



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

— **TELECOM** ESCUELA
TÉCNICA **VLC** SUPERIOR
DE INGENIERÍA DE
TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de
Telecomunicación

Gestión de Identidades en Cloud con Entra ID y Análisis de
Flujos OAuth 2.0

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de
Telecomunicación

AUTOR/A: Navarro Rueda, Pablo

Tutor/a: López Patiño, José Enrique

CURSO ACADÉMICO: 2023/2024

Resumen

En el contexto actual del mundo empresarial, marcado por una creciente migración hacia entornos cloud y el abandono gradual de las infraestructuras tradicionales de centro de datos, la gestión efectiva de identidades y accesos se presenta como un desafío crucial para las organizaciones y los profesionales de seguridad.

Este trabajo de fin de grado pretende abordar el reto comenzando por una introducción al entorno cloud mostrando la evolución de las organizaciones y su transición de una gestión de las identidades en directorio activo a realizarla a través de Entra ID. Se llevará a cabo un análisis detallado de las diversas herramientas que Entra ID proporciona, explorando sus funcionalidades y su aplicación para lograr una gestión eficaz de identidades, abordando aspectos claves como la seguridad.

Finalmente, estudiaremos los diferentes flujos de autenticación OAuth 2.0 que existen clasificándolos en base a su seguridad. Concluiremos examinando exhaustivamente un caso de uso: una aplicación que emplea un flujo considerado no seguro para acceder a datos de riesgo o de alta seguridad de la organización. Con base en este análisis, se propondrán diversos métodos y estrategias que aprovechen las herramientas ofrecidas por Entra ID para mitigar adecuadamente estos riesgos y fortalecer la seguridad de las identidades en el entorno cloud.

Palabras clave: ciberseguridad; Entra ID ; Azure ; computación en nube ; gestión de identidades ; OAuth ; Acceso seguro.

Resum

En l'actual escenari empresarial, marcat per una creixent migració cap a entorns cloud i l'abandonament gradual de les infraestructures tradicionals de centres de dades, la gestió efectiva d'identitats i accessos es presenta com un repte crucial per a les organitzacions i els professionals de la seguretat.

Aquest treball de fi de grau pretén abordar aquest repte començant per una introducció a l'entorn cloud, mostrant l'evolució de les organitzacions i la seua transició d'una gestió d'identitats en Directori Actiu a realitzar-la a través d'Entra ID. Es durà a terme una anàlisi detallada de les diverses eines que proporciona Entra ID, explorant les seues funcionalitats i la seua aplicació per a aconseguir una gestió eficaç d'identitats, abordant aspectes clau com la seguretat.

Finalment, estudiarem els diferents fluxos d'autenticació OAuth 2.0 que existeixen, classificant-los en funció de la seua seguretat. Conclourem examinant de manera exhaustiva un cas d'ús: una aplicació que utilitza un flux considerat no segur per a accedir a dades de risc o de seguretat alta de l'organització. A partir d'aquesta anàlisi, es proposaran diversos mètodes i estratègies que aprofiten les eines oferides per Entra ID per a mitigar adequadament aquests riscos i enfortir la seguretat de les identitats en l'entorn cloud.

Paraules clau: ciberseguretat; Entra ID; Azure; computació en núvol; gestió d'identitats; OAuth; accés segur.



Abstract

In the current business landscape, marked by an increasing migration to cloud environments and the gradual abandonment of traditional data center infrastructures, effective identity and access management is emerging as a crucial challenge for organizations and security professionals.

This final degree project aims to address this challenge by starting with an introduction to the cloud environment, showcasing the evolution of organizations and their transition from managing identities in Active Directory to doing so through Entra ID. A detailed analysis of the various tools provided by Entra ID will be conducted, exploring their functionalities and application for achieving effective identity management, addressing key aspects such as security.

