UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
INGENIERO DE SISTEMAS

TEMA:
ESTUDIO DE MEJORA EN LA SEGURIDAD DE APLICACIONES WEB Y DE APLICACIONES MÓVILES MEDIANTE EL PROTOCOLO OPENID CONNECT OPENSOURCE UTILIZADO EN IDENTITY SERVER 4

AUTORES:
RICARDO ANTONIO PONCE BEDOYA
OSWALDO RAMIRO ROJAS CEVALLOS

DIRECTOR:
JOSÉ LUIS AGUAYO MORALES

Quito, febrero de 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Ricardo Antonio Ponce Bedoya con documento de identificación N° 1711933034 y Oswaldo Ramiro Rojas Cevallos con documento de identificación N° 1500583339, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad con el tema: "ESTUDIO DE MEJORA EN LA SEGURIDAD DE APLICACIONES WEB Y DE APLICACIONES MÓVILES MEDIANTE EL PROTOCOLO OPENID CONNECT OPENSOURCE UTILIZADO EN IDENTITY SERVER 4", mismo que ha sido desarrollado para optar por el título de INGENIERO DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado por la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

RICARDO ANTONIO
PONCE BEDOYA
CI: 1711933034

OSWALDO RAMIRO
ROJAS CEVALLOS
CI: 1500583339

Quito, febrero 2020

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, ESTUDIO DE MEJORA EN LA SEGURIDAD DE APLICACIONES WEB Y DE LAS APLICACIONES MÓVILES MEDIANTE EL PROTOCOLO OPENID CONNECT OPENSOURCE UTILIZADO EN IDENTITY SERVER 4, realizado por Ricardo Antonio Ponce Bedoya y Oswaldo Ramiro Rojas Cevallos, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, febrero 2020

.....

JOSÉ LUIS AGUAYO MORALES

CI: 1709562597

ESTUDIO DE MEJORA EN LA SEGURIDAD DE APLICACIONES WEB Y DE APLICACIONES MÓVILES MEDIANTE EL PROTOCOLO OPENID CONNECT OPENSOURCE UTILIZADO EN IDENTITY SERVER 4

STUDY TO IMPROVE THE SECURITY OF WEB APPLICATIONS AND MOBILE APPLICATIONS USING THE OPENID CONNECT OPENSOURCE PROTOCOL USED IN IDENTITY SERVER 4

José L. Aguayo[1], Ricardo A. Ponce[2], Oswaldo R. Rojas[3]

Resumen

La creación de cuentas de acceso por plataforma y el acceso a ellas utilizando las tecnologías actuales son susceptibles a múltiples ataques con el fin de obtener información sensible y útil para el atacante. Según gobierno de Estados Unidos, se invierte varios billones de USD para solucionar los incidentes de ciber defensa, los cuales cerca del 40% son de acceso no autorizado. Según ***Kaspersky Lab***, hay más de 110 mil ataques diarios de malware que corresponde a Ecuador. Este problema es común en aplicaciones móviles y web.

La unificación del proceso de autenticación soluciona en parte el impacto de riesgos que presentan los diferentes tipos de desarrollos en aplicaciones web y móviles, lo que facilita únicamente la autenticación con encriptación JWT hacia los usuarios registrados en el Servidor de Identidad. Las características que se utilizarán en este artículo se basan en el Servidor de Identidad con el protocolo OpenID Connect, lo cual puede ser usado por desarrolladores y futuras investigaciones de universidades.

La metodología a utilizar para análisis de la documentación es el mapeo sistemático con el objetivo de identificar y evidenciar cada una de las diferentes etapas de solución más posibles fallos.

La implementación realizada de la solución del Servidor de Identidad otorgó un nivel elevado de seguridad frente al acceso no autorizado en aplicaciones web y móviles, lo cual representa una mejora en la seguridad. Después de analizada la implementación de este artículo existen posibles variables que no han sido validadas y las cuales seguramente van a alterar el resultado de esta investigación.

Palabras claves: Servidor de Identidad, OpenID Connect, aplicaciones móviles y web, Aplicación informática, codificación, software de código abierto.

Abstract

The creation of access account by platform and access to them using current technologies are susceptible to multiple attacks in order to obtain sensitive and useful information for the attacker. According to the United States government, we invite several billon USD to resolve cyber defense incidents, which are about 40% unauthorized access. According to Kaspersky Lab, there are more than 110 thousand daily malware attacks correspond to Ecuador. This problem is common in mobile and web applications.

The unification of the authentication process partially solves the impact of risks presented by

[1] Docente de la Carrera de Ingeniería de Sistemas – Universidad Politécnica Salesiana, Sede Quito – Campus Sur.
Autor para correspondencia: jaguayo@ups.edu.ec
[2] Estudiante de Ingeniería de Sistemas – Universidad Politécnica Salesiana, Sede Quito – Campus Sur.
Autor para correspondencia: rponce@est.ups.edu.ec
[3] Estudiante de Ingeniería de Sistemas – Universidad Politécnica Salesiana, Sede Quito – Campus Sur.
Autor para correspondencia: orojas@est.ups.edu.ec

different types of developments in web and mobile applications which only facilitates authentication with JWT Encryption to users registered in Identity Server.
The features that will be used in this article are based on Identity Server with the OpenID Connect protocol, which can be used by developers and future university research.
The methodology used for the analysis of the documentation is the systematic mapping with the objective of identifying and evidencing each of the different stages of solution plus possible failures. The implementation of the Identity Server solution provided a high level of security against unauthorized access in web and mobile applications, which represents an improvement in security. After analyzing the implementation of this article, there are possible variables that have not been validated and which will surely alter the result of this research.

Keywords: Identity Server, OpenID Connect, mobile and web applications, computer application, coding, open source software.

1. Introducción

El gobierno de Estados Unidos destina $5.7b USD[1] para solucionar los incidentes de ciber defensa, los cuales el 38%[1] son de acceso no autorizado. Actualmente los desarrolladores al momento de aplicar sus conocimientos en aplicaciones públicas se ven afectados por ataques de acceso no autorizado y es importante que estén al tanto de cómo se realiza este tipo de irrupciones para poder combatirlos.
En América Latina, según ***Kaspersky Lab***, hay más de 746 mil ataques de malware diarios y el 15%[2] está asociando con usuarios en Ecuador. Este problema es común en aplicaciones móviles y web.

