrar un equilibrio entre seguridad y usabilidad, asegurando así la protección de la información sensible y la continuidad operativa de las organizaciones.

INTRODUCCIÓN

En un mundo cada vez más interconectado, la seguridad de las aplicaciones y los servicios web, es fundamental para proteger la integridad de los datos y la privacidad de los usuarios.

No es un secreto que, en la actualidad, surgen constantemente nuevos servicios que las organizaciones utilizan para obtener información. Esta información debería estar restringida únicamente a los sistemas o personas autorizadas. Ante el incremento continuo de las amenazas cibernéticas, resulta fundamental comprender los riesgos inherentes a la exposición de dichos servicios y adoptar las medidas adecuadas para salvaguardar la información crítica de las organizaciones.

Al poner en disponibilidad un servicio, una empresa habilita un acceso potencial a los datos, lo que a su vez representa una vulnerabilidad para la propia organización. Por lo tanto, es imperativo que los servicios expuestos dispongan de sólidas medidas de seguridad, ya sean de naturaleza externa o interna, que protejan tanto a la organización como a sus clientes.

Con el crecimiento exponencial de las APIs (Application Programming Interfaces) como facilitadoras de la comunicación entre sistemas, surge la necesidad de implementar mecanismos de autenticación y autorización robustos. En este contexto, OAuth y su sucesor OAuth 2.0 se han convertido en estándares clave para la gestión de accesos y la protección de recursos.

OAuth, permite a los usuarios otorgar acceso limitado a sus datos en un servicio a terceros, sin necesidad de compartir credenciales. Esta característica es esencial en entornos donde la interoperabilidad y la colaboración son prioritarias. Sin embargo, la implementación incorrecta de estos protocolos puede dar lugar a vulnerabilidades críticas, lo que resalta la importancia de seguir directrices de seguridad, como las proporcionadas por OWASP (Open Web Application Security Project). OWASP como veremos más adelante, ofrece un marco integral para identificar y mitigar riesgos asociados con la seguridad en aplicaciones web, siendo su Top Ten una referencia vital para los desarrolladores y arquitectos de sistemas.

La adopción de un API Gateway también juega un papel crucial en la arquitectura de servicios modernos. Este componente actúa como un intermediario

que no solo gestiona las solicitudes y respuestas entre los clientes y los servicios backend, sino que también implementa políticas de seguridad, autenticación y monitoreo. La combinación de un API Gateway con prácticas recomendadas de OWASP y la adopción de OAuth 2.0 crean un entorno más seguro y eficiente para el manejo de datos sensibles.

Esta tesis explorará las interrelaciones entre OAuth, OAuth 2.0, API Gateway y las mejores prácticas de OWASP, proporcionando un análisis detallado de cómo estas tecnologías pueden integrarse para mitigar riesgos en la seguridad de las aplicaciones.

Además a través de estudios de caso y ejemplos teóricos, se demostrarán las estrategias efectivas para fortalecer la seguridad de las APIs y garantizar la protección de los datos en un entorno digital complejo.

CAPÍTULO 1. Control De Acceso - Seguridad Informática

Control de acceso abarca la aprobación de acceso, por lo que el sistema adopta la decisión de conceder o rechazar una solicitud de acceso de un usuario, sobre los recursos o base a los que puede acceder.

Autenticación y control de acceso a menudo se combinan en un solo proceso, por lo que el acceso está aprobado sobre la base o recurso de la autenticación exitosa, la autenticación se puede lograr a través de varios factores, que puede ser algún dato que la persona tiene o que sabe, en la actualidad la manera más común de realizar este proceso se genera a través de un usuario y contraseña por lo tanto una persona puede acceder a la información cuando pueda demostrar algo que sabe (contraseña).

Para una mayor seguridad en la autenticación se puede agregar un sistema multifactorial, es decir que se puede implementar dos o más factores, esto permitirá que si existe un robo de uno de los factores el farsante podrá ingresar el primero factor de seguridad, pero al tratar de averiguar dos o más factores sería complicado.

Una vez que el usuario fue autenticado el siguiente paso a seguir es el control de acceso que permite y asegura que los usuario adecuado o autorizados puedan acceder a la información y puedan realizar los procesos a los que están autorizados (ver, modificar, agregar y/o eliminar la información).

1.1 Métodos De Autenticación

Cada año incrementa el uso de servicios IT en cada empresa, es por esto que surge la necesidad de agregar seguridad a cada desarrollo. Para esto podemos segurizar un servicio, donde se da una alineación entre seguridad y usabilidad, permitiendo desalentar a usuarios malintencionados y restringir el uso de los usuarios legítimos.

Para poder conseguir la alineación necesaria se debe implementar el uso de estándares que ayudan a mejor la seguridad brindando la información correcta a los usuarios autorizados.

El enfoque que se debe tener en cuenta al agregar segurizar servicios, es autenticar a los usuarios que los van a consumir estableciendo un nivel de confianza

entre los datos y el servicio, la autorización debe ser solicitada por los usuarios que van a realizar alguna tarea y por último tener herramientas necesarias que nos permitan durante el proceso identificar las partes involucradas, permitiendo garantizar la responsabilidad de cada parte.

Para cubrir los aspectos mencionado, exploraremos la funcionalidad de OAuth y sus derivaciones. Este es un estándar que permite dar seguridad a los servicios y sistemas expuestos.

A continuación, se especifica detalles del mismo.

1.2 OAuth

OAuth (Open Authorization) es una manera simple de autenticación que permite realizar flujos de información de manera segura para aplicaciones de escritorio, móviles y web. Este estándar surge por la necesidad del envió continuo de credenciales entre el cliente y el servidor, permitiendo un acceso limitado de la información.

OAuth en su página indica que: [1] “OAuth permite delegar la autorización de acceso de APIs y no es un protocolo de autenticación”, por lo tanto, OAuth permite proporciona flujos de autenticación específicos, para acceder a la información desde distintas aplicaciones o servicios desarrollados.

1.2.1 Tipos De Roles En OAuth

OAuth define 4 tipos de roles, que se detalla a continuación [2]:

Cliente: Es la aplicación de cara al cliente (aplicación de escritorio, web o móvil) desde donde se accede, a servicios determinados. Lo esencial para este proceso es contar con la autorización del sistema y la validación del usuario.

Resource Owner: Para este rol el usuario es el dueño del recurso, es el que autoriza a una aplicación acceder a su cuenta y el acceso va estar limitado al scope.

Resource Server: Servidor que aloja los recursos protegidos y es capaz de aceptar y responder a solicitudes de recursos protegidos mediantes access token.

Authorization Server: Es el servidor que verifica la identidad del usuario y devuelve una access token a la aplicación cliente. El servidor de autorización debe asegurar que los tokens de acceso no puedan ser creados, alterados o descubiertos

por individuos no autorizados.

El servidor de autorización puede ser el mismo servidor que el de recursos o una entidad separada. Un único servidor de autorización puede emitir access token aceptados por múltiples servidores de recursos.

1.2.2 Flujo OAuth

OAuth puede tener algunas diferencias sobre su flujo en función del Authorization Grant Type, es por eso por lo que OAuth puede implementarse en diversas condicionales y varía dependiendo si es por aplicaciones web, móvil o escritorio [2].

A continuación, se describirá el flujo que tiene OAuth:

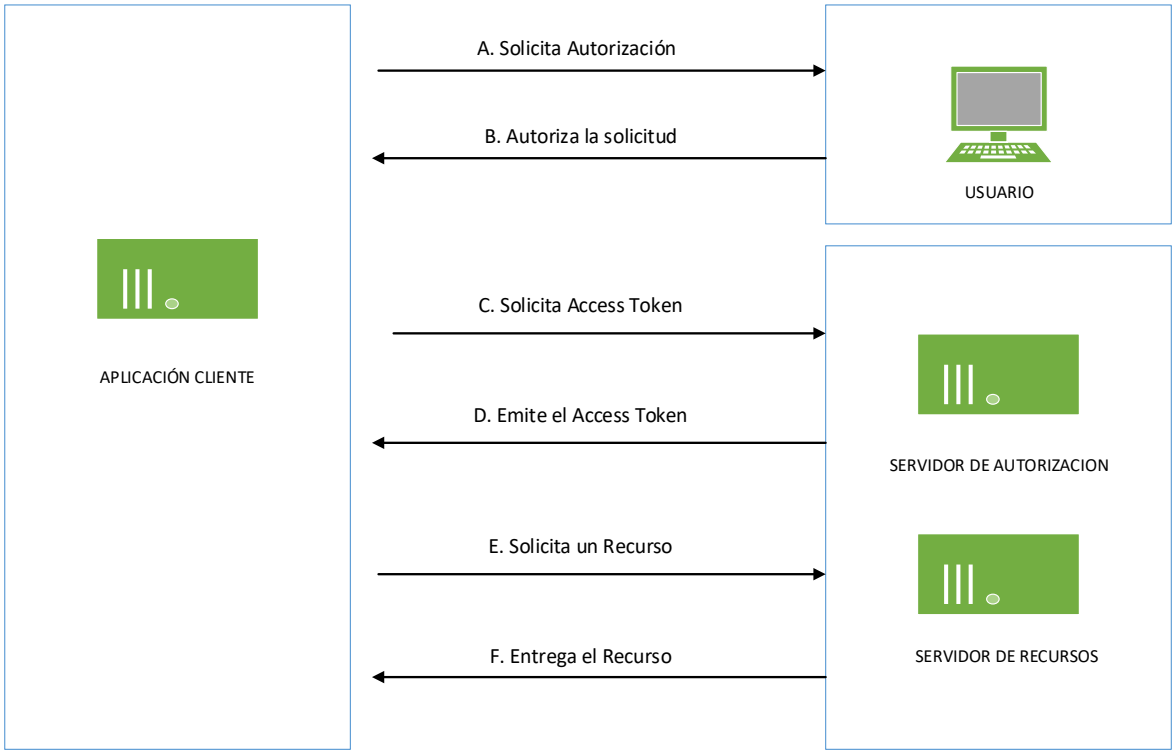

Figura 1: Flujo de OAuth

a) La aplicación cliente solicita autorización para acceder a un recurso de un servicio determinado.

b) El usuario recibe y autoriza la solicitud y la aplicación cliente recibe una autorización.

c) La aplicación cliente solicita un Access Token presentando la autenticación de su

identidad y la autorización otorgada al servidor de autorización, demostrando que es un cliente valido.

d) El servidor de autorización emite un Access Token a la aplicación cliente, si la identidad de la aplicación es autenticada y la autorización es válida. Al realizar este proceso finaliza la atentación.

e) La aplicación cliente solicita un recurso presentando el access token otorgado en el paso anterior enviando al servidor de recursos.

f) El servidor de recursos provee el recurso a la aplicación si el access token es válido.

1.2.3 Endpoints

En OAuth tienes dos endpoints o URL en el servidor para autorizar, utilizan tls y debe ser conocidas por la aplicación cliente. [3]

- Authorization Endpoint: con este endpoint el usuario tiene interacción ya que debe identificarse, es decir utilizado por el cliente para obtener la autorización del propietario del recurso.

  Si los parámetros enviados sin valor deben ser tratados como si hubieran sido omitidos de la solicitud y el servidor de autorización debe ignorar los parámetros de solicitud no reconocidos.

- Token Endpoint: en este endpoint no existe interacción de usuario sino de las aplicaciones. Se utiliza este endpoint para obtener un access token presentando su autorización o refresh token.

  Este endpoint es utilizado en casi todos los flujos de autenticación excepto en el flujo de implicit ya que emite un access token directamente. El servidor de autorización DEBE ignorar los parámetros de solicitud no reconocidos. Los parámetros de solicitud y respuesta NO DEBEN incluirse más de una vez.