Finally, we will study the different OAuth 2.0 authentication flows, classifying them based on their security. We will conclude by thoroughly examining a use case: an application that employs a flow considered insecure to access high-risk or high-security data within the organization. Based on this analysis, various methods and strategies will be proposed that leverage the tools offered by Entra ID to adequately mitigate these risks and strengthen identity security in the cloud environment.

Keywords: cybersecurity; Entra ID; Azure; cloud computing; identity management; OAuth; secure access.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la ingeniería de telecomunicación

CONCEPT (ABET)	CONCEPTO (traducción)	¿Cumple? (S/N)	¿Dónde? (páginas)
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	11-31
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	1
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	32 - 45
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	46 - 49
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	51
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	51



Índice

Capítulo 1.	Introducción	1
1.1	Motivación.....	1
1.2	Objetivos.....	1
1.3	Estructura de la memoria	1
1.4	NTT DATA	2
Capítulo 2.	Entorno cloud.....	3
2.1	Beneficios de la migración al cloud.....	3
2.1.1	Alta disponibilidad	3
2.1.2	Escalabilidad	3
2.1.3	Confiabilidad	3
2.1.4	Previsibilidad.....	3
2.1.5	Seguridad.....	4
2.1.6	Gestión de los recursos.....	4
2.2	Modelos de implementación.....	4
2.2.1	Privada.....	4
2.2.2	Pública.....	4
2.2.3	Híbrida.....	4
2.2.4	Multicloud	4
2.3	Modelos de servicio.....	5
2.3.1	Infrastructure as a Service (IaaS).....	5
2.3.2	Platform as a Service (PaaS)	6
2.3.3	Software as a Service (SaaS).....	6
2.3.4	Modelo de responsabilidad compartida.....	6
2.4	Principales proveedores de cloud computing	7
Capítulo 3.	Evolución de la gestión de identidades	8
3.1	Identidad digital	8
3.2	Identity Provider (IdP)	8
3.3	Active Directory	8
3.4	Entra ID	9
Capítulo 4.	Entra ID y Gestión de Identidades en un entorno Cloud.....	11
4.1	Funcionamiento	11
4.2	Identity and Access Management (IAM).....	11



4.2.1	Usuarios.....	11
4.2.2	Grupos	12
4.2.3	Roles.....	13
4.2.4	Registro de dispositivos e IPs.....	14
4.3	Gestión de aplicaciones	14
4.3.1	App Registrations	15
4.3.2	Enterprise Applications	15
4.4	Herramientas de seguridad.....	15
4.4.1	Multi-Factor Authentication (MFA).....	15
4.4.2	Privileged Identity Management (PIM).....	16
4.4.3	Risk Detections.....	17
4.4.4	Self Service Password Reset (SSPR)	18
4.4.5	Identity Secure Score.....	18
4.4.6	Herramientas de análisis de logs	18
4.4.7	Conditional Access.....	19
Capítulo 5.	Flujos OAuth 2.0.....	22
5.1	OAuth 2.0 y OpenID Connect	22
5.2	Funcionamiento de un flujo OAuth 2.0	22
5.2.1	Refresh token.....	23
5.2.2	Solicitud del token.....	24
5.2.3	Extensión flows	25
5.3	Tipos de flujo.....	25
5.3.1	Authorization Code Flow	26
5.3.2	Authorization Code Flow with PKCE.....	27
5.3.3	Client Credentials Flow	28
5.3.4	Implicit Flow	28
5.3.5	Resource Owner Password Flow.....	29
5.3.6	Device Authorization Flow	30
Capítulo 6.	Caso de uso	32
6.1	Integración con Implicit Flow.....	32
6.1.1	Creación de la aplicación en Entra ID	32
6.1.2	Código	34
6.1.3	Funcionamiento	37
6.1.4	Problemas de la implementación.....	39
6.1.5	Análisis del token	40
6.2	Transición a Authorization Code Flow.....	42