Según Hacking Ético conocer el ataque es la mejor defensa[3], el acceso no autorizado puede darse por diferentes tipos de irrupciones como: ataques de Fuerza Bruta, Caballos de Troya y similares. Por lo general este tipo de inconveniente se puede dar por sitios desarrollados en el cual la autenticación se realiza en un solo paso[3], es decir ingresando un usuario y contraseña en el mismo evento.

Adicionalmente puede existir algunos casos de Ingeniería Social que pueden caer en este tipo de vulnerabilidades[4].

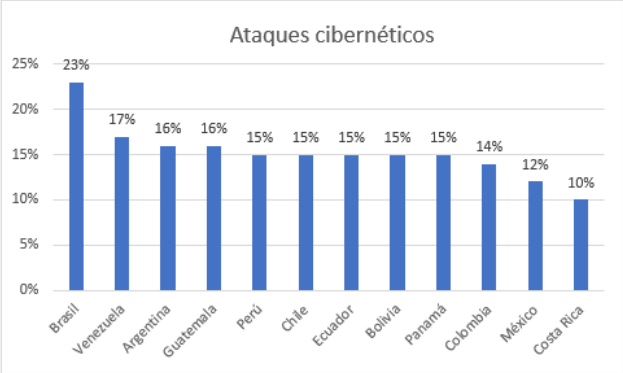

Figura 1. Kaspersky Lab 60 % ataques cibernéticos [2]

Este artículo pretende dar a conocer e impulsar la seguridad en aplicaciones actuales mejorando el acceso a los sitios web y aplicaciones móviles mediante el uso de un Marco de Desarrollo llamado Servidor de Identidad basado en el protocolo OpenID Connect y que es un software OpenSource que brinda una solución viable a nivel de seguridad y en tiempos desarrollo, además de cero coste en la adquisición y entendimiento de este Marco de Desarrollo dando la posibilidad para que, en futuros desarrollos, la herramienta antes mencionada pueda ser implementada en aplicaciones Web y Móviles.

Según WSO2, el Servidor de Identidad es una solución que ofrece una garantía a futuro en las empresas, dado que es una plataforma abierta la cual los desarrolladores pueden trabajar de manera sencilla y adaptarla a las necesidades que tengan[5].

2. Estado del arte

La Universidad de London en el primer estudio de campo en la implementación de seguridad de Google OpenID[6] examino las vulnerabilidades de seguridad en accesos no autorizados en el proceso de autenticación de 103 sitios web RP de los principales 1000 sitios web[6], direccionando el abandono del protocolo anterior.

El estudio realizado por en la investigación de Mainka, Vladislav y Schwenk en la Universidad de Bochmun[7] presento un 75% de vulnerabilidades en ataques monofásicos en una evaluación de bibliotecas con OpenID Connect

abusando del defecto lógico y no de un error de implementación [7] en el proceso de verificación de credenciales.

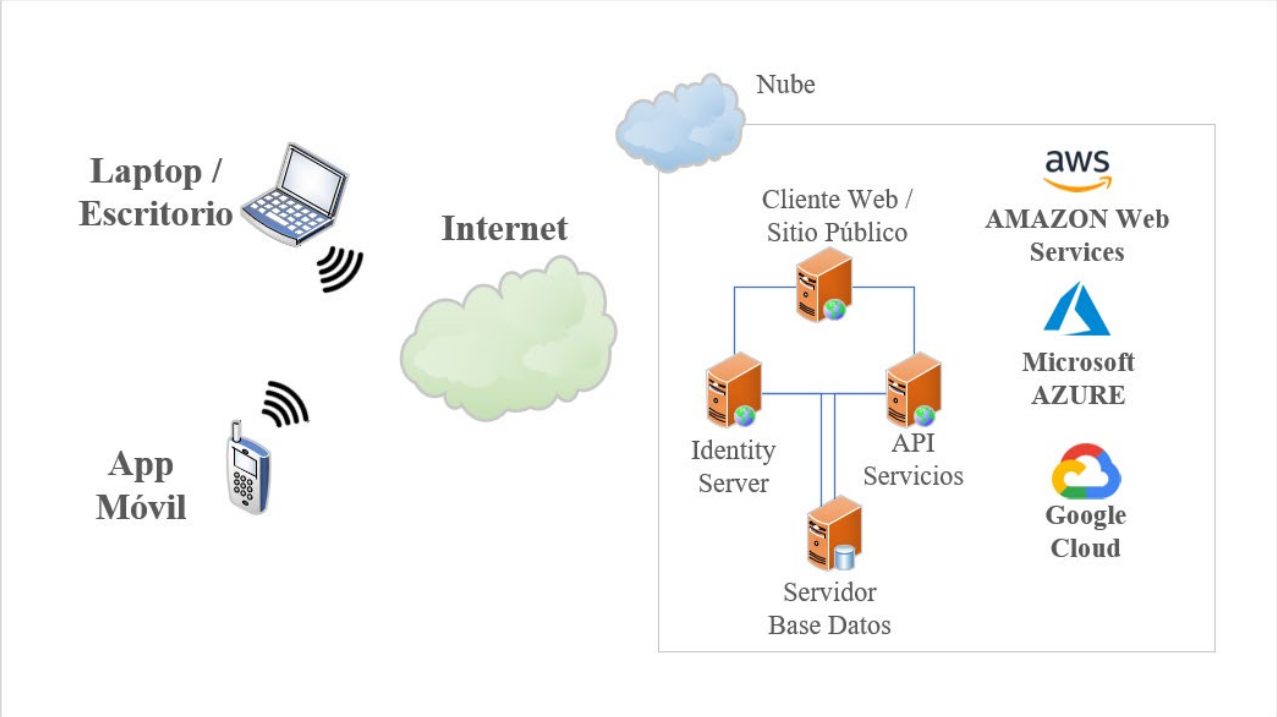

Figura 2. Arquitectura de Implementación para Pruebas

3. Levantamiento de información del análisis de seguridad

Acorde a PANDA MEDIACENTER[8] Los desarrolladores deberían implementar en sus apps medidas más sofisticadas, precisamente lo que recomienda Facebook y Amazon y los demás proveedores de Cloud.