  Ademes tiene un Endpoint cliente:

- Redirection endpoint: utilizado por el servidor de autorización para devolver la respuesta que contiene credenciales de autenticación al cliente a través del agente de usuario propietario del recurso.

1.2.4 Scope

Los scope son los permisos o listas de identificadores separados por espacios que permiten la autorización y consentimiento dentro de un recurso protegido. A su vez el servidor de autorización utiliza el parámetro de respuesta scope para informar al cliente del alcance del access token emitido, es decir, determina a que se solicita acceso por parte de la aplicación y que se autoriza por parte del usuario. Los scope dependen del entorno, es por eso que no siempre son iguales en cada sitio.

El cliente debe solicitar un access token con el mínimo de scope necesarios y el servidor de autorización debe tener en cuenta la identidad del cliente al elegir como cumplir con el scope solicitado y pueda emitir un access token con menos derecho que los solicita.

1.2.5 Inconvenientes De OAuth 1.0

En su versión OAuth1.0, fue diseñado para protección de datos personales, pero su seguridad tuvo severas fallas cuando actores de amenazas, aprovecharon a los usuarios desprevenidos y obtuvieron accesos a sus cuentas robando sus credenciales de acceso.

1.3 OAuth 2.0

OAuth 2.0 es un framework de una versión mejorada de OAuth, que contempla la misma funcionalidad y que permite que las aplicaciones tengan acceso limitado a las cuentas de los usuarios. Funciona delegando la autenticación del usuario a un servicio que gestione las cuentas, otorgando el acceso a las aplicaciones propias de la organización o de terceros. Con esta nueva versión se arreglaron las deficiencias que tenía el protocolo original resolviendo las complicaciones en la programación. [4]

Una característica importante de OAuth2 es que se basa en SSL que cifra la conexión, permite manejar el access token de manera segura y limitada en tiempo debido a que el access token caduca, además, es funcional por la utilización de scopes, también permite compartir datos sin tener que revelar la información del usuario, por último, se distingue de la versión anterior porque no utiliza firmas criptográficas para las comunicaciones en el flujo.

Para esta versión, los roles, como el flujo, son los mismos que OAuth que fueron detallados en el punto anterior **TIPOS DE ROLES EN OAUTH y FLUJO DE OAUTH**.

OAuth2 aumenta el rendimiento general del protocolo e incorpora grant Types más específicos. Algo destacable es que la validez de los tiempos de access tokens es más corta, lo que permite que el resource owner pueda revocar las autorizaciones.

1.3.1 Tipos De Clientes

OAuth2 reconoce dos tipos de clientes en función de autenticar de forma segura con el servidor de autorización, manteniendo la confidencialidad de las credenciales del usuario:

- Confidenciales: mantienen las credenciales de manera confidencial y capaz de autenticar de forma segura utilizando otros medios.
- Públicos: capaces de mantener sus credenciales e incapaces de autenticar de manera segura al cliente por cualquier otro medio.

1.3.2 Token E Identificadores De OAuth 2.0

Dentro de OAuth se maneja algunos token e identificadores que son:

Client Id: Identificador que representa la aplicación en el servidor de autenticación, es una cadena única que representa el registro de información proporcionada por el cliente, este identificador no es un secreto ya que está expuesto al propietario del recurso y es exclusivo del servidor de autorización. Aunque es público es mejor que terceros no puedan adivinarlo, es por eso que algunas implementaciones usan una cadena hexadecimal de 32 caracteres.

Client Secret: Clave secreta que es conocida por la aplicación y el servidor de autorización, esto permite comprobar si la aplicación está autorizada. Debe ser aleatoria para que no pueda ser adivinada, por lo que se debe evitar el uso de bibliotecas comunes de UUID, la manera segura de generar un secret es utilizar una biblioteca criptográfica, generando un valor de 256 bits y convertirlos en una representación hexadecimal.

Access Token: Clave proporcionada por el servidor de autorización que

permite el acceso a los servicios o Apis durante a un recurso protegido. Las credenciales del access token deben trasmitirse utilizando TLS con la autenticación del servidor.

Access token pueden tener varios formatos, estructuras y métodos de utilización como propiedades criptográficas, esto varía según los requisitos de seguridad del servidor de recursos.

Refresh Token: Clave proporcionada por el servidor de autorización a la aplicación que permite solicitar un nuevo access token cunado deja de ser invalido. Los refresh token deben ser confidenciales al momento del tránsito y almacenamiento y deben compartirse solo entre el servidor de autorización y al cliente que lo solicito.

En caso de que la autenticación del cliente no sea posible, el servidor de autorización debe implementar otros medios para detectar el abuso del access token.

1.3.3 Procesos De Autorización De Oauth2

Este estándar define 4 procesos o flujos de autorización (grant types), que permiten a la aplicación obtener el access token que da acceso a los datos expuestos. Los siguientes flujos son la respuesta a todos los escenarios del negocio que se pueden presentar al consumir algún servicio o Apis: [5]

- Authorization Code
- Implicit
- Password Credentials
- Client Credentials

1.3.3.1 Authorization Code

Para este flujo se requiere la intervención del usuario, este deberá autorizar de forma explícita el acceso a sus datos por parte de la aplicación y obtendrá un código que será proporcionado por el servidor de autenticación para que pueda recibir un access token [6].

El código de autorización que se proporciona, tiene algunos beneficios de

seguridad, uno de ellos es la autenticación del cliente, así como también la transmisión del access token directamente al cliente.

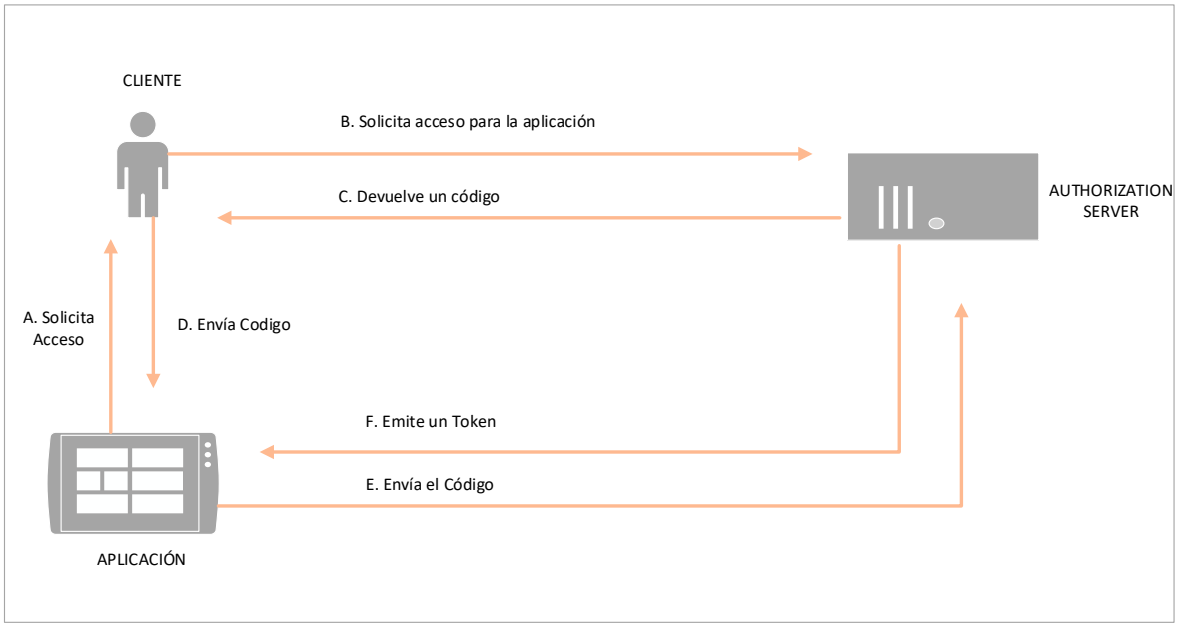

Figura 2: Authorization Code

La aplicación solicita consentimiento al usuario para acceder a los datos, enviando los scope a los que quiere acceder.

a. El usuario solicita permisos por parte de la aplicación y envía aprobación del acceso al servidor.
b. El servidor devuelve un código de acceso al usuario.
c. El código es re direccionado llegando a la aplicación.
d. La aplicación solicita acceso al servidor de autenticación enviando el código que obtuvo.
e. El servidor de autenticación envía el access token a la aplicación.

Solicitud de Autorización

El cliente construye el URI de solicitud (Uniform Resource Identifier) que agrega los parámetros al componente de consulta del URI endpoint.

a. **Login del Usuario:** Lo primero que se realiza en este flujo, es el inicio de sesión. El usuario inicia sesión y el cliente genera y envía una solicitud de sesión al servidor de autorización.

b. **Usuario se re direcciona al Servidor de Autenticación:** El cliente genera la solicitud de inicio de sesión para el servidor de autenticación, esta solicitud se envía de forma re direccionada y la información se enviará como el parámetro GET.

   Los parámetros que se envían son los siguientes:

- Client Id: Identificador de la aplicación.
- Redirect Uri o Call back: URL donde el servidor de autorización enviara el código de autorización, después que inicie sesión.
- Response Type: Tipo de respuesta que el servidor de autorización devolverá.
- Scope: Lista de permisos que la aplicación solicita al usuario. Se requiere cuando el cliente quiere acceder al servidor de recursos.
- State: Este parámetro es opcional, se puede utilizar para recuperar la información sobre la aplicación del cliente.

   Él envió de los parámetros son de la siguiente manera:

   https://authorization.server.com/authorize
   ?**response_type**=code
   &**client_id**=213f6de8-g987-4854-6r34-80a9b385cca7
   &**redirect_uri**=https://client.example.com/callback
   &**state**=confidencial
   &**scope**=Write

c. **Validación de la Solicitud:** El servidor de autorización debe validar los siguientes parámetros:

   - Client Id: Verifica si el cliente tiene el Id o si se le permite al cliente realizar los solicitado.
   - Redirect URI: El cliente puede usar el Redirect URI o si el URI se encuentra asociado con el cliente.
   - Response Type: Verifica si el cliente le permite usar el tipo de respuesta.
   - Scope: Verifica si el cliente se le permite usar los scope.

d. **Inicio de Login:** El servidor de autorización muestra el formulario de inicio de sesión.

e. **Credenciales:** En este paso el usuario debe insertar el nombre de usuario y contraseña para el inicio de sesión.

f. **Consentimiento:** Después de validar las credenciales, el servidor de autorización le pide al usuario los consentimientos específicos de los scopes, lo cual el usuario decide otorgar uno o más scope.

g. **Retorno del Código:** El servidor de autorización genera un código de autorización y lo envía al cliente mediante al URI especificado.

Cuando los parámetros fueron validados correctamente el servidor retornara la siguiente cadena con el respectivo código.

```
https://responsecliente.com/oauth2/callback
?code=xxxxxxx
&state= confidential
```

h. **Envió del código de Validación:** El cliente llama al servidor de autorización para validar el código recibido. Para este proceso envía:

- Código de autorización para validar.
- Client Id enviado apara asegurar que la solicitud proviene de ese cliente, así como también del Client Secret.
- Client Secret es la contraseña del cliente y debe almacenarse en un lugar seguro.