6.2.1	Cambios en Entra ID	42
6.2.2	Cambios en el código	43
6.2.3	Funcionamiento	44
6.2.4	Análisis de la securización	45
6.3	Proteger la identidad y los accesos a la aplicación	46
6.3.1	Bloquear acceso salvo a usuarios asignados.....	46
6.3.2	Definir política de Conditional Access para el bloqueo	47
6.3.3	Definir política de Conditional Access para solicitar MFA	49
Capítulo 7.	Conclusiones	51
7.1	Conclusiones y trabajos futuros	51
7.2	Relación con los estudios cursados.....	51
Agradecimientos		52
Bibliografía		53
Anexos.....		55
Anexo I – Glosario de términos		55
Anexo II – Objetivos de Desarrollo Sostenible.....		57



Índice de ilustraciones

Ilustración 1 - Hybrid cloud y multicloud	5
Ilustración 2 - Modelos de servicio cloud	5
Ilustración 3 - Modelo de responsabilidad compartida	6
Ilustración 4 - Cuotas de Mercado, 4º trimestre 2023	7
Ilustración 5 - Entorno con Entra ID integrado	9
Ilustración 6 - Usuario en Entra ID	12
Ilustración 7 - Grupo en Entra ID	13
Ilustración 8 - Roles de Entra ID	13
Ilustración 9 - Funcionamiento conjunto de las áreas de la GID	14
Ilustración 10 - Métodos de autenticación soportados por Entra ID	16
Ilustración 11 - Análisis de PIM	16
Ilustración 12 - Análisis de los riesgos	17
Ilustración 13 - Identity Secure Score	18
Ilustración 14 - Sign-in logs	19
Ilustración 15 - Funcionamiento de Conditional Access	19
Ilustración 16 - Condiciones de aplicación de una política de Conditional Access	20
Ilustración 17 - Acciones que puede realizar una política de Conditional Access	20
Ilustración 18 - Uso conjunto de OpenID Connect y OAuth 2.0	22
Ilustración 19 - Funcionamiento general de un flujo OAuth 2.0	23
Ilustración 20 - Ejemplo de tokens	24
Ilustración 21 - Petición HTTP	24
Ilustración 22 - Flujos standard y extension	25
Ilustración 23 - Flujos clasificados según su seguridad	25
Ilustración 24 - Authorization Code Flow	26
Ilustración 25 - Authorization Code Flow with PKCE	27
Ilustración 26 - Client Credentials Flow	28
Ilustración 27 - Implicit Flow	29
Ilustración 28 - Resource Owner Password Flow	30
Ilustración 29 - Device Authorization Flow	31
Ilustración 30 - Creación de la App registration	33
Ilustración 31 - Configuración de la app registration para funcionar con Implicit Flow	33
Ilustración 32 - Configuración de los permisos de la API	34
Ilustración 33 - Página de login de Microsoft	38
Ilustración 34 - Página principal	38



Ilustración 35 - Llamada a Graph.....	39
Ilustración 36 - Captura con Wireshark del tráfico con Implicit Flow.....	40
Ilustración 37 - Creación de un client secret desde Entra ID	42
Ilustración 38 - Configuración para Implicit Flow.....	43
Ilustración 39 - Página de inicio nueva	45
Ilustración 40 - Captura de Wireshark con Authorization Code Flow	46
Ilustración 41 - Bloqueo a usuario no asignado a la app.....	47
Ilustración 42 - Creación de la política de bloqueo.....	48
Ilustración 43 - Mensaje de bloqueo por política de Conditional Access	48
Ilustración 44 - Creación de política para requerir MFA	49
Ilustración 45 - MFA de Authenticator App requerido.....	50

Capítulo 1. Introducción

1.1 Motivación

Este proyecto surge del creciente interés por la ciberseguridad y como forma de poner en prueba lo aprendido en el campo de la gestión de identidades durante mis prácticas en NTT Data. Durante este periodo he utilizado la herramienta Entra ID, de la mano de profesionales en la materia que me han transmitido su entusiasmo y ganas de continuar aprendiendo.