Los investigadores de Darmstadt Fraunhofer recomiendan emplear "un esquema de control de accesos" el cual, sin embargo; brilla por su ausencia en la mayoría de las 750 mil aplicaciones que se han analizado[8].

4. Levantamiento de información protocolo OpenID Connect

De acuerdo con Welivessecurity actualmente, el 50% de vulnerabilidades en sistemas son de origen de diseño[9].

Esto se diferencia de fallas en implementación definidas como bugs. OpenID Connect es la capa de identidad simple por encima del protocolo OAUTH 2.0.

Permite a los clientes verificar la identidad del usuario final en función de la autentificación realizada por un servidor de autorización, así como para obtener información de perfil básica sobre el usuario final.

OpenID Connect permite a clientes de todo tipo, incluidos los basados en web y móviles generar Tokens de identidad basados en JSON (JWT) mediante flujos OAuth 2.0[10].

Los Tokens de ID son semejantes al concepto de tarjeta de identidad con formato JWT, suscrito por Open ID (OP) y contiene las siguientes características:

- Confirma la identidad del usuario.
- Define la autoridad emisora.
- Genera un público particular.
- Es firmado de forma digital.
- Puede cifrarse.

El pseudo código de Token ID se encapsula en un objeto JSON:

```
{
  "nbf": 1579380156,
  "exp": 1579383756,
  "iss": "https://ns.bayteq.com:60210",
  "aud": [],
  "client_id": " client",
  "sub": "d775-4b8d-b571-a70ebbce10b6",
  "auth_time": 1579380155,
  "idp": "local",
  "given_name": "usuario
  "scope": ["openid","profile", "email"],
  …
}
```


Figura 3. Desencriptado JWT [1]

En el encabezado del Token ID la notificación JSON se codifica en una cadena de base 64 URL-safe.


```
eyJhbGciOiJSUzI1NiIsImtpZCI6IjYxN
0NBMTUwMkVFMUFDOEEwMzA4
MDUyQTgzRTNERDEzQjCRDg0Njk
iLCJ0eXAiOiJKV1QiLCJ4NXQiOiJZ
WHloVUM3aHJJb0RDQVVxZy1QZE
U3SzloR2sifQ.eyJuYmYiOjE1Nzg4ND
k0OTEsImV4cCI6MTU3ODg1MzA5M
SwiaXNzIjoiaHR0cHM6Ly9ucy5iYXl0
ZXEuY29tOjYwMjEzIiwiYXVkIjpbIm
h0dHBzOi8vbnMuYmF5dGVxLmNvb
To2MDIxMy9yZXNvdXJjZXMiLCJhc
GkxIl0sImNsaWVudF9pZCI6InJvY2xp
ZW50X2FwaV9jb3J1Iiwic3ViIjoiODI0
YTQ3ZjktZTllNi00NGQ2LTg2M2EtN
zI5YmNlOTBlZjdjIiwiYXV0aF90aW11
IjoxNTc4ODQ5NDkxLCJpZHAiOiJsb
2NhbCIsImNvb3BDb2RlIjoiQ0FDUEV
DMDIwMTgiLCJmdW5jdGlvbmFsaX
RpZXMiOiJbXSIsInNjb3BlIjpbImFwa
TEiXSwiYW1yIjpbInB3ZCJdfQ.PboA
p6iaH9y7C3c20ki5Sw0VVmScjsSP1u
RusmRAYZ2gPQNXY9o7JYGwOV4-
OoL8QK226HLRMXeUiEbofMImgiqp
pmA2f9RUfvwT_wyowHu6yb4wjug4v
Sorv8Ud9nOcyVjIYFgcK9OH7WpWv
TpFelEC8gYXeY04YepFh3x6AE6g_d
Nv1fZF5GSSkPUs2zgqOLEzAupvf5g8
2C8rhhd2cmQhUAuFEMyqDBhOFtNg
gvjAIJw6Y7od1s5BkM53DSC8bm_hl
BUuf8eNtI9Aq6Pm9WsLr9o1AWA28q
Dt12hQutGb_yEQHlBgQ4Dwv6_1Jfo-
tw6ub3_vzUukKVZqEg
```


Figura 4. JWT de Sesión [1]

La autentificación se da mediante un proveedor de identidad el cual realiza la validación la sesión o la credencial de usuario.

La estructura y codificación se basan en Json Web Token (JWT) RFC 7519[11].

La solicitud de un Token de identificación tiene lugar con el proveedor de identidad comprobando la sesión o credencial. El Token ID es solicitado mediante el protocolo OAuth 2.0 siendo este el mecanismo de autorización para las aplicaciones que obtienen Tokens de acceso en plataformas web y aplicaciones móviles.

Las tres opciones de flujo que contiene esta investigación:

- Flujo de código de autorización utilizada para aplicaciones web tradicionales y aplicaciones móviles nativas.
- Flujo Implícito (***Implicit Flow***) basado en JavaScript para aplicaciones que no disponen de Backend.
- Flujo híbrido utilizado de forma poco frecuente permitiendo que el Frontend y Backend reciban de forma separada los Tokens. Se aplica para aplicaciones móviles híbridas.

4.1 Tabla de comparación de Flujos

Propiedad de flujo	Código	Implícita	Híbrida
Redirecciona miento de navegador	x	x	x
Solicitud de Backend		?	x
Tokens desplegados al navegador	?	x	x
Cliente autenticado	x	?	x

Tabla 1. Flujo de autenticación [2]

El Token ID puede ser utilizado para aplicarlo en diferentes formas de inicio de sesión.

- **Sesión sin estado:** El Token ID puede desarrollar una sesión denominada sin estado de forma ligera utilizando una cookie en el explorador, con lo cual se suprime el requerimiento de

almacenamiento de sesiones por parte del servidor.