En este paso es donde el servidor de autorización realiza las validaciones anteriores, verifica el código de autorización enviado asegurando que este intacto, no alterado, no alterado y emitido por ese cliente.

```
grant_type=authorization_code
&code=xxxxxxxxxxx
&redirect_uri=https://responsecliente.com/oauth2/callback
&client_id=213f6de8-g987-4854-6r34-80a9b385cca7
&client_secrect=nH7TbHkgsjOWfgfhe45QD5EOtOH16nIusaVzZ4EI=
```

i. **Access Token Recibido**: El access token es creado por el servidor de autorización y lo devuelve al cliente. Con el token se genera más información:

- Token type: es uno de los más requeridos que permite otorgar acceso al portador.
- Expires_in: Duración del token.
- Refresh_token: Permite renovar el access token cuando caduca.

Visualizándose de la siguiente manera:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token" : "xxxxxxx",
  "token_type" : "Bearer",
  "expires_in" : 3600,
  "refresh_token" : "xxxxxxx",
  "scope" : "Write"
}
```

j. **Uso Access Token:** Con el access token adquirido puede usarlo para autenticarse en el servidor de recursos.

k. **Validación Token:** El servidor de recursos recibe una solicitud debe verificar la presencia e integridad del token.

l. **Acceder a los Recursos:** Cuando el servidor de recursos valida el token devolverá el recurso al cliente y eso puede mostrarle al usuario.

- Token type: Bearer es uno de los más requeridos que permite otorgar acceso al portador.
- Expires_in: Duración del token.
- Refresh_token: Permite renovar el access token cuando caduca.

m. **Uso Access Token:** Con el access token adquirido puede usarlo para autenticarse en el servidor de recursos.

n. **Validación Token:** El servidor de recursos recibe una solicitud debe verificar la presencia e integridad del token.

o. **Acceder a los Recursos:** Cuando el servidor de recursos valida el token devolverá el recurso al cliente y eso puede mostrarle al usuario.

1.3.3.2 Implicit

En este flujo el usuario también interactúa para dar consentimiento explícito, pero no necesita de un código como en el caso del flujo Authorization Code ya que el cliente recibe un access token directamente [6].

Al emitir el access token durante este flujo, el servidor de autenticación no autentica al cliente, en alguno de los casos la identidad del cliente puede ser verificada a través de la URI de direccionamiento que utiliza para entregar el access token al cliente. Al utilizar este flujo puede mejorar la respuesta y la eficiencia de algunos clientes, debido a que reduce la cantidad de viajes de ida y vuelta necesaria para obtener el access token.

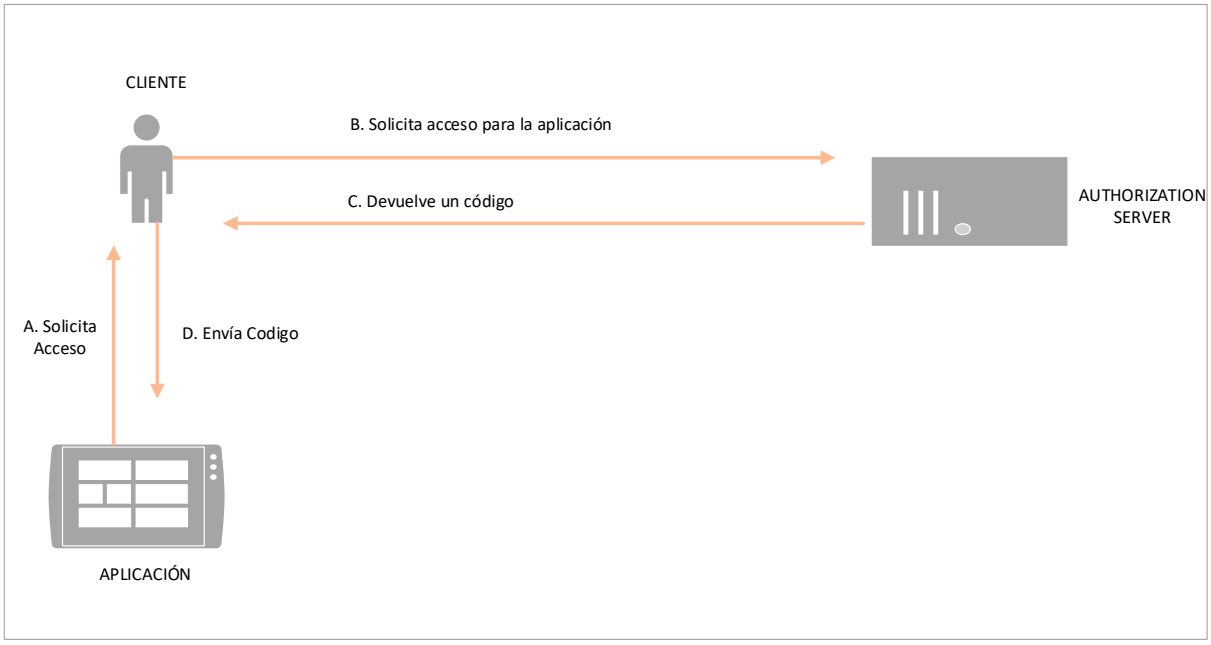

Figura 3: Implicit

a. La aplicación solicita consentimiento al usuario para acceder los datos donde envía los scope a los que quiere acceder.

b. El servidor de autorización dará una respuesta inmediata generando un

access token devuelto al cliente.

Cuando se utiliza el tipo de concesión implícita, el token de acceso se transmite en el fragmento de URI, lo que puede exponerlo a terceros no autorizados.

Los parámetros que se envían en este flujo son los siguientes:

https://authorization.server.com/authorize
?**response_type**=token
&**client_id**=213f6de8-g987-4854-6r34-80a9b385cca7
&**redirect_uri**=https://client.example.com/callback
&**state**=confidencial
&**scope**=Write

Si los parámetros son los correctos la respuesta seria la siguiente:

https://responsecliente.com/oauth2/callback
?**access_token**=xxxxxx
&**token_type**=example
&**expires_in**=3600
&**state**=confidencial

Una desventaja de este flujo es que no tiene un refresh token porque no existe un canal secundario, es por eso que necesita obtener un nuevo access token, además es posible que alguna aplicación maliciosa pueda inyectar el token de algún otro usuario y podría enviar información personal a otro usuario.

1.3.3.3 Password Credentials

Para este flujo se necesita la interacción del usuario debido a que debe proporcionar su usuario y contraseña en la aplicación cliente y no en el servidor de autenticación. Para este proceso los dueños del recurso deben confiar plenamente en la aplicación para que puedan introducir sus credenciales [6].

Las credenciales solo deben usarse cuando existe un grado de confianza entre el propietario de recurso y el cliente y cuando no existe otro tipo de gran type disponible, como es el caso de no poder aplicar el flujo de authorization code.

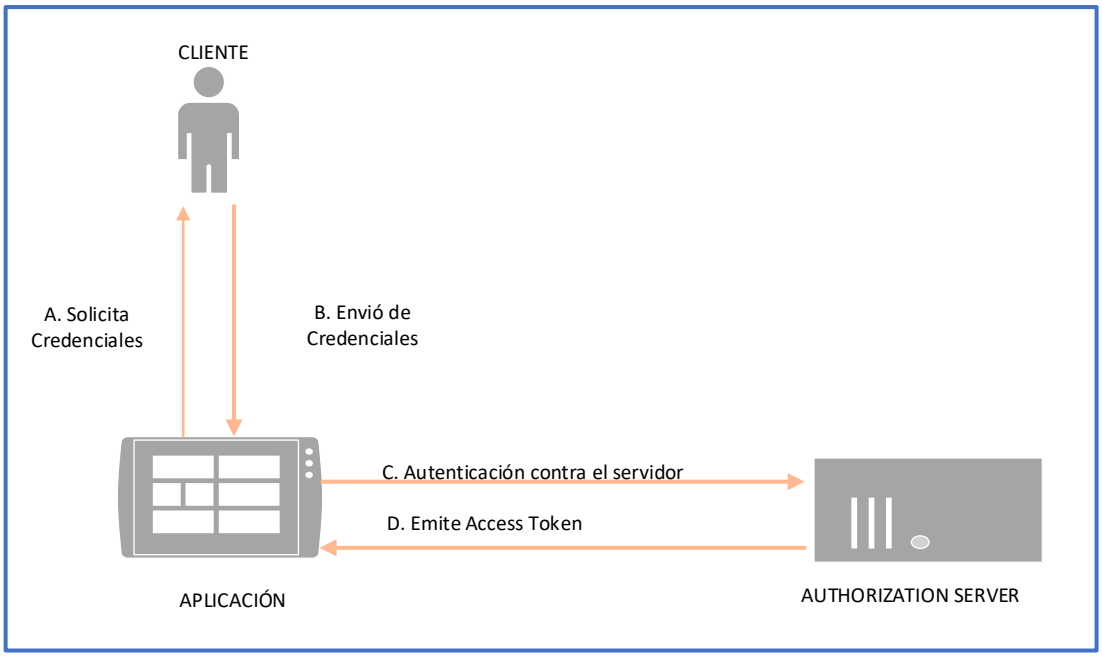

Figura 4: Password Credentials

a. Se solicita las credenciales del usuario.

b. Como la aplicación es de confianza el usuario proporciona las credenciales sobre la aplicación, delegando el poder autenticarse contra el servidor

c. La aplicación se autentica en nombre del usuario con las credenciales y se autentica contra el servidor

d. El servidor de autenticación devuelve el access token con su respectivo refresh si lo requiere.

Los parámetros que se envían en este flujo son los siguientes:

grant_type=password
&**username**=xxxxxxxxxxx
&**password**=xxxxxxxxxxx
&**client_id**=213f6de8-g987-4854-6r34-80a9b385cca7
&**client_secrect**=nH7TbHkgsjOWfgfhe45QD5EOtOH16nIusaVzZ4EI=
&**scope**=Write

Si los parámetros enviados son los correctos retornaría la siguiente información:

Content-Type: application/json
{
"access_token" : "xxxxxxx",

"token_type" : "Bearer",
"expires_in" : 3600,
"refresh_token" : "xxxxxxx",
"scope" : "Write"
}

El servidor de autorización después de validar todas las credenciales devuelve el access token, para este flujo necesita un front end y un banck end.

1.3.3.4 Client Credentials

Para este flujo la aplicación pertenece al usuario por lo que no es necesario que se autentique es decir el access token que envía el servidor se obtiene autenticando solo a la aplicación y no al usuario [6].

La diferencia de los demás flujos es que no necesita la interacción del usuario en el proceso por lo que la aplicación actúa en su propio nombre y no puede actuar en nombre del usuario.

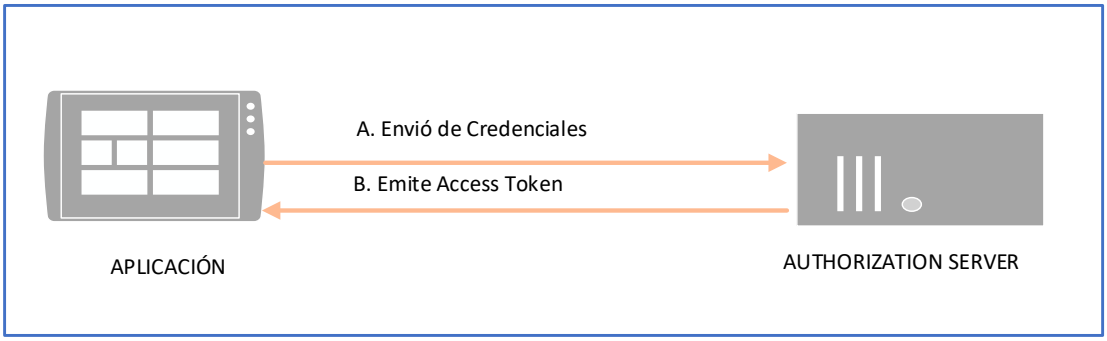

Figura 5: Client Credentials

a. La aplicación envía las credenciales al servidor de autenticación.
b. El servidor de autenticación revisa las credenciales enviadas para comprobar que los datos enviados por la aplicación sean la del usuario en caso de ser correcta emite el access token.

   Para este flujo los datos que son enviados por la aplicación son los siguientes:

   grant_type=client_credencials
   &**client_id**=213f6de8-g987-4854-6r34-80a9b385cca7
   &**client_secrect**=nH7TbHkgsjOWfgfhe45QD5EOtOH16nIusaVzZ4EI=
   &**scope**=Write

La respuesta seria la siguiente:

```
Content-Type: application/json
{
  "access_token" : "xxxxxxx",
  "token_type" : "Bearer",
  "expires_in" : 3600,
  "refresh_token" : "xxxxxxx",
  "scope" : "Write"
}
```

1.3.4 Registro de la Aplicación

Antes de utilizar OAuth 2, se debe registrar la aplicación para poder ocupar el estándar, esto se lo hace a través de un registro por parte del desarrollador o por API que debe proporcionar la siguiente información:

- Nombre de la Aplicación
- Callback URL - Redirect URI

Callback es donde el servidor regresara al usuario la respuesta después de que se autorice o deniegue la solicitud.

Cuando la aplicación se encuentre registrada, se emitirá las credenciales del cliente, es decir, se creará el Cliente id y el Client Secret. Client Id es una cadena que permite identificar a la aplicación y que se utiliza para generar la URL de autorización, mientras que el Client Secret se utiliza para autenticar a la identidad.

1.3.5 Errores de Respuesta

Si la solicitud falla, puede deberse a que la URI de redirección es incorrecta o inválida, el servidor de autorización debe informar al propietario del recurso, pero no debe redirigir automáticamente al agente de usuario [7].

A continuación, se presentan los errores que pueden provocar los flujos de autenticación:

- **Invalid_request:** Se produce cuando en la solicitud falta un parámetro requerido, incluye un parámetro no valido o incluye un parámetro más de una vez.
- **Unauthorized_client:** Sucede cuando el cliente no está autorizado a solicitar

un código de autorización.

- **Access_denied:** El servidor de autorización o el propietario del recurso rechazaron la solicitud.
- **Invalid_scope:** El scope solicitado es inválido, desconocido o está mal formado.
- **Server_error:** El servidor de autorización encontró una condición inesperada que no le permite cumplir con lo solicitado.
- **Temporarily_unavailable:** Este error se produce cuando el servidor de autorización no puede manejar la solicitud debido a una sobrecarga o mantenimiento temporal del servidor.

1.3.6 Problemas en el protocolo OAuth - OAuth2

Como se mencionó anteriormente OAuth surge para la protección de datos personales, pero su protocolo no pudo garantizar la seguridad. En abril del 2009 descubrieron que existían lagunas de seguridad al momento de la autenticación del protocolo. Entre abril y mayo de 2017, millones de usuarios fueron perjudicados por ataques de phishing a través de OAuth. Estos ataques consistían en correos fraudulentos que solicitaban autorización para acceder a información de la cuenta mediante una interfaz falsa, presentada como 'Google Apps'.

1.3.7 Consideraciones de Seguridad

Al querer utilizar OAuth2 debe seguir una serie de medidas y recomendaciones para poder implementar de manera segura el protocolo.

- El servidor de autenticación debe requerir el registro de cualquier URI de re direccionamiento para recibir la respuesta de autorización, además debe utilizar cualquier otro medio para proteger los recursos contra usuarios potencialmente maliciosos.
- El servidor de autenticación no debe procesar automáticamente las solicitudes de autorización repetidas sin que el cliente sea autenticado.
- El Access token debe mantenerse confidencial al ser enviado y almacenado y solo debe ser compartido entre el servidor de autorización, servidores de

recursos para los que es válido el access token y el cliente para el que se emite el access token.

- Las credenciales del Access Token solo deben ser transmitidas utilizando TLS.
- Refresh token debe mantenerse como confidencial en tránsito y almacenamiento y solo deben compartirse entre el servidor de autorización y al cliente que se emitió el Refresh token.
- El servidor de autenticación debe mantener el enlace entre el Refresh token y con el cliente al que se emitió. El medio para transmitir un Refresh token deben ser únicamente usando TLS.
- Los scope no deben incluir información confidencial del cliente o del propietario del recurso en texto plano, ya que pueden ser enviados por canales inseguros o ser almacenados de forma insegura.
- El cliente debe validar el certificado TLS del servidor de autenticación para evitar ataque ataques man in the middle.
- El servidor de autorización debe evitar ataques que adivinen access token, códigos de autorización, refresh token, contraseñas del resource owner y credenciales del cliente. Es por eso que los tokens deben ser generados a menor o igual 2^{-128} y recomendablemente inferior o igual 2^{-160}.
- El servidor de autenticación deben implementar protección contra CSRF(Cross-Site Request Forgery) para endpoint de autorización y garantizar que algún cliente malicioso no pueda obtener autorización sin consentimiento y conocimiento explícito del resource owner.
- Para evitar un ataque de Code Injection el servidor de autenticación y el cliente debe desinfectar y validar cuando sea posible cualquier valor recibido y en particular el valor de los parámetros state y redirect_uri.

1.4 OpenId Connect

Es una capa de identidad simple sobre el protocolo de OAuth2 que permite a los clientes verificar la identidad del usuario final basándose en la autenticación

realizada por un servidor de autorización, permitiendo obtener información de perfil básico sobre el usuario final. OICD implementa la autenticación como una extensión del proceso de autorización de OAuth2, los clientes solicitan el uso de OICD al incluir el valor del alcance de opeinid en la solicitud de autorización, la información sobre la autenticación realizada devuelve un JSON Web Token (JWT) denominado ID Token. [8]

OICD es pensado para autenticación y aumenta las funcionalidades de OAuth para la autorización, las cuales son:

- ID token permite saber quién es el usuario.
- Endpoint Userinfo que permite recuperar más información del usuario.
- Scopes estándar.
- Conjunto de claims que permiten obtener más datos.

1.4.1 ID Token

Id token se solicitan a través del protocolo OAuth2, es un token de seguridad que contiene claims sobre la autenticación de un usuario final por parte de un servidor de autorización, cuando se usa un cliente y otros claims solicitados, el id token se representa como un JWT.

Los flujos que se pueden solicitar un Id Token son los siguientes:

Authorization Code: Cuando se utiliza este flujo los access token se devuelven desde el endpoint de token. El flujo de Authorization code devuelve un código de autorización al cliente que luego puede intercambiarse por un id token y un access token. El realizar este proceso, proporciona beneficios ya que no expone ningún token al agente de usuario y posiblemente a otras aplicaciones maliciosas. El servidor de autenticación también puede autenticar al cliente antes que intercambie el código de autorización por un access token. Este flujo puede ser adecuado para los clientes que pueden mantener de forma segura un client secret entre ellos y el servidor de autorización.

Implicit flow: Al utilizar este flujo los tokens se devuelven desde el endpoint de autorización y no se utiliza el endpoint token, este flujo es utilizado por los clientes implementados en un navegador que utilizan un lenguaje de programación.

El id token y el access token se devuelve directamente al cliente que puede exponerlo al usuario final y las aplicaciones que tienen acceso al agente de usuario.

Al obtener un id token por medio de open id Connect, proporcionara información sobre el usuario del cual se autorizó, la información retorna en formato JWT, que es un estándar que pretende transmitir de manera segura la información entre dos partes en formato JSON y puede ser verificada porque está firmada digitalmente.

La estructura que tiene el token se presenta en tres partes separadas por puntos que puede ser similar a:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.**eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ**.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Cada información representa lo siguiente:

- **HEADER:** es la cabecera que contiene dos partes que es el tipo de token y el algoritmo que se está utilizando para ser firmado.
- **PAYLOAD:** contiene los claims que se pueden definir como los derechos que tiene un usuario; al igual otros tipos de derechos adicionales, el Payload se encuentra en Base64 y es la segunda parte del token.
- **Signature:** es utilizada para verificar que el mensaje no ha sido modificado durante él envió; el token compara las firmas y si son iguales, el token no ha sido modificado.

EL id token descifrado se vería de la siguiente manera:

```
{(HEADER)
"typ": "JWT",
"alg": "RS256",
"kid": "default_ssl_key"
}.
{(PAYLOAD)
   "iss": "https://accounts.example.com",
   "aud": "s7HktUspc3",
   "exp": 1311280970,
   "iat": 1311281970,
```

"auth_time": 1311280969,
}.
(ASIGNATURE)

1.4.2 Userinfo

Es un recurso protegido de OAuth2 que devuelve claims al cliente, mostrando información de perfil de usuario previamente consentida. Userinfo puede estar codificado en un JSON y además puede estar empaquetado en un JWT firmado o cifrado, el cliente realiza una solicitud al endpoint de userinfo utilizando un access token obtenido por medio de la autenticación de openid connect. [9]

La respuesta se visualizaría de la siguiente manera:

```
{
  "sub": "248289761001",
  "name": "xxxxxx",
  "given_name": "xxxxx",
  "family_name": "xxxxx",
  "email": "yyyyy@example.com"
}
```

1.4.3 Scopes

Utilizados por una aplicación durante la autenticación para autorizar el acceso a los detalles de un usuario, como por ejemplo nombre, foto, ids. Cada ámbito devuelve un conjunto de atributos de usuario que se llaman claims. Las solicitudes de open id Connect deben contener el valor de scope de opein id, para identificar que la aplicación pretende utilizar OIDC para verificar la identidad del usuario.

1.4.4 Claims

Son destinadas para proporcionar a una aplicación detalles del usuario, como nombre, correo, etc. y se encuentran predefinidas para el protocolo OIDC, estos claims se devuelven en un id token y también se encuentran disponibles a través de userinfo endpoint.

1.5 Autenticación mediante Certificado

La autenticación mediante certificados, es un método de seguridad que utiliza certificados digitales para verificar la identidad de una entidad, esta entidad puede ser una persona, un dispositivo o una aplicación. Este proceso se basa en la

criptografía de clave pública (PKI) y está emitido por una Autoridad Certificadora (CA), lo que garantiza una comunicación segura y autenticada. [10]

Los componentes de un certificado son:

- **Claver pública:** Se utiliza para cifrar información y se incluye en el certificado.
- **Clave privada:** Clave mantenida en secreto por el propietario del certificado, se utiliza para descifrar la información cifrada con la clave pública.
- **Firma digital:** Es generada por la CA (Autoridad Certificadora), asegura que el certificado no ha sido alterado y que fue emitido por una CA confiable.
- **Autoridad Certificadora (CA):** Entidad que confianza que emite y gestiona los certificados digitales.

1.5.1 Proceso de autenticación

El uso de certificado digitales abarca diversos escenarios en los que pueden proporcionar autenticación, integridad y cifrado.

- **Autenticación de usuario:** se utiliza para conectarse a un sistema o a una red, para lo cual presenta un certificado como prueba de su identidad, para esto lo primero que hace es presentar el certificado, lo cual envía al servidor, luego se valida la firma del certificado utilizando la clave pública de la CA, esto asegura que el certificado sea autentico y por último el servidor valida la información del certificado como la identidad del usuario y la validez.
- **Autenticación de dispositivos:** el certificado puede ser utilizado en dispositivos ya sea computadoras o móviles de alguna empresa para ser autenticados, esto se utiliza en entornos de autenticación de red.
- **Autenticación en servicios web y Apis:** El cliente que puede ser una aplicación o servicio presenta un certificado para acceder a un servicio web, luego el servidor verifica ese certificado para asegurar que tengan los permisos para acceder al servicio solicitado.

CAPÍTULO 2. Estructura - Metodología De Implementación

A continuación, se detalla el proceso de implementación de autenticación mediante OAuth, utilizando para ello una herramienta denominada API Gateway, que facilita la aplicación de estas metodologías.

Un API Gateway es un componente arquitectónico crucial que opera como un punto de entrada único para todas las solicitudes que se dirigen a un sistema basado en micro servicios. Su función principal es gestionar y enrutar estas solicitudes hacia los servicios backend adecuados, mientras maneja múltiples aspectos clave como la autenticación, el control de acceso, el balanceo de carga y el monitoreo del rendimiento. Al centralizar y consolidar estas funciones, un API Gateway optimiza la interacción entre los clientes y los diversos servicios dentro de un entorno distribuido, garantizando una administración más eficiente y segura del tráfico de datos.

2.1 Definición y concepto de Api Gateway

Un API Gateway actúa como intermediario entre las aplicaciones cliente y los servicios de backend en una arquitectura de microservicios. Funciona como un único punto de acceso para diversas API, gestionando tareas clave como la composición de solicitudes, el enrutamiento y la conversión de protocolos. Su propósito es regular el tráfico de las API y aplicar medidas de seguridad, facilitando así la gestión mediante un control centralizado. Esto permite a los desarrolladores enfocarse en los servicios individuales sin preocuparse por las complejidades de la red de API, incluyendo funciones como la autenticación de usuarios y la limitación de velocidad. [11]

El API Gateway puede entenderse como un servidor que se sitúa entre los clientes y los servicios de backend, actuando como un punto de entrada para el acceso de los clientes a estos servicios. Su función principal es ocultar la arquitectura interna del sistema y ofrecer una API unificada que se adapte a las necesidades específicas de cada cliente.

El API Gateway también se encarga de tareas clave, como enrutar las solicitudes a los servicios de backend correspondientes, proporcionar una capa de seguridad para la API, gestionar el acceso a los servicios, equilibrar la carga entre

diferentes servicios, y realizar otras funciones esenciales.

Asimismo, el API Gateway puede incorporar funcionalidades adicionales, como la autenticación y autorización de usuarios, el almacenamiento en caché de las respuestas de la API, y la conversión de protocolos y formatos de datos. [11]

2.2 Topología de Api Gateway

La topología de un API Gateway se refiere a la estructura y disposición de sus componentes dentro de una arquitectura de sistemas. Esta topología puede diferir según los requisitos específicos de la aplicación, el volumen de tráfico anticipado y los requisitos de escalabilidad. A continuación, se presenta una visión general de las arquitecturas comunes de un API Gateway: [11]

2.2.1 Topología Básica

En su forma más simple, un API Gateway actúa como un punto único de entrada para todas las solicitudes. Esta topología incluye los siguientes componentes:

- **Cliente:** Entidad que realiza una solicitud al API.
- **Api Gateway:** Recibe todas las solicitudes de los clientes y las direcciona a los servicios de backend correspondientes.
- **Servicios Backend:** Servicios o microservicios que realizan las normativas de operación o procesan datos y responden a las solicitudes del Api Gateway.

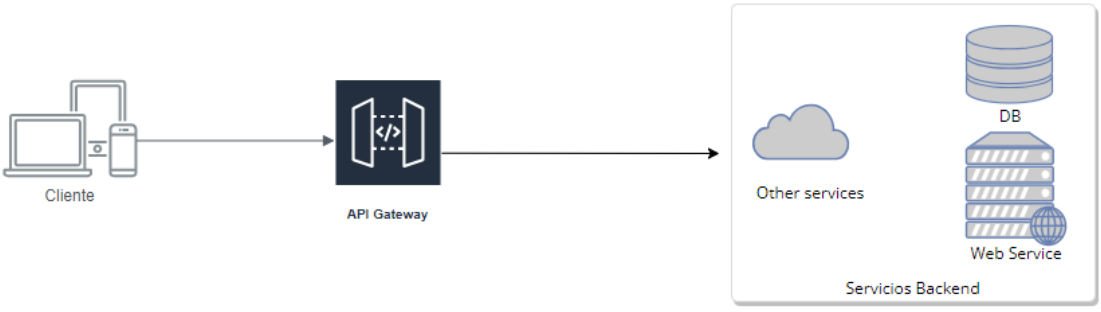

Figura 6: Topología Básica

2.2.2 Topología con funciones de seguridad

Api Gateway permite agregar módulos adicionales para manejar funciones

específicas como seguridad, transformación de datos y cache.

- **Cliente**
- **Api Gateway:** Se puede agregar los siguientes módulos
    - **Enrutador:** Redirige las solicitudes a los servicios o micro servicios adecuados.
    - **Seguridad:** Gestiona la autenticación y autorización.
    - **Transformación de datos:** Puede realizar modificaciones en los datos solicitados como en la respuesta.
    - **Cache:** Almacena respuestas para mejorar el rendimiento.
- **Servicio Backend.**

Figura 7: Topología con funciones de seguridad

2.2.3 Topología con balanceo de carga y alta disponibilidad

Permite manejar grandes volúmenes de tráfico y asegura la alta disponibilidad, permite implementar varias instancias del Api Gateway detrás de un balanceador de carga.

- **Cliente.**
- **Balanceador de Carga:** Distribuye las solicitudes entre las instancias de Api Gateway.
- **Instancias de Api Gateway:** Múltiples instancias que manejan las solicitudes y las enrutan a los backend.
- **Servicios de Backend**

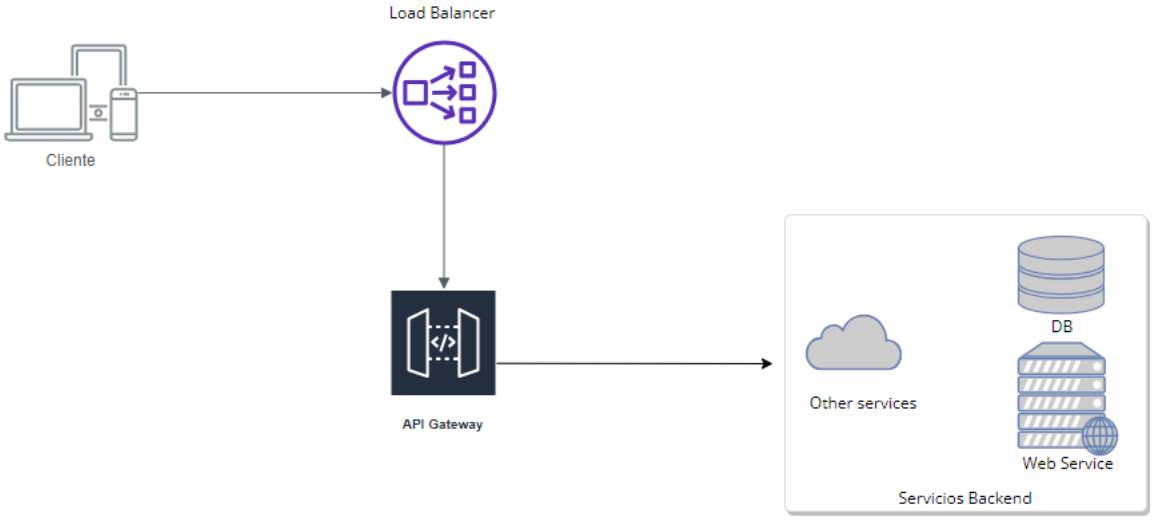

Figura 8: Topología con balanceo de carga y alta disponibilidad

2.2.4 Topología con agregación de servicios

Api Gateway puede realizar la agregación de datos de múltiples servicios backend en una sola respuesta.

- **Cliente**
- **Api Gateway**
  - **Enrutador**
  - **Módulo de Agregación:** combina respuesta múltiple.
- **Servicios Backend**

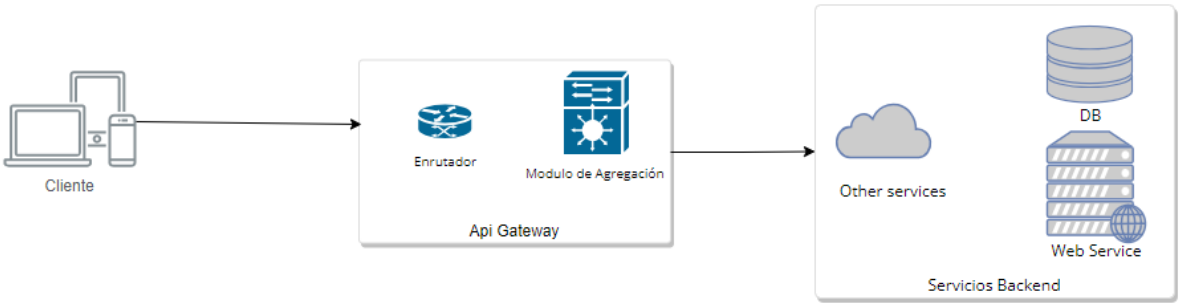

Figura 9: Topología con agregación de servicios

2.2.5 Topología con microservicios internos

El Api Gateway puede tener interacciones más complejas con otros micros servicios dentro de la misma infraestructura, manejando la comunicación entre

servicios y el enrutamiento interno.

- **Cliente.**
- **Api Gateway.**
    - **Enrutador.**
    - **Módulo de comunicación interna:** Facilita la comunicación entre microservicios.
- **Microservicios internos:** Servicios que pueden interactuar entre si además lo pueden hacer con el Api Gateway.

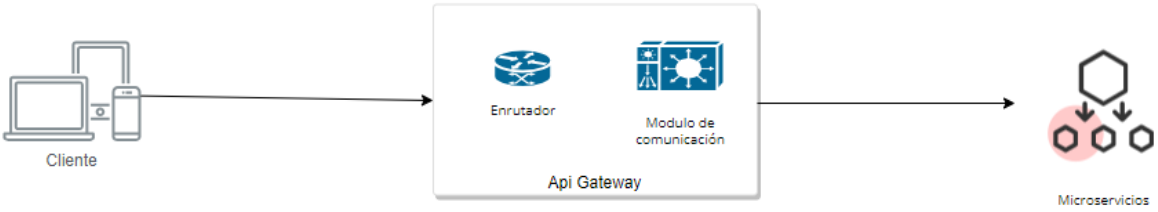

Figura 10: Topología con microservicios internos

2.3 Funcionalidades de Api Gateway

Agregar un Api Gateway a la infraestructura de una organización puede brindar una capa de seguridad que es fundamental para proteger a los servicios, microservicios o Apis y los sistemas backend asociados de la empresa.

2.3.1 Autenticación y Autorización

Mediante esta funcionalidad el api Gateway permite que los usuarios y aplicativos autorizados puedan acceder a los servicios de la empresa, agregándoles el nivel de acceso apropiado, esto ayudaría a mejor la seguridad y el control sobre los servicios. [11]

- **Autenticación:** Lo que realiza el Api Gateway es la verificación de la identidad de usuarios o aplicaciones que hacen las solicitudes, lo cual pueden utilizar los siguientes métodos:
    - **Proveedores de identidad:** Pueden ser otras herramientas que se integran con el Api Gateway que pueden ser LDAP o algún gestor en la nube que gestione el acceso.

- **Token JWT:** Este método valida la identidad y que permisos tiene el usuario mediante un token que es cifrado, debido que contiene información sobre el usuario y sus permisos.
- **Oauth 2.0:** Permite el uso de un token de acceso que autoriza usuarios o aplicaciones sin la necesidad de compartir las credenciales directamente.

- **Autorización:** Después que el usuario o aplicativo fue autenticado el Api Gateway controla que recursos son los acreditados, lo realiza de la siguiente manera:
  - **Control de acceso mediante roles:** El Api Gateway define que permisos tiene el usuario según el rol asignado.
  - **Control de acceso mediante atributos:** Se aplican políticas de acceso con más detalle para esto se basa mediante atributos del usuario, recursos o entonos.

Para tener mayor detalle de la autenticación se puede revisar el Capítulo 1 sección OAuth.

2.3.2 Control de Acceso

El control de acceso asegura que usuarios o aplicativos están autorizados para que interactúen con los recursos y servicios, por lo cual el Api Gateway puede utilizar los siguientes mecanismos para agregar el control de acceso:

- **Listas de control de acceso (ACLs)**: Las listas son políticas que determinan que usuarios o grupos de usuarios puede acceder a ciertos recursos, por lo cual el Api Gateway puede aplicar esas políticas para el control de acceso a diferentes endpoints.

  Mediante las listas de control de puede configurar permisos detallados que pueden utilizar ciertos métodos HTTP que podrían ser GET, POST, etc. Así como también indicaría que recursos son los que están accesibles para cada usuario o grupo.

- **Reglas basadas en IP:** El Api Gateway puede ser configurado para denegar o permitir solicitudes basadas en la IPs de origen, esto ayudara a restringir el

acceso a ciertas IPs conocidas o bloquear algunas IPs sospechosas.

Mediante este mecanismo se puede implementar allowlists o blocklists, que permitirán o deniegan el acceso a ciertas IPs específicas.