Este Trabajo de Fin de Grado es la mejor oportunidad para probar mi conocimiento sobre el funcionamiento de las herramientas utilizadas, su aplicación en la práctica y demostrar la importancia de asegurar la identidad digital en una aplicación web.

1.2 Objetivos

El objetivo principal del trabajo es profundizar en el ámbito de la gestión de identidades digitales en el cloud, centrándose en la importancia de securizar la identidad y el uso de un flujo OAuth 2.0 seguro. Se busca:

- Explorar en detalles las herramientas que ofrece Entra ID, comprendiendo su funcionalidad y utilidad.
- Analizar las características de los diferentes flujos OAuth 2.0 y las posibles vulnerabilidades en la seguridad que la elección de un flujo no recomendado puede provocar.
- Demostrar con un caso práctico cómo abordar estas vulnerabilidades y utilizando las herramientas de Entra ID proteger los accesos.
- Contribuir al conocimiento y comprensión de la importancia de la gestión de identidades, así como a la implementación de prácticas de seguridad sólidas para crear un entorno seguro.

Completando estos objetivos, no solo se busca ampliar el entendimiento teórico sobre la gestión de identidades, sino también proporcionar recomendaciones prácticas y soluciones concretas para abordar los desafíos de seguridad asociados con la autenticación y autorización en entornos cloud.

1.3 Estructura de la memoria

El trabajo consta de 7 capítulos:

- **Introducción.** Presentación general del proyecto dando un contexto inicial
- **Entorno Cloud.** Introducción a los fundamentos del entorno cloud
- **Evolución de la gestión de identidades.** Se explora como era la gestión de identidades antes de Entra ID y como ha sido la transición
- **Entra ID.** Introducción a las diferentes posibilidades de gestión que ofrece Entra gracias a sus diferentes herramientas
- **Flujos de OAuth 2.0.** Análisis de que es un flujo OAuth 2.0, su funcionamiento y distintos tipos.
- **Caso de uso.** Descripción detallada de un caso práctico en el que se integra una aplicación web con Entra ID, buscando que el proceso sea seguro.
- **Conclusiones.** Análisis de los resultados obtenidos.

Aparte, existen anexos con información relevante:

- **Glosario de términos.** Terminología técnica usada a lo largo de la memoria.

- **Objetivos de desarrollo sostenible.** Inclusión de información relevante sobre cómo el proyecto contribuye a los ODS.

1.4 NTT DATA

NTT Data es una empresa de servicios de telecomunicaciones y tecnología de la información con sede en Japón, que emplea a aproximadamente 300,000 personas en todo el mundo. Se especializa en sistemas de información y consultoría informática. En 2006, NTT Data adquirió Everis, una consultora española, para expandir su presencia en Europa, fusión que se completó en 2021, consolidando a NTT Data como una de las principales consultoras en España.

En 2024 la compañía ha sido reconocida como Global Top Employer por sus políticas y prácticas de gestión de personal a nivel mundial, obteniendo certificaciones en Asia Pacífico, Europa, América Latina y América del Norte. En España, ha sido galardonada como Top Employer por noveno año consecutivo. Estos reconocimientos muestran el compromiso de NTT Data con la innovación, la calidad y la satisfacción del cliente, posicionándola como líder en telecomunicaciones y tecnología de la información

Capítulo 2. Entorno cloud

El cloud computing ha cambiado completamente cómo las organizaciones acceden, almacenan y gestionan sus recursos informáticos. No es solo un simple servicio a través de Internet que ofrece infraestructura de TI tradicional, como máquinas virtuales, almacenamiento y bases de datos, sino también una amplia gama de innovaciones.

El cloud permite implementar y escalar tecnologías avanzadas como el Internet of Things (IoT), el Machine Learning y la inteligencia artificial (IA). Estas capacidades no solo fomentan la creatividad y la innovación, sino que también brindan a las organizaciones una gran cantidad de beneficios que les permiten explorar nuevos horizontes y resolver problemas complejos de manera más eficiente y efectiva.