- **Paso de identidad a terceras partes:** El Token de identificación puede ser utilizado pasando a otros componentes de aplicación o ciertos servicios de Backend cuando se desea identificar al cliente.
- **Intercambiar tokens:** Intercambiar el Token de ID se lo puede realizar mediante un Token de acceso en el punto de conexión del Token de la autorización de un servidor OAuth 2.0.
- **Flujo de código:** Desplegado en dos etapas.

	Etapa 1	Etapa 2
Objetivo	1 autenticación de usuario 2 percepción de la validación de usuario	1 autenticación del cliente (opcional) 2 intercambio de códigos para Tokens
Ruta	Solicitud inicial de canal	Solicitud final de canal
Par	Finalización de autorización	Token de conexión
Resultado satisfactorio	Código autorizado (input del paso 2)	Tokens de identificación más el de Acceso OAuth 2.0

Tabla 2. Flujo de autenticación en dos etapas [3]

4.1.1 Flujo etapa 1

La autenticación Open ID es una solicitud de acceso de OAuth 2.0 para ingresar a la identidad de usuario dada por el valor Open ID dentro del parámetro SCOPE[10].

```
HTTP/1.1 301 Found
Location:
https://openid.temp.com/login?
response_type=code
&scope=openid
&client_id=o6DefSjlp5
&state=bg1hgktmeik
&redirect_uri=https%3A%2F%2Fclient
.test.com%2Fcb
```

Figura 5. Petición de acceso a Login [2]

OP (Proveedor OpenID) invoca al cliente redirect_url con el código de acceso autorizado o con un código de error si la solicitud es invalido.

```
HTTP/1.1 301 Found
Location: https://client.temp.com/cb?
Code=Pq2x2PCfYPPZP7XyPcJV
&state=bf1jgkp2ehj
```

Figura 6. Solicitud inválida [2]

4.1.2 Flujo etapa 2

Código de acceso, es la credencial intermedia que se encargará de codificar la autorización obtenida en la primera etapa. La recuperación del Token ID procede en enviar el código desde el RP (Relying Party) al OP (Proveedor OpenID Connect).

En esta solicitud se la debe realizar de forma directa al canal inferior debido a estos dos motivos encontrados[10].

- Autenticar validando los clientes confidenciales al OP (Proveedor OpenID) antes de develar los Tokens.
- Entregar tokens al RP (Relying Party) evitando exponerlos con el navegador.

En el punto de conexión de Token del OP (Proveedor OpenID) se produce el intercambio de código.

```
POST /token http / 1.1
HOST: open ID.temp.com
Content-type:application/-www-fond-
urlencode
autorization:basic
dyYDbHPTb4G1NymZEGnpnG1N3K
wGrand_type=authorization_code
&code=Pr2x2PCfYPPZcZP7Yz6cJB
&redirect_url=https%3a%ef%2fclient.te
mp.com%2fcb
```

Figura 7. Petición de acceso a Login con nivel de Autorización [2]

Una vez realizado la validación correspondiente el OP (Proveedor OpenID) devuelve un JSON con el Token de acceso, el Token ID y Token de actualización.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6IjYx
N0NBMTUwMkVFMUFDOEEwMzA
4MDUyQTgzRTNERDEzQjJCRDg0Nj
kiLCJ0eXAiOiJKV1QiLCJ4NXQiOiJZ
WHloVUM3aHJJb0RDQVVxZy1QZE
U3SzloR2sifQ.eyJuYmYiOjE1Nzg4ND
k0OTEsImV4cCI6MTU3ODg1MzA5M
SwiaXNzIjoiaHR0cHM6Ly9ucy5iYXI0
ZXEuY29tOjYwMjEzIiwiYXVkIjpbIm
h0dHBzOi8vbnMuYmF5dGVxLmNvb
To2MDIxMy9yZXNvdXJjZXMiLCJhc
GkxIl0sImNsaWVudF9pZCI6InJvY2xp
ZW50X2FwaV9jb3JlIiwic3ViIjoiODI0
YTQ3ZjktZTllNi00NGQ2LTg2M2EtN
zI5YmNlOTBlZjdjIiwiYXV0aF90aW1l
IjoxNTc4ODQ5NDkxLCJpZHAiOiJsb
2NhbCIsImNvb3BDb2RlIjoiQ0FDUEV
DMDIwMTgiLCJmdW5jdGlvbmFsaX
RpZXMiOiJbXSIsInNjb3BlIjpbImFwa
TEiXSwiYW1yIjpbInB3ZCJdfQ.PboA
p6iaH9y7C3c20ki5Sw0VVmScjsSP1u
RusmRAYZ2gPQNXY9o7JYGwOV4-
OoL8QK226HLRMXeUiEbofMImgiqp
pmA2f9RUfvw-
T_wyowHu6yb4wjug4vSorv8Ud9nOcy
VjI-
YFgcK9OH7WpWvTpFelEC8gYXeY0
4YepFh3x6AE6g_dNv1fZF5GSSkPUs
2zgqOLEzAupvf5g82C8rhhd2cmQhU
AuFEMyqD-B-
hOFtNggvjAIJw6Y7od1s5BkM53DSC
8bm_hlBUuf8eNtl9Aq6Pm9WsLr9o1A
WA28qDt12hQutGb_yEQHlBgQ4Dwv
6_1Jfo-tw6ub3_vzUukKVZqEg",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

Figura 8. Token de acceso autorizado [2]

El Token ID será validado por el RP (Relying Party) antes de ser aprobado tiene estructura JWT.

5. Materiales y Métodos

Se definieron como materiales utilizados para validar esta implementación la plataforma del Servidor de Identidad con OpenID Connect como herramienta de solución para unificación de la validación y autenticación de los registros en plataformas web y móviles.

Para los diferentes escenarios de prueba se utilizó la validación de la herramienta OWASP la cual provee las diferentes alarmas, alertas, definiciones, métodos, parámetros, evidencia y

posibles soluciones a implementar dentro de los parámetros alto, medio y bajo.

5.1 Metodología

Para el desarrollo de los diferentes escenarios se utilizará la metodología del proyecto piloto, por lo cual ejecuta normalmente los mismos pasos pensados para el estudio real, pero en una escala más pequeña con el objetivo de detectar los posibles fallos o problemas del estudio en el desarrollo de los diferentes escenarios[12]

5.2 Captación de información de usuario

Open ID Connect: Define varios tipos de notificaciones o atributos destinados a proporcionar datos del cliente.