2.3.3 Monitoreo y Registro

El monitoreo y registro es un aspecto fundamental en la seguridad y el mantenimiento de un Api Gateway, estos mecanismos permiten al administrador tener una visión del tráfico de las Apis, permitiendo detectar si existe problemas o realizar alguna auditoria de seguridad, a continuación, se detalla este mecanismo:

- **Registro de Actividades:** El API Gateway captura y almacena detalles sobre cada solicitud y respuesta, incluyendo URL, método HTTP, estado de la respuesta, tiempos de procesamiento, IP del cliente, encabezados HTTP, parámetros, cuerpos de solicitud y respuesta, y códigos de estado HTTP. También se registran eventos de seguridad, como fallos en autenticación o patrones sospechosos, para identificar posibles amenazas.

  Estos registros son cruciales para la auditoría y cumplimiento normativo, permitiendo a los auditores verificar el acceso y las acciones realizadas, lo que ayuda a cumplir con regulaciones como GDPR o HIPAA.

- **Monitoreo de Métricas:** El API Gateway recopila y muestra métricas en tiempo real sobre el tráfico de las API, incluyendo el número de solicitudes, tiempos de respuesta y uso de ancho de banda, datos esenciales para comprender el flujo de tráfico y detectar posibles anomalías.

  Es posible configurar alertas basadas en ciertos umbrales, como aumentos inesperados en el tráfico o en las tasas de error. Estas alertas pueden enviarse por correo electrónico, SMS o integrarse con sistemas de gestión de incidentes. Además, se emplean herramientas de análisis para examinar el tráfico, identificar patrones de uso y planificar la capacidad futura. Plataformas de análisis y visualización como Grafana o Kibana son utilizadas para estos fines.

2.3.4 Seguridad en el despliegue y configuraciones de un Api Gateway

Permite garantizar que el sistema sea resistente a ataques y funcione de

manera segura desde el momento que se pone en marcha, a continuación, se detalla un poco más sobre este mecanismo.

- **Configuración Segura:** La configuración segura del API Gateway debe seguir el principio de seguridad por defecto, aplicando configuraciones seguras de manera predeterminada para minimizar exposiciones. Esto incluye desactivar funcionalidades no necesarias y restringir el acceso a configuraciones sensibles. Se deben definir y configurar políticas de seguridad claras para el acceso a las APIs, abarcando autenticación, autorización y cifrado, además de implementar políticas estrictas para contraseñas. También es crucial controlar las versiones de configuraciones y despliegues utilizando sistemas de control de versiones como Git, lo que facilita la auditoría y permite revertir cambios si es necesario.
- **Protección Contra Configuraciones Incorrectas:** Es fundamental realizar validaciones de configuración para evitar vulnerabilidades y brechas de seguridad. Se deben usar herramientas y técnicas para asegurar que las configuraciones sigan las mejores prácticas de seguridad. Además, es necesario llevar a cabo revisiones regulares de la configuración del API Gateway para mantener la alineación con las políticas de seguridad y corregir posibles errores. Es recomendable combinar revisiones manuales con auditorías automatizadas para detectar y solucionar problemas de configuración de manera efectiva.
- **Gestión de Incidentes y Respuesta:** Elaborar y mantener un plan de respuesta a incidentes específico para el API Gateway, asegurando que el equipo de seguridad esté preparado para gestionar cualquier problema relacionado con él. Llevar a cabo simulaciones de incidentes permite evaluar tanto la efectividad del plan como la capacidad del equipo para reaccionar ante situaciones de seguridad. Es esencial implementar procedimientos de recuperación que permitan restaurar rápidamente el API Gateway a un estado seguro en caso de incidente, además de mantener copias de seguridad periódicas de configuraciones y datos críticos para facilitar dicho proceso de recuperación.

2.4 Casos de Estudio

Api Gateway es una herramienta esencial en la arquitectura de servicios, microservicios y sistemas distribuidos, podríamos decir que su propósito principal es el actual como un punto de entrada único para las diferentes solicitudes de los clientes o usuarios, en otras palabras, un Api Gateway maneja la comunicación entre el cliente y los distintos servicios que componen una aplicación.

2.4.1 Caso de Estudio: Aplicación Financiera

Una entidad financiera al tener aplicaciones, tiene requisitos específicos en cuanto seguridad, rendimiento y escalabilidad, a continuación, se detalla más sobre este caso.

- **Componente de una aplicación:** los componentes básicos son los siguientes que incluyen:
    - **Servicio de autenticación y autorización:** Gestiona la autenticación del cliente o usuario y el acceso, implementación de autenticación mediante múltiples factores y protege las credenciales.
    - **Servicio de gestión de transacciones:** Supervisa y documenta las transacciones efectuadas por el cliente, garantizando la integridad y el orden adecuado de cada transacción.
    - **Servicio de gestión de cuenta:** Gestiona la información de la cuenta del cliente que incluye transacciones, saldos y detalles de la cuenta.
    - **Servicios de notificaciones:** Envía al cliente notificaciones sobre su cuentan, los cuales pueden ser los movimientos que fueron efectuados, transacciones o alertas de seguridad.
- **Rol del Api Gateway:** El Api Gateway tiene un papel crucial ya que debe coordinar y manejar las solicitudes entre los servicios, a continuación, se detalla las funciones específicas.
    - **Enrutamiento de las solicitudes:** El Api Gateway es el que se encargar a dirigir las solicitudes entrantes a los servicios correspondientes, un claro ejemplo seria que una solicitud quiera verificar el saldo de una cuenta por lo que se tendría que dirigir a un

servicio de gestor de cuentas mientras que otra solicitud para realizar una transferencia se dirigiría al servicio de transferencia.

- **Autenticación y Autorización:** El Api Gateway se encarga en la autenticación inicial de las solicitudes, asegurándose que los clientes se encuentren correctamente autenticados antes que sus solicitudes lleguen a los servicios o microservicios correspondientes. Además, se puede manejar la autorización de acceso a distintos recursos que se basan en los permisos del cliente o usuario.
- **Seguridad:** El Api Gateway implementa medidas de seguridad, las cuales pueden ser la encriptación de los datos en tránsito y la protección contra los ataques DDoS (Denegación de servicios). Esta herramienta también puede integrar sistemas de detección de fraudes y la prevención de accesos no autorizados.
- **Monitoreo y Registro:** El Api Gateway puede centralizar métricas y recoger logs de trafico de las solicitudes, tiempos de respuesta y los posibles errores que se pueden presentar. Esto ayudar en la monitorización del rendimiento del sistema y las resoluciones de problemas.
- **Respuestas:** Puede existir solicitudes que requieran datos de múltiples servicios, por lo que el Api Gateway tiene la capacidad de combinar respuestas de varios microservicios en una sola respuesta.

- **Ejemplo de proceso de Transferencia de Fondos.**

  El usuario desea realizar una transacción de fondos a otra cuenta, se detalla cómo se realiza la transacción mediante el Api Gateway.

  - **Solicitud:** El cliente realiza una transacción a través de la web o aplicación móvil, la solicitud será envidad al Api Gateway.
  - **Autenticación:** El Api Gateway verificará la autenticación para que el cliente está correctamente autenticado, esta parte podrá utilizar un token JWT o podrá utilizar un servicio de autenticación.
  - **Autorización:** Después de ser autenticado, el Api Gateway verificara

que el cliente tenga los permisos suficientes para realizar la transacción y que tenga los fondos en su cuenta.

- **Enrutamiento de servicios:** El Api Gateway dirige la solicitud al servicio correspondiente en este caso de transacciones, la cual proporcionara la cuenta origen y destino, así como el monto a ser enviado.
- **Procesamiento de Transacción:** El servicio realiza la transacción de fondos, luego actualizara las cuentas y registrara la transacción en el servicio de transacciones.
- **Notificaciones:** El servicio de notificaciones enviara una confirmación de la transacción realizada al cliente.
- **Respuesta al cliente:** El Api Gateway combinara las respuestas del servició de transacciones como el de notificaciones enviando una respuesta consolidad al cliente.

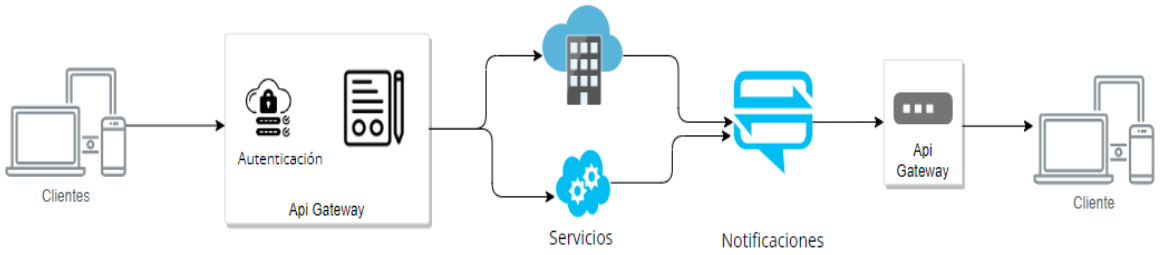

Figura 11: Ejemplo de proceso de Transferencia de Fondos.

CAPÍTULO 3. Estándares, Riesgos, Tendencias y Desafíos

La gestión de la seguridad informática no solo está regulada por buenas prácticas sino también por normas y estándares que aseguran la protección de datos y la privacidad, siguiendo esas normas podremos mitigar los riesgos que se van presentando día a día.

La inteligencia artificial (IA) y el aprendizaje automático (ML) están revolucionando la detección de amenazas y la automatización de respuestas. Sin embargo, surgen desafíos como la evolución de amenazas, la protección de datos en la nube, y la adaptación a nuevas regulaciones. Estos avances y desafíos configuran el futuro de la seguridad y el control de acceso, exigiendo una adaptación continua.

3.1 Estándares

En el contexto de la seguridad de la información, los API Gateway juegan un papel crucial al gestionar y dirigir el tráfico entre los clientes y los servicios backend. [12] “La norma ISO/IEC 27001 ofrece un marco completo para implementar y mantener un SGSI (sistema de gestión de seguridad de la información), que es aplicable a todos los elementos de la infraestructura tecnológica, incluidos los API Gateways”.

Para garantizar que un API Gateway opere de manera segura y conforme a los estándares internacionales, debe adherirse a varios controles especificados en la norma ISO/IEC 27001. Los controles que se destacan son la gestión de acceso (A.9), la seguridad de las comunicaciones (A.13), y la gestión de incidentes (A.16). Estos controles aseguran que el API Gateway proteja adecuadamente la información durante la transmisión, gestione de manera segura el acceso a los datos, y responda eficazmente a cualquier incidente de seguridad.

Asimismo, el cumplimiento de la norma implica la implementación de medidas para la gestión de riesgos (A.6), la protección de la infraestructura física (A.11), y el desarrollo y mantenimiento seguro de sistemas (A.14). Estos aspectos son fundamentales para proteger tanto el API Gateway como los servicios que gestiona, asegurando así una robusta protección de la información y conformidad con los requisitos legales y contractuales aplicables (A.18).

3.1.1 OWASP

En su página lo describe de la siguiente manera [13]“Es una fundación sin ánimo de lucro que no está afiliado con ninguna compañía de tecnología, su objetivo es determinar y combatir un software inseguro, es por eso por lo que produce varios materiales de diferentes tipos de manera abierta y colaborativa”.

OWASP ofrece recursos y directrices que, aunque no son normas oficiales como las de organismos de estandarización, se han convertido en estándares de facto en la industria de la seguridad de aplicaciones. [14]

3.1.1.1 Top 10 Año 2021