2.1 Beneficios de la migración al cloud

La migración al cloud se ha convertido en una prioridad para numerosas organizaciones en los últimos años, que buscan actualizarse y aprovechar los beneficios de este entorno digital. A continuación, se exploran las ventajas fundamentales de la migración:

2.1.1 Alta disponibilidad

En un entorno cloud, la infraestructura está diseñada para minimizar el tiempo de inactividad de forma que se puede garantizar que los recursos estén accesibles en todo momento sin interrupciones. Proveedores de servicios cloud como Azure ofrecen garantías de tiempo de actividad a través de sus contratos de nivel de servicio, proporcionando a las organizaciones la seguridad de que sus servicios estarán disponibles incluso durante eventos imprevistos, pudiendo así operar con confianza y asegurar la continuidad del negocio.

2.1.2 Escalabilidad

Los recursos en un entorno cloud son escalables, es decir, se pueden ajustar dinámicamente para satisfacer las demandas cambiantes. Esto permite a las organizaciones manejar picos de tráfico sin comprometer el rendimiento y optimiza los costos mediante el modelo "Pay-as-you-go", donde solo se paga por los recursos utilizados, eliminando los gastos derivados de recursos contratados no necesarios.

2.1.3 Confiabilidad

La confiabilidad se refiere a la capacidad de un sistema para recuperarse de errores y seguir funcionando de manera continua. La infraestructura es resistente gracias a su diseño descentralizado, que distribuye los recursos en múltiples regiones globales de forma que durante una caída de un centro de datos, otros centros pueden seguir operando ya que en algunos casos, el sistema cambia automáticamente a otra región para asegurar la continuidad del servicio sin intervención manual.

2.1.4 Previsibilidad

La previsibilidad permite una planificación y ejecución confiables de proyectos, asegurando operaciones eficientes y rentables. La previsibilidad del rendimiento anticipa los recursos necesarios para ofrecer una experiencia positiva a los clientes y la previsibilidad de costos permite pronosticar y controlar el gasto en cloud computing gracias al seguimiento en tiempo real y el análisis de datos ayudan a identificar patrones y planificar mejor las implementaciones de recursos.

2.1.5 Seguridad

Este entorno adopta un modelo de responsabilidad compartida entre el proveedor del servicio cloud y el cliente, proporcionando una defensa más efectiva y focalizada. Además, con este modelo se asegura una respuesta ágil frente a vulnerabilidades y permite personalizar la seguridad según requisitos específicos. Al dividir las responsabilidades, el proveedor asegura la infraestructura del cloud contra ataques, mientras el cliente se enfoca en proteger sus datos y gestionar el acceso a sus aplicaciones.

2.1.6 Gestión de los recursos

Las herramientas y servicios integrados permiten una supervisión continua en tiempo real, asegurando una operatividad óptima y la capacidad de responder rápidamente a cualquier incidencia. Se logran identificar automáticamente los recursos problemáticos e incluso iniciar procesos de reparación sin intervención manual.

2.2 Modelos de implementación

No existe un enfoque único de cloud computing que se adapte a todas las organizaciones. Debido a la diversidad y evolución de las necesidades empresariales han surgido varios modelos de implementación del cloud computing, cada uno diseñado para abordar desafíos y situaciones específicas.

2.2.1 Privada

Es una infraestructura exclusiva para una sola entidad, por lo que ofrece mayor control y privacidad en comparación con una implementación de cloud pública. Esto permite a las organizaciones personalizar sus recursos, proporcionando flexibilidad y adaptabilidad a los requisitos del negocio. La infraestructura puede estar alojada localmente o con un proveedor externo, pero siempre en una red privada dedicada, siendo ideal para entidades que requieren alta seguridad, como agencias gubernamentales y financieras.