Valor	Variable asociada
Email	email, email_verified
Phone	phone_number, phone_number_verified
Profile	name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, updated_at
address	Address

Tabla 3. Información adicional encapsulada en el Token de Autorización [4]

El formato de notificación JSON y en el cual viene encapsulado en formato JWT es:

```
{
"sub": "Melisa",
"email": "melisa@mundo.com.ec",
"email_verified" true,
"name": "Melisa Andrade",
"given_name": "Melisa",
"family_name": "Andrade",
"phone_number": "+593 (2) 2418452",
"profile":
"https://temp.com/users/melisa",
"https://temp.com/groups": [ "audit",
"admin"]
}
```

Figura 9. Información adicional a encapsular [4]

5.1.1 Autorización endpoint

El servidor OP (Proveedor OpenID) es un punto de conexión donde se solicita al usuario su autenticación y se concede al mismo un acceso y una identidad mediante un Token ID[10].

5.1.2 Token endpoint

Se permite a la aplicación cliente mediante el punto de conexión, intercambiar la información recibida desde la conexión autorizada por el identificador de Token y otro de acceso, se puede implementar varios formatos de concesión OAuth 2.0 en la entrega de Tokens, como JWTISAML2.0[10].

5.1.3 UserInfo Endpoint

Despliega la información del usuario al cliente, para lo cual requiere un Token de acceso aprobado[10].

```
GET /userinfo HTTP /1.1
Host: openid.temp.com
AUTORIZATION:MELISA
P2BY43ijJT
```

Figura 10. Encabezado de información de Autorización [4]

Para la aplicación de escenarios se podrá implementar endpoints opcionales como: WebFinger, Metadatos provistos por el proveedor, JWK, registro del cliente, administrador de sesión.

5.2 Identity Server 4

El Servidor de Identidad se enmarca dentro del OpenSource, OpenID Connect y OAuth 2.0 implementado para .Net Core e incorpora los protocolos y puntos de extensión requeridos en la autenticación mediante Token[5], con una sesión de inicio y acceso vía API en aplicaciones.

El Servidor de Identidad combina middlewares y servicios, los cuales se pueden configurar en la clase de inicio.

5.2.1 Configuración de servicios

En el sistema de Interfaz se puede agregar los servicios del Servidor de Identidad invocando los certificados X.509, claves RSA y EC a utilizar en firmas con Token y validación, su configuración puede utilizar el algoritmo de firma RS256[13] el cual viene implementado en la configuración por defecto en el Servidor de Identidad.

```
public void ConfigureServices(IServiceCollection services)   {
var migrationsAssembly =
typeof(Startup).GetTypeInfo().Assembly.GetName().Name;
var connectionString =
Configuration.GetConnectionString("DefaultConnection");
var schemaName =
Configuration.GetConnectionString("SchemaIdentityServer4");
services
    .AddMvc()
    .SetCompatibilityVersion(
        CompatibilityVersion.Version_2_2
    );
services
    .AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            connectionString, dbOpts =>
            dbOpts.MigrationsAssembly(
                typeof(Startup).Assembly.FullName)
            )
        );
services
    .AddIdentity<ApplicationUser, ApplicationRole>
        (options =>options.Stores.MaxLengthForKeys=128)
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();
services
    .AddIdentityServer(options => {
    options.Events.RaiseSuccessEvents =
    true;
    options.Events.RaiseFailureEvents =
    true;
    options.Events.RaiseErrorEvents =
    true;
    options.UserInteraction.LoginUrl =
    "/iniciar-sesion";
    options.UserInteraction.
    LoginReturnUrlParameter
    ="cliente";
    })
    .AddSigningCredential(new X509Certificate2(
    Path.Combine(
        Environment.ContentRootPath,
        "IdentityServerDmlAuth.pfx"
    ), "Clave123"))
    .AddConfigurationStore(configDb =>
    {
    configDb.DefaultSchema = schemaName;
    configDb.ConfigureDbContext = db =>
        db.UseSqlServer(connectionString, sql =>
            sql.MigrationsAssembly(migrationsAssembly));
    })
    .AddOperationalStore(operationalDb =>
    {
        operationalDb.DefaultSchema = schemaName;
        operationalDb.ConfigureDbContext = db =>
            db.UseSqlServer(connectionString, sql =>
                sql.MigrationsAssembly(migrationsAssembly));
    })
    .AddAspNetIdentity<ApplicationUser>();
}
```

Figura 11. Configuración de servicios en el Servidor de Identidad [5]

En este artículo únicamente se enlista las configuraciones claves con sus métodos.

AddSigningCredential
AddConfigurationStore
AddOperationalStore

Figura 12. Configuraciones que va a contener el Servidor de Identidad [5]

Desde una lista de memoria de objetos se permite configurar el Servidor de Identidad con las API con lo cual se puede crear el prototipo ya que no se necesita realizar consultas dinámicas a la base de datos en el momento de su ejecución.

Este tipo de configuración podría ser utilizado en ambientes de producción y también en caso de que no sea un inconveniente el cambio de valor en la aplicación al ser ejecutada o reiniciada.

Lista de los mecanismos de almacenamiento (stores):

AddInMemoryClients
AddInMemoryIdentityResources
AddInMEmoryApiResources

Figura 13. Configuración en modo de memoria para el Servidor de Identidad [5]

5.2.2 Tiendas de prueba

La Clase TestUser es quien estructura un usuario, la estructura credenciales y notificaciones de un usuario en el Servidor de Identidad[13] es similar a utilizar almacenes y está alineado para desarrollo de prototipos y no para producción.

En este documento se enlista los servicios adicionales que contiene el Servidor de Identidad.

AddExtensionGrantValidator
AddSecretParser
AddSecretValidator
AddResourceOwnerValidator
AddProfileService
AddAuthorizeIntereactionResponseGen
erator
AddCustomAuthorizeRequestValidator
AddCustomTokenRequestValidator
AddRedirectUriValidator
AddAppAuthRedirectUriValidator
AddJwBearerClientAuthentication
AddMutualTIsSecretValidators

Figura 14. Servicios adicionales [5]

5.2.3 Cache

El Servidor de Identidad utiliza datos de configuración de usuario, estos pueden ser recibidos desde una BDD u otra procedencia[13], lo cual si es recurrente es costoso. Se enlista los diferentes tipos de uso de cache sin embargo en este documento no serán analizados.