A01:2021-Broken Access Control: este punto hace referencia a las vulnerabilidades que permiten al usuario acceder a funcionalidades o recursos que no deberían tener acceso o deberían estar restringidas, esto sucede cuando un servicio o aplicativo no valida adecuadamente las autorizaciones del usuario permitiendo que interactúen con datos o funciones que no tienen permisos.

Los casos pueden ser los siguientes:

- **Violación de Roles:** Sucede cuando un usuario con rol básico puede acceder a funciones administrativas.
- **Elevación de privilegios:** Puede pasar cuando una persona no autorizada encuentra la manera de elevar sus permisos a través de técnicas como manipulación de datos en alguna solicitud.
- **Manipulación de URLs:** Sucede cuando el usuario puede manipular algún parámetro en la URL para acceder como otra persona.

Para mitigar estos problemas, se recomienda la implementación de un control de acceso basado en roles, esto debe aplicarse en las restricciones del servidor y no solo en el usuario, además es importante revisar y auditar los permisos y acceso regularmente.

A02:2021-Cryptographic Failures: antes llamado Sensitive Data Exposure (Exposición de Datos Sensibles) este punto se centra en las vulnerabilidades que exponen la divulgación no autorizada de información confidencial, esta fuga puede incluir contraseñas, información personal identificable, datos financieros y datos de

salud. La causa de este tipo de exposición se da porque no existe un cifrado adecuado, configuraciones inseguras, almacenamiento inadecuado y errores en el código.

Algunos casos pueden ser:

- **Fuga de datos personales:** Exposición de datos personales en la configuración de servicios, esto permitiría el acceso público a datos sensibles.
- **Filtración de datos financieros:** Exposición de datos de tarjetas de crédito, se ha visto en los últimos años se han publicado millones de números de tarjeta debido a un cifrado inadecuado.

Para poder mitigar este tipo de problemas, se recomienda cifrar los datos que se encuentren en tránsito y también los que están en reposo, gestionar las claves criptográficas de manera segura, validar y filtrar adecuadamente los datos para prevenir exposiciones de forma accidental, restringir el acceso a datos sensibles.

A03:2021-Injection: Sucede cuando una aplicación permite insertar datos no confiables en una consulta o comando que luego pueden ser ejecutados, esto puede permitir que el atacante ejecute código malicioso o manipular consultas de manera inesperada.

Este tipo de vulnerabilidad pueden llevar a exponer los datos sensibles, así como a la modificación o eliminación de datos, esto comprometería la seguridad del sistema.

Existe varios tipos que son:

- **XML Injection:** Sucede cuando la estructura XML es alterado para modificar el comportamiento de la aplicación.
- **SQL Injection:** Las consultas SQL son manipuladas para acceder o modificar datos no autorizados.
- **LDAP Injection:** Las consultas LDAP son manipuladas para modificar o exponer datos de directorio.

Se puede prevenir de la siguiente manera:

- Uso de bibliotecas de seguridad para manejar de forma segura las consultas.
- Realizar pruebas de penetración y análisis para corregir vulnerabilidades existentes en el código.
- Se debe limpiar los permisos de las cuentas de la base de datos.

A04:2021- Insecure Design: La vulnerabilidad por diseño inseguro, surge cuando no se considera la seguridad durante la etapa de diseño de la aplicación. Aunque las medidas de seguridad se apliquen adecuadamente en el código, un diseño defectuoso de la aplicación puede crear vulnerabilidades que no se pueden resolver completamente solo con ajustes en el código.

Un diseño seguro es crucial, porque trata las vulnerabilidades en una fase temprana del ciclo de vida del desarrollo de software. Arreglar problemas de seguridad en etapas posteriores, como en la codificación o en las pruebas, suele ser más costoso y complicado. Un diseño adecuado puede disminuir notablemente la superficie de ataque y ofrecer una mejor protección contra diversas amenazas.

Causas que puede provocar este tipo de vulnerabilidad:

- Incluir funcionalidades sin tener en cuenta cómo podrían ser aprovechadas por un atacante.
- Utilizar componentes de terceros sin evaluar su seguridad ni considerar cómo se integran con la aplicación.
- Omitir un análisis de riesgos adecuado durante la fase de diseño puede resultar en lagunas críticas en la seguridad.

Mejores prácticas de diseño:

- Incluye a expertos en seguridad desde las etapas iniciales del desarrollo para garantizar que la seguridad sea una parte esencial del diseño.
- Implementa varias capas de seguridad para proteger la aplicación, incluyendo controles de acceso, autenticación y cifrado.
- Forma a los diseñadores y arquitectos de software en principios de diseño seguro y en la aplicación de buenas prácticas de seguridad.

A05:2021- Security Misconfiguration: La configuración de seguridad

incorrecta se produce cuando un sistema, aplicación o infraestructura no está configurado adecuadamente para protegerse contra ataques. Esto puede ser causado por configuraciones predeterminadas inseguras, falta de actualizaciones o ajustes incorrectos que exponen datos o funcionalidades sensibles.

Causas comunes:

- Exponer servicios o interfaces que no son necesarios para el funcionamiento de la aplicación puede ampliar la superficie de ataque.
- Muchas aplicaciones y servidores incluyen configuraciones predeterminadas que pueden no ser seguras, como contraseñas por defecto o permisos extensos.
- Exponer o manejar de manera incorrecta variables de entorno que contienen información sensible o credenciales.

Mejores prácticas:

- Asegurarse de que los permisos sean lo más restrictivos posible y ajusta los accesos para que solo los usuarios y servicios necesarios cuenten con los permisos adecuados.
- Asegurarse de que todos los componentes de infraestructura (servidores, bases de datos, redes) estén configurados teniendo en cuenta la seguridad.
- Modificar las configuraciones predeterminadas a valores más seguros y desactiva las funciones que no se utilizan.

A06:2021- Vulnerable and Outdated Components: Las aplicaciones suelen utilizar bibliotecas, frameworks y otros componentes de terceros para acelerar el desarrollo y añadir funcionalidades. Sin embargo, si estos componentes tienen vulnerabilidades conocidas o que no se actualizan regularmente y pueden convertirse en un vector de asalto para los atacantes.

Causas comunes:

- Las bibliotecas pueden depender de otras bibliotecas que también podrían contener vulnerabilidades.
- Usar versiones antiguas de bibliotecas o frameworks que tienen

vulnerabilidades conocidas.

- No estar informado sobre las actualizaciones de seguridad o las vulnerabilidades de los componentes utilizados

Mejores prácticas:

- Evaluar la seguridad y la reputación de los componentes antes de integrarlos en tu aplicación. Asegúrate de que se mantengan activamente y que tengan un historial de seguridad sólido.
- Optar por utilizar componentes de fuentes confiables y con buena reputación. Revisar las políticas de seguridad y los informes de vulnerabilidad de estos componentes.
- Emplear herramientas y servicios para monitorear y gestionar las vulnerabilidades en los componentes de software. Herramientas como Snyk, Dependabot y OWASP Dependency-Check pueden ayudar a identificar componentes vulnerables.

A07:2021- Identification and Authentication Failures: Las fallas en la identificación y autenticación, ocurren cuando una aplicación no protege adecuadamente el proceso de identificación (quién eres) y la autenticación (verificación de identidad) de los usuarios. Esto puede permitir a los atacantes acceder a recursos o realizar acciones que deberían estar restringidas.

Causas comunes:

- Procedimientos de recuperación de contraseñas inseguros que permiten a los atacantes restablecer contraseñas sin una verificación adecuada.
- Mensajes de error que divulgan información sobre los mecanismos de autenticación y pueden ser usados para realizar ataques.
- Uso de contraseñas que son fáciles de adivinar o que no están adecuadamente protegidas mediante técnicas como cifrado o hashing seguro.

Mejores prácticas:

- Verificar que los procedimientos de recuperación de contraseñas sean seguros y requieran una verificación adecuada antes de permitir el

restablecimiento de la contraseña.

- Verificar que los controles de acceso estén correctamente implementados y que los usuarios solo puedan acceder a los recursos y funciones a los que están autorizados.
- Implementar monitoreo y registro de eventos de autenticación para detectar posibles intentos de acceso no autorizado o actividades sospechosas.
- Establecer políticas de contraseñas que exijan combinaciones de caracteres y eviten contraseñas débiles. Asegurarse de almacenar las contraseñas utilizando técnicas seguras como el hashing.

A08:2021- Software and Data Integrity Failures:

Este tipo de riesgo sucede por fallas de integridad del software y los datos de la aplicación. Esto puede abarcar problemas como la alteración del código, la inserción de código malicioso y la modificación de datos.

Causas comunes:

- No usar firmas digitales para asegurar que el software y las actualizaciones provengan de una fuente confiable y no hayan sido modificados.
- Procedimientos poco seguros en la distribución de software y actualizaciones que pueden facilitar la introducción de código malicioso.
- Emplear componentes de terceros sin verificar su integridad o sin aplicar controles adecuados para su protección.

Mejores prácticas:

- Aplicar controles de integridad durante el desarrollo y la implementación para salvaguardar el código y los datos contra alteraciones no autorizadas.
- Utilizar únicamente componentes y bibliotecas de fuentes confiables y verifica su integridad antes de integrarlos en tu aplicación.
- Aplicar cifrado robusto para proteger los datos tanto en tránsito como en reposo, y asegúrate de que los mecanismos de cifrado sean sólidos y estén actualizados.

- Verifica que el software y las actualizaciones estén firmados digitalmente para asegurar su autenticidad y la integridad de los componentes.

A09:2021- Security Logging and Monitoring Failures:

Contar con un sistema de registro y monitoreo adecuado es crucial para la seguridad de una aplicación, ya que ofrece visibilidad sobre las actividades y eventos dentro del sistema. Sin registros detallados y monitoreo en tiempo real, puede resultar complicado detectar, comprender y responder a incidentes de seguridad, lo que podría aumentar la exposición y el daño potencial. Implementar buenas prácticas en estas áreas es esencial para mantener una postura de seguridad efectiva y cumplir con los requisitos de auditoría y cumplimiento.

Causas comunes:

- Los registros pueden incluir información sensible que debe ser protegida adecuadamente para prevenir la exposición de datos.
- No mantener los registros durante el tiempo necesario o almacenarlos de manera que no sean fácilmente accesibles o analizables.
- Fallar en registrar eventos críticos de seguridad, como intentos de acceso fallidos, cambios en configuraciones y operaciones con datos sensibles.

Mejores prácticas:

- Conservar los registros durante el tiempo requerido por las políticas de seguridad y cumplimiento, y asegúrate de que estén disponibles para análisis y auditorías.
- Garantizar que se registren eventos clave, como intentos de acceso, modificaciones en la configuración y operaciones con datos sensibles. Incluye detalles como la hora, el usuario, la IP de origen y la acción ejecutada.