2.2.2 Pública

En este modelo, los recursos informáticos, como servidores y almacenamiento, son propiedad y están gestionados por un proveedor de servicios cloud, que se encarga del mantenimiento, eliminando esa carga para las empresas. Además, la implementación pública ofrece escalabilidad casi ilimitada, con recursos disponibles a demanda para satisfacer necesidades empresariales en cualquier momento. La alta fiabilidad es una ventaja garantizada por una amplia red de servidores que aseguran la continuidad del servicio y la disponibilidad de los recursos.

2.2.3 Híbrida

Es una combinación de una infraestructura privada con una pública, que permite a las organizaciones mover datos y aplicaciones entre ambos entornos según sea necesario, brindando flexibilidad y opciones de implementación. Permite a las empresas mantener datos confidenciales en su propio centro de datos mientras ejecutan otras cargas de trabajo en un cloud público, logrando así una mayor seguridad. Este modelo es ideal para organizaciones con una infraestructura de IT existente, que prefieren no mover todas sus operaciones al cloud público debido a los costos y la complejidad, optando por la implementación híbrida para optimizar costos y operaciones.

2.2.4 Multicloud

Este último modelo implica el uso de múltiples servicios de diferentes proveedores, tanto públicos como privados. A diferencia de la implementación híbrida, que combina un cloud privado con

uno público, en el multicloud las empresas eligen los servicios más adecuados de distintos proveedores para tareas específicas o ubicaciones concretas. De esta forma se aprovechan beneficios como; la redundancia de datos y servicios, la optimización de costos y la capacidad de aprovechar las fortalezas de diferentes proveedores, sin embargo, también presenta desafíos en términos de integración, gestión y seguridad.

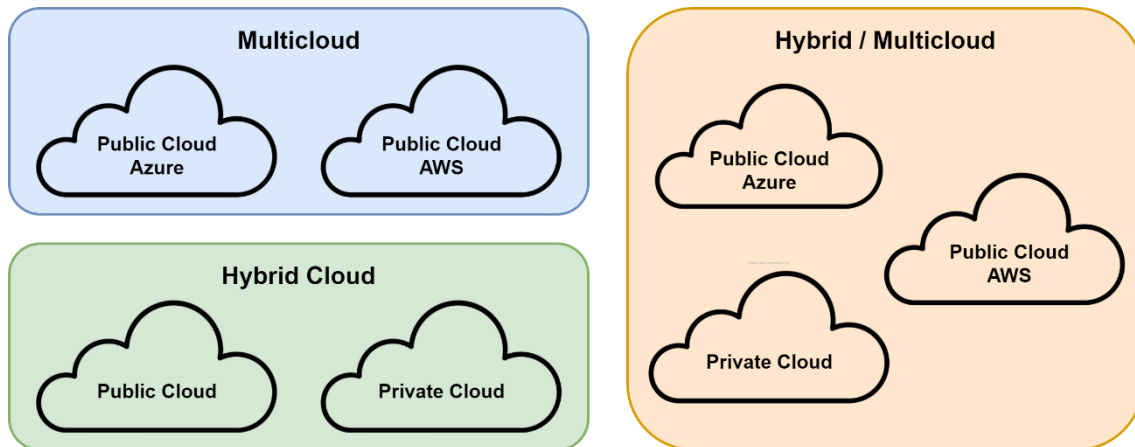


Ilustración 1 - Hybrid cloud y multicloud (Elaboración propia)

2.3 Modelos de servicio

Aparte de los modelos de implementación, existen los modelos de servicio cloud, que se clasifican según el nivel de flexibilidad y control que ofrecen al cliente.