AddInMemoryCaching
AddClientStoreCache
AddResourcesStoreCache
AddCorsPolicyCache

Figura 15. Servicios adicionales de trabajo con cache [5]

Configuración inicial de la aplicación, su ruta y accesos a los modos con los cuales trabaja:

```
public void Configure(IApplicationBuilder app,
IHostingEnvironment env) {
    if (env.IsDevelopment()) {
    app.UseDeveloperExceptionPage();
    app.UseDatabaseErrorPage();
    } else {
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
    }
    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();
    app.UseIdentityServer();
    app.UseMvc(routes =>
    {
    routes.MapRoute("MyRoute",
    "/iniciar-sesion",
    new { controller = "Account",
    action = "Login" });
    routes.MapRoute(
    name: "default",
    template:
    "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Figura 16. Configuración de la aplicación en el Servidor de Identidad [6]

5.2.4 Proceso de Evaluación

De acuerdo a los principios Ronald L Rivest del MIT[14] realiza un proceso de evaluación de 3*160 repeticiones ($0 \leq t \leq 480$) de SHA256 con hash de 256 bits en el diseño de algoritmos MD4 y MD5.

$$\begin{cases} T_1 = \sum_1^{\{256/512\}}(e) + Ch(e,f,g) + K_t^{\{256/512\}} + W_t \\ T_2 = \sum_0^{\{256/512\}}(a) + Maj(a,b,c) \\ h = g \\ g = f \\ f = e \\ e = d + T_1 \\ d = c \\ c = b \\ b = a \\ a = T_1 + T_2 \end{cases}$$

Figura 17. Fórmula Akashi Staoh

Actualizando el valor del hash intermedio llamado ***i-th*** de la siguiente manera donde + es la adición de mod 2^{32} para SHA256.

$$(H_0^{(i)}, H_1^{(i)}, H_2^{(i)}, H_3^{(i)}, H_4^{(i)}, H_5^{(i)}, H_6^{(i)}, H_7^{(i)}) = (a + H_0^{(i-1)}, b + H_1^{(i-1)}, c + H_2^{(i-1)}, d + H_3^{(i-1)}, e + H_4^{(i-1)}, f + H_5^{(i-1)}, g + H_6^{(i-1)}, h + H_7^{(i-1)})$$

Figura 18. Fórmula Akashi Staoh

Cuando estos pasos se repiten N veces y se procesan todos los N bloques de mensajes, el valor de hash final es obtenido.

$$H^{(N)} = (H_0^{(N)}, H_1^{(N)}, H_2^{(N)}, H_3^{(N)}, H_4^{(N)}, H_5^{(N)}, H_6^{(N)}, H_7^{(N)})$$

Figura 19. Fórmula Akashi Staoh

Los tamaños de los valores hash son 256 bits (32 bits x 8) para SHA256.

Escenario 1:

Wildcard Directive: Define las directivas de los diferentes orígenes que están o no establecidas[15]. El método utilizado es GET con los parámetros Content-Security-Policy, se identificó tres URL's que proveen esta alerta, evidenciando el siguiente resultado:

```
img-src 'self' data:; default-src 'self';
font-src 'self' https://fonts.gstatic.com/;
style-src 'self'
https://fonts.googleapis.com/ 'unsafe-
inline'; object-src 'none'; frame-
ancestors 'self'
https://dominio.azurewebsites.net:60210
; sandbox allow-forms allow-same-
origin allow-scripts; base-uri 'self';
script-src * data: https://www.mi-sitio-
web.com 'unsafe-inline' 'unsafe-eval'
```

Figura 20. Evidencia de directiva escenario 1[7]

Escenario 2:

Style-src unsafe-inline: Esta directiva es la encargada de especificar las fuentes válidas para los diferentes estilos de letras; fue analizada con el método GET con el parámetro Content-Security-Policy[16]. Dando como evidencia el siguiente resultado:

```
img-src 'self' data:; default-src 'self';
font-src 'self' https://fonts.gstatic.com/;
style-src 'self'
https://fonts.googleapis.com/ 'unsafe-
inline'; object-src 'none'; frame-
ancestors 'self'
https://dominio.azurewebsites.net:60210
; sandbox allow-forms allow-same-
origin allow-scripts; base-uri 'self';
script-src * data: https://www.mi-sitio-
web.com 'unsafe-inline' 'unsafe-eval'
```

Figura 21. Evidencia de directiva escenario 2[4]

Escenario 3:

Absence of Anti-CSRF Tokens: Esta directiva define una técnica de suplantación de petición en sitios web; este ataque va directo al explorador del afectado y pide que se valide ciertos servicios como: cuenta de correo, cuenta bancaria entre otros; y envíe una solicitud mediante el sitio web frágil o débil[17] antes mencionado. En base a esto la petición será realizada suplantando el nombre del usuario registrado y después explotando todas las posibilidades a las que pueda acceder con dicho usuario final. El método utilizado fue GET[18].

Escenario 4:

Cookie Without Secure Flag: Esta directiva está definida de acuerdo al indicador de seguridad en la cookie, siempre que se utilice conexión HTTP que no esté cifrada, evitando que esta sea interceptada por un atacante, ya que al no establecer un parámetro de seguridad la cookie se enviará en texto sin cifrar por lo cual en caso de ataque se puede inducir este evento mediante enlaces a un usuario, sea directa la petición o utilizando un sitio alterno, incluso el dominio que emita la cookie no alojará el contenido para que acceda. El método utilizado fue GET y el parámetro se evidencia en el siguiente código:

```
Set-Cookie: .AspNetCore.Antiforgery.8Erxr3TmmQQ
```

Figura 22. Evidencia de directiva escenario 4[2]

Escenario 5:
Incomplete or No Cache-control and Pragma HTTP Header Set: Esta directiva define que el control que se realiza en el encabezado HTTP del cache no se lo ha registrado o configurado correctamente lo que permite que los servidores proxy y el navegador almacenen contenido en el cache[19]. El método utilizado es GET, el parámetro cache-control lo cual da como evidencia:

```
public,max-age=86400, must-revalidate
```

Figura 23. Evidencia de directiva escenario 5[2]

5.2.5 Definición de recursos

La definición en el sistema va acorde a los recursos que se protegen acorde a los datos de perfil mediante el uso de IResourceStore[20].