- Verificar que los registros estén cifrados y solo sean accesibles para personal autorizado, para prevenir la exposición de información sensible.
- Configurar alertas para eventos críticos y define procedimientos para responder a incidentes de seguridad en función de los registros.

A10:2021- Server-Side Request Forgery (SSRF):

SSRF es una vulnerabilidad en la que un atacante manipula a un servidor para que envíe una solicitud HTTP a un destino que el atacante elige. Esto puede permitir al atacante acceder a servicios internos o restringidos que, de otro modo, no estarían disponibles para usuarios externos, como servicios internos de infraestructura, bases de datos o recursos en la nube.

Implementar adecuadamente prácticas de seguridad para prevenir SSRF ayuda a salvaguardar los sistemas y servicios internos contra accesos no autorizados y disminuye el riesgo de comprometer la seguridad general de la aplicación.

Causas comunes:

- La infraestructura de red no está suficientemente segmentada para proteger los servicios internos contra accesos no autorizados.
- La aplicación permite a los usuarios ingresar URLs o direcciones IP que el servidor utiliza para realizar solicitudes internas sin una validación adecuada.
- El servidor no filtra adecuadamente las solicitudes destinadas a destinos internos o restringidos.

Mejores prácticas:

- Resguarda los recursos internos mediante controles de autenticación y autorización adecuados para garantizar que solo los usuarios y servicios autorizados tengan acceso a ellos.
- Validar y sanitizar toda entrada de usuario que pueda usarse para realizar solicitudes, como URLs o direcciones IP, para prevenir que usuarios maliciosos especifiquen destinos internos.
- Configurar los servidores y servicios internos para que no sean accesibles desde el exterior. Emplea redes y políticas de firewall para restringir el acceso únicamente a las instancias que lo requieran.
- Implementar listas blancas de URLs o direcciones IP a las que el servidor puede enviar solicitudes, y bloquea las solicitudes a destinos no autorizados.

3.2 Riesgos

Las claves expuestas y los datos confidenciales constituyen una creciente fuente de preocupación, lo que implica que la protección de la API va más allá de su resguardo en sí; también es fundamental asegurar las claves, credenciales y otros datos sensibles que no deben ser divulgados públicamente. Estas medidas de seguridad son, en ocasiones, pasadas por alto por los desarrolladores.

3.2.1 Análisis de vulnerabilidades

- **Exposición de claves y credenciales:** La falta de gestión adecuada de claves puede llevar a su exposición, permitiendo a actores malintencionados acceder a sistemas críticos. Es crucial implementar prácticas de seguridad como la rotación de claves y el uso de herramientas de gestión de secretos.
- **Casos documentados:**
    - **IRS 2015:** El incidente en el que el IRS (Internal Revenue Service) expuso más de 300,000 registros expuestos, subraya la necesidad de evaluaciones de seguridad exhaustivas en las APIs. La falta de controles de acceso robustos puede permitir que datos sensibles sean accedidos sin autorización. [15]
    - **Sistema Aadhar:** La brecha en el sistema de identificación Aadhar, destaca la importancia de proteger las APIs que manejan información personal y biométrica. La implementación de autenticación multifactor y el cifrado de datos en tránsito y en reposo son esenciales para salvaguardar esta información. [16]
- **Caso Facebook:** La vulnerabilidad que permitió a atacantes obtener el token de acceso, ilustra la necesidad de asegurar el código fuente de las aplicaciones. Los controles de seguridad en el desarrollo, como la revisión de código y pruebas de penetración, son vitales para identificar y remediar vulnerabilidades antes de que puedan ser explotadas. [17]

Estrategias de mitigación

- Se debe educar, concientizar y capacitar a los desarrolladores sobre las mejores prácticas de seguridad y la importancia de proteger las credenciales y datos sensibles.

- Cada cierto tiempo se debería realizar auditorías de seguridad para identificar y remediar vulnerabilidades.
- Aplicar cifrado tanto durante la transmisión como en almacenamiento para garantizar la protección de los datos sensibles.
- Establecer políticas de acceso basadas en roles (RBAC) y asegurar que solo personal autorizado tenga acceso a la información crítica.

Este análisis demuestra que la seguridad en la gestión de APIs no solo se trata de proteger la interfaz, sino de implementar un enfoque integral que contemple la protección de todas las claves y datos confidenciales asociados. La combinación de tecnologías adecuadas, prácticas de desarrollo seguras y un enfoque proactivo puede mitigar significativamente los riesgos asociados a las vulnerabilidades de seguridad.

3.3 Impacto Comercial

La mayoría de las empresas se esfuerza por brindar y garantizar que la seguridad se encuentre bien integrada en sus productos o servicios, es fundamental que los consumidores que utilicen los servicios, entiendan las implicaciones de seguridad para su uso. Si existiese un conjunto débil en la interfaz, al consumir una Api, provocaría que la empresa se exponga a varios problemas de seguridad relacionada a la confidencialidad, integridad, disponibilidad y responsabilidad.

Por todo esto es fundamental que las aplicaciones, sistemas de amenazas, flujos de datos y arquitectura o diseño, se conviertan en parte fundamental de ciclo de vida del desarrollo. Por tal motivo se debe realizar una revisión de código específico de seguridad, las pruebas de penetración rigurosas se deberían considerar como requisito fundamental.

3.4 Tendencias y desafíos

Las nuevas tecnologías están transformando la seguridad informática y el control de acceso. La inteligencia artificial (IA) y el aprendizaje automático (ML) están cambiando la forma en que detectamos amenazas y automatizamos respuestas. Las técnicas de autenticación biométrica están avanzando, y la tecnología blockchain proporciona métodos innovadores para la gestión segura de

identidades. La arquitectura Zero Trust está cambiando el enfoque de seguridad al reducir la confianza implícita en los usuarios. Sin embargo, no dejamos de enfrentar retos como la evolución de las amenazas, la protección de datos en la nube y la adaptación a nuevas regulaciones. Estos desarrollos y desafíos están dando forma al futuro de la seguridad y el control de acceso, lo que requiere una adaptación constante.

4. Conclusiones

En el contexto actual de la tecnología y el desarrollo de software, la seguridad de los servicios, tanto internos como externos, es una prioridad crucial. La integración de mecanismos de seguridad robustos no solo protege los datos y la infraestructura, sino que también fortalece la confianza del usuario y la integridad del sistema. Esta tesis ha explorado y analizado diversos enfoques para garantizar esta seguridad, centrándose en la autenticación mediante OAuth 2.0, el uso de un API Gateway y la integración de las recomendaciones del OWASP Top Ten.

La autenticación mediante OAuth 2.0 se ha destacado como un estándar fundamental para la autorización segura en servicios modernos. Su capacidad para permitir un control granular de acceso y su compatibilidad con múltiples tipos de aplicaciones la convierten en una herramienta esencial para proteger la integridad de las sesiones de usuario y los recursos accesibles. OAuth 2.0 no solo proporciona un marco para la autenticación segura, sino que también ofrece mecanismos de tokenización que limitan la exposición de credenciales y mitigan riesgos asociados con la gestión de usuarios.

El uso de un API Gateway complementa esta estrategia al actuar como una capa de intermediación que facilita la gestión y seguridad de las interfaces de programación de aplicaciones (APIs). Un API Gateway centraliza la autenticación, el enrutamiento de solicitudes, y la implementación de políticas de seguridad, lo que reduce la superficie de ataque y simplifica el manejo de las conexiones entre servicios internos y externos. Su capacidad para aplicar reglas de seguridad y controlar el acceso a nivel de API fortalece la protección contra amenazas y vulnerabilidades.

Finalmente, la consideración de las recomendaciones del OWASP Top Ten

proporciona una perspectiva valiosa sobre las amenazas más comunes y críticas en el ámbito de la seguridad de aplicaciones web. Al abordar estas vulnerabilidades con medidas proactivas, es posible mitigar riesgos y mejorar la resiliencia del sistema frente a ataques. Integrar prácticas recomendadas por OWASP en el desarrollo y la operación de servicios ayuda a mantener un estándar elevado de seguridad y a reducir la probabilidad de brechas y compromisos.

En conclusión, la combinación de OAuth 2.0 para autenticación, un API Gateway para gestión y seguridad, y la implementación de medidas basadas en el OWASP Top Ten constituye un enfoque integral para la protección de servicios tanto internos como externos. Este enfoque holístico no solo mejora la seguridad de los sistemas, sino que también optimiza la gestión de riesgos y contribuye a una arquitectura más robusta y confiable. La continua evolución de las amenazas y las tecnologías requiere una vigilancia constante y una adaptación de las estrategias de seguridad para garantizar una protección efectiva en el dinámico entorno digital actual.

5. Bibliografía

[1] «oauth,» [En línea]. Available: https://oauth.net/. [Último acceso: 16 03 2024].

[2] «digitalocean,» [En línea]. Available: https://www.digitalocean.com/community/tutorials/una-introduccion-a-oauth-2-es. [Último acceso: 20 03 2024].

[3] «auth0,» [En línea]. Available: https://auth0.com/docs/authenticate/protocols/oauth. [Último acceso: 22 03 2024].

[4] «oauth,» [En línea]. Available: https://oauth.net/2/. [Último acceso: 28 03 2024].

[5] «datatracker,» [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc6749#section-1.1. [Último acceso: 10 04 2024].

[6] «datatracker,» [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc6749#section-1.3.1. [Último acceso: 20 04 2024].

[7] «salesforce,» [En línea]. Available: https://help.salesforce.com/s/articleView?id=sf.remoteaccess_oauth_flow_errors.htm&type=5. [Último acceso: 23 04 2024].

[8] auth0. [En línea]. Available: https://auth0.com/es/intro-to-iam/what-is-openid-connect-oidc. [Último acceso: 27 04 2024].

[9] «identityserver4,» [En línea]. Available: https://identityserver4.readthedocs.io/en/latest/endpoints/userinfo.html. [Último acceso: 05 05 2024].

[10] «okta,» [En línea]. Available: https://www.okta.com/identity-101/digital-certificate/. [Último acceso: 10 05 2024].

[11] «hubspot,» [En línea]. Available: https://blog.hubspot.es/website/que-es-api-gateway. [Último acceso: 12 05 2024].

[12] «NORMA ISO 27001,» 20 07 2024. [En línea]. Available: https://www.normaiso27001.es/#h1.

[13] «owasp,» [En línea]. Available: https://owasp.org/Top10/A00-about-owasp/. [Último acceso: 17 07 2024].

[14] «owasp,» [En línea]. Available: https://owasp.org/Top10/es/. [Último acceso: 20 07 2024].

[15] «irs,» [En línea]. Available: https://www.irs.gov/newsroom/news-releases-for-march-2015. [Último acceso: 27 07 2024].

[16] «elcopernico,» [En línea]. Available: https://elcopernico.com/2018/09/28/privacidad-y-seguridad-el-caso-de-aadhaar-en-india/. [Último acceso: 05 08 2024].

[17] «cnet,» [En línea]. Available: https://www.cnet.com/news/privacy/facebook-says-data-leak-is-from-old-vulnerability-that-no-longer-exists/. [Último acceso: 07 08 2024].