```
public void Configure(IApplicationBuilder app,
IHostingEnvironment env) {
    if (env.IsDevelopment()) {
    app.UseDeveloperExceptionPage();
    app.UseDatabaseErrorPage();
    } else {
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
    }
    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();
    app.UseIdentityServer();
    app.UseMvc(routes =>
    {
    routes.MapRoute("MyRoute",
    "/iniciar-sesion",
    new { controller = "Account",
    action = "Login" });
    routes.MapRoute(
    name: "default",
    template:
    "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Figura 24. Especificación de los servicios que contiene el Servidor de Identidad [7]

5.2.6 Recursos de Identidad

Se define un nombre único, adicionalmente se asignan notificaciones arbitrarias que incluyen un Token de identidad[20]:

```
private static IEnumerable<IdentityResource>
IdentityResources
=> new[] {
    new IdentityResources.OpenId()
    , new IdentityResources.Profile()
    , new IdentityResources.Email()
    , new IdentityResources.Address()
};
```

Figura 25. Campos adicionales en el encapsulamiento [8]

5.2.7 Definición de recursos vía API

Los Tokens de acceso para API deben estar registrados en los recursos[20].

```
private static IEnumerable<ApiResource>
ApiResources
=> new[] {
    new ApiResource(
    Constant.API_RESOURCE_DML_API,
    "API Centralizado") {
    ApiSecrets = {
    new Secret(
    Constant.API_RESOURCE_SECRET.Sha256())
    }}
    , new ApiResource
    {
    Name = "api2"
    , ApiSecrets =
    {
        new Secret("secret".Sha256())
    }
    , UserClaims =
    {
        JwtClaimTypes.Name
        , JwtClaimTypes.Email
    }
    }
};
```

Figura 26. Configuración de los recursos del API de Servicios [9]

6. Resultados y discusión

6.1 Inicio de sesión

Al iniciar sesión para la emisión de Tokens mediante el Servidor de Identidad, se registran dos controladores de cookies, el primero para la autenticación y el segundo para claves externas, mediante la clase IdentityServerConstants.

6.1.1 Invalidación de configuración de administrador de cookies

Se invalida la administración de cookies para la sesión Implícita para aplicaciones de Frontend y para las demás sesiones se mantiene dicha configuración.

6.1.2 Flujo inicio de sesión

Para la solicitud de acceso, conexión y autorización se utiliza el Marco de Desarrollo del Servidor de Identidad mediante credenciales otorgadas a cada cliente registrado en el Servidor, mediante lo cual la clase UserInteraction otorga la directiva de redireccionamiento usada por el parámetro returnURL.

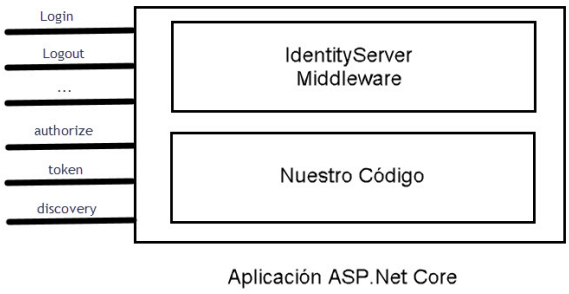

Figura 27. Flujo de autenticación y autorización[21] [10]

Definiendo la validación del parámetro que returnUrl con la ubicación destino se mitiga los ataques de redirección abierta.

El análisis de disponibilidad:

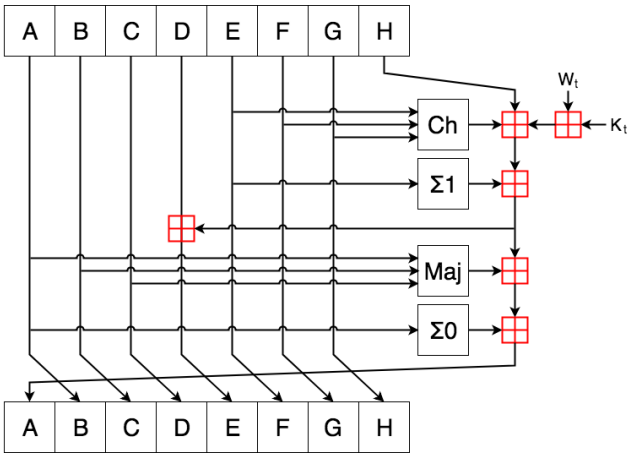

Figura 28. La iteración en la función de compresión de la familia SHA-2[22].

Utilizando las funciones Ch, Σ1, Maj y Σ0 con las variables asociadas:

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

Figura 29. Procedimiento.

La brecha en las infracciones de acceso inicial es del 0,00297. El resultado obtenido posterior a las 480 evaluaciones fue de 0,00167 en las infracciones con una mejora del 0.0013

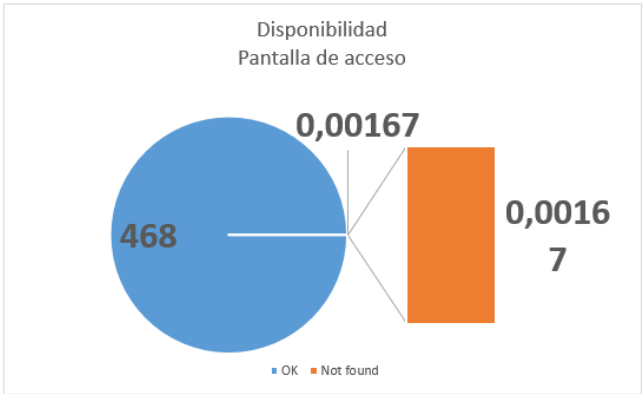

Figura 30. Cuadro estadístico acceso de Seguridad Login[21] [10]

7. Conclusiones

Dentro del análisis expuesto en la primera parte del documento sobre de los riegos actuales a sistemas web con vulnerabilidades del 75% en sus procesos de autenticación, es posible demostrar que existe actualmente herramientas para mitigar este tipo de problemática.

En este sentido, un Marco de Desarrollo como el Servidor de Identidad que se basa en el Protocolo OpenID Connect para dar seguridad a la autenticación de aplicaciones móviles y web actuales es provee las herramientas necesarias al implementar un entorno seguro se provee una mejorando la seguridad en más del 0,001 de las infracciones, la metodología desarrollada es similar a la utilizada por Wan, Chen y XiaoFeng Wan[23] y Li, Mitchell y Chen[24].

En concordancia ante lo expuesto el Servidor de Identidad es un Marco de Desarrollo para la implementación ágil de aplicaciones que busca facilitar el acceso seguro a los sitios futuros al implementarse una autenticación y autorización, identificando las vulnerabilidades que permiten al ataque tener éxito e inicie sesión en el RP tal cual lo expuesto por Li y Mitchell[6] de 103 RP analizados con una brecha del 1.9%.

Referencias

[1] «Cybersecurity-Risk-Determination-Report-FINAL_May-2018-Release.pdf». .

[2] «Kaspersky Lab registra un alza de 60% en ataques cibernéticos en América Latina | Blog oficial de Kaspersky». [En línea]. Disponible en: https://latam.kaspersky.com/blog/kaspersky-lab-registra-un-alza-de-60-en-ataques-

ciberneticos-en-america-latina/13266/. [Accedido: 14-ene-2020].
[3] A. d'information (France) conseil, installation et sécurisation des systèmes, *Seguridad informática - Hacking ético. Conocer el ataque para una mejor defensa (4a edición)*. Ediciones ENI, 2018.
[4] «Unauthorised access», *CERT NZ*. [En línea]. Disponible en: https://www.cert.govt.nz/individuals/explore/unauthorised-access/. [Accedido: 19-ene-2020].
[5] «Welcome to IdentityServer4 (latest) — IdentityServer4 1.0.0 documentation». [En línea]. Disponible en: http://docs.identityserver.io/en/latest/. [Accedido: 14-ene-2020].
[6] W. Li y C. J. Mitchell, «Analysing the Security of Google's implementation of OpenID Connect», *ArXiv150801707 Cs*, ago. 2015.
[7] C. Mainka, V. Mladenov, J. Schwenk, y T. Wich, «SoK: Single Sign-On Security — An Evaluation of OpenID Connect», en *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, Paris, 2017, pp. 251-266, doi: 10.1109/EuroSP.2017.32.
[8] «Los desarrolladores están descuidando la seguridad del "login" tanto en Android como en iOS - Panda Security Mediacenter». [En línea]. Disponible en: https://www.pandasecurity.com/spain/mediacenter/dispositivos-moviles/seguridad-login-android-ios/. [Accedido: 14-ene-2020].
[9] «10 consejos para el desarrollo seguro de aplicaciones | WeLiveSecurity». [En línea]. Disponible en: https://www.welivesecurity.com/la-es/2015/03/12/10-consejos-desarrollo-seguro-de-aplicaciones/. [Accedido: 14-ene-2020].
[10] «OpenID Connect explained | Connect2id». [En línea]. Disponible en: https://connect2id.com/learn/openid-connect. [Accedido: 14-ene-2020].
[11] J. Bradley, N. Sakimura, y M. Jones, «JSON Web Token (JWT)». [En línea]. Disponible en: https://tools.ietf.org/html/rfc7519.html. [Accedido: 14-ene-2020].
[12] «¿Qué es un estudio piloto? Definición y razones para hacerlo», *Clinic Cloud*, 09-jun-2017. [En línea]. Disponible en: https://clinic-cloud.com/blog/que-es-un-estudio-piloto-definicion/. [Accedido: 11-feb-2020].
[13] «Startup — IdentityServer4 1.0.0 documentation». [En línea]. Disponible en: http://docs.identityserver.io/en/latest/topics/startup.html. [Accedido: 14-ene-2020].
[14] N. Nedjah y L. de M. Mourelle, *Embedded Cryptographic Hardware: Methodologies and Architectures*. Nova Publishers, 2004.
[15] «Wildcard Detected in Domain Portion of Content Security Policy (CSP) Directive | Netsparker». [En línea]. Disponible en: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/wildcard-detected-in-domain-portion-of-content-security-policy-csp-directive/. [Accedido: 20-ene-2020].
[16] «CSP: style-src», *MDN Web Docs*. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/style-src. [Accedido: 20-ene-2020].
[17] «TLS cookie without secure flag set». [En línea]. Disponible en: https://portswigger.net/kb/issues/00500200_tls-cookie-without-secure-flag-set. [Accedido: 20-ene-2020].
[18] «Anti-CSRF Tokens para prevenir Cross Site Request Forgery (CSRF) - Backtrack Academy». [En línea]. Disponible en: https://backtrackacademy.com/articulo/anti-csrf-tokens-para-prevenir-cross-site-request-forgery-csrf. [Accedido: 20-ene-2020].
[19] «Cache-Control», *Documentación web de MDN*. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Cache-Control. [Accedido: 20-ene-2020].
[20] «Defining Resources — IdentityServer4 1.0.0 documentation». [En línea]. Disponible en:

http://docs.identityserver.io/en/latest/topics/resources.html. [Accedido: 14-ene-2020].
[21] «Sign-in — IdentityServer4 1.0.0 documentation». [En línea]. Disponible en: http://docs.identityserver.io/en/latest/topics/signin.html. [Accedido: 14-ene-2020].
[22] «SHA-2», *Wikipedia, la enciclopedia libre*. 04-dic-2019.
[23] R. Wang, S. Chen, y X. Wang, «Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services», en *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 365-379, doi: 10.1109/SP.2012.30.
[24] W. Li, C. J. Mitchell, y T. Chen, «OAuthGuard: Protecting User Security and Privacy with OAuth 2.0 and OpenID Connect», *ArXiv190108960 Cs*, ene. 2019.