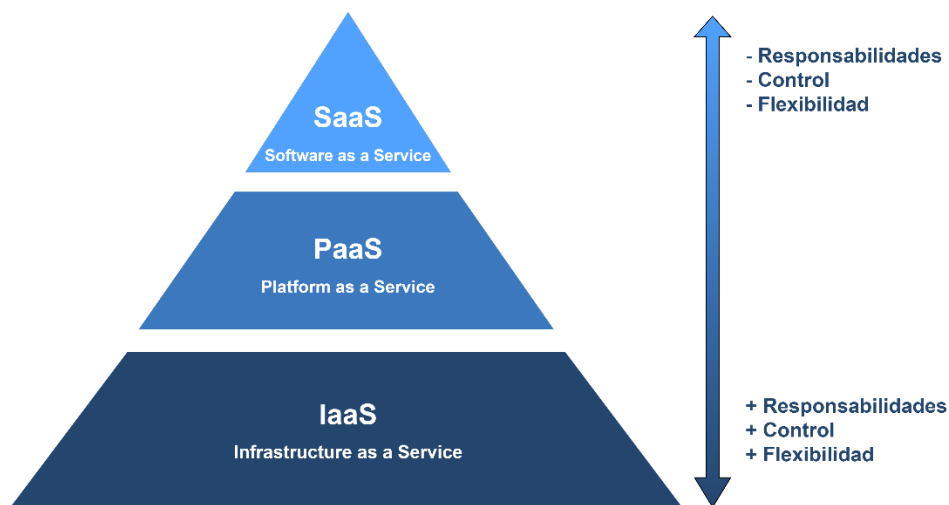


Ilustración 2 - Modelos de servicio cloud (Elaboración propia)

2.3.1 Infrastructure as a Service (IaaS)

IaaS es la categoría más flexible, proporcionando un alto grado de control sobre los recursos. El proveedor mantiene el hardware físico, la conectividad de red y la seguridad física, mientras que el cliente es responsable de la instalación y configuración del sistema operativo, la gestión de la red y el almacenamiento de datos.

Con IaaS, las organizaciones alquilan hardware de un centro de datos en el cloud, configurándolo según sus necesidades específicas, lo que permite implementar y escalar recursos de manera eficiente sin invertir en infraestructura física costosa. Las máquinas virtuales son un ejemplo clásico de IaaS, donde los usuarios tienen pleno control sobre su configuración y administración, instalando sistemas operativos, software y aplicaciones según sus necesidades.

2.3.2 Platform as a Service (PaaS)

PaaS representa un punto intermedio entre IaaS y SaaS. En este modelo, el proveedor gestiona la infraestructura física, la seguridad, la conectividad a Internet, los sistemas operativos, las herramientas de desarrollo y los servicios de inteligencia empresarial.

Una ventaja es que los usuarios no tienen que gestionar licencias ni aplicar actualizaciones ni parches, ya que, PaaS ofrece una plataforma lista para usar, permitiendo a los equipos de desarrollo crear, probar y desplegar aplicaciones rápidamente. Un ejemplo de PaaS es Microsoft Azure App Service, una plataforma completamente administrada que facilita la creación, implementación y escalado de aplicaciones web y móviles

2.3.3 Software as a Service (SaaS)

SaaS es el modelo más completo de servicios cloud desde la perspectiva del producto, puesto que se alquila o utiliza una aplicación completamente desarrollada. Un ejemplo conocido de SaaS es Office 365 donde los usuarios utilizan aplicaciones basadas en el cloud completamente desarrolladas.

Aunque este modelo es el menos flexible, es el más fácil de implementar ya que requiere la menor cantidad de conocimientos técnicos o experiencia. Los usuarios pueden acceder a las aplicaciones a través de Internet, sin necesidad de gestionar la infraestructura subyacente ni preocuparse por actualizaciones o mantenimiento, ya que el proveedor se encarga de todo esto.

2.3.4 Modelo de responsabilidad compartida

Tras comentar uno por uno los diferentes modelos de servicio, se puede observar que en un entorno cloud se sigue un modelo en el que las responsabilidades se distribuyen entre el proveedor y el cliente en función del modelo de servicio utilizado. Este enfoque permite una mejora en la seguridad y la protección de los datos, ya que cada parte se libera de ciertas responsabilidades lo que les permite centrarse en aquellas que les corresponden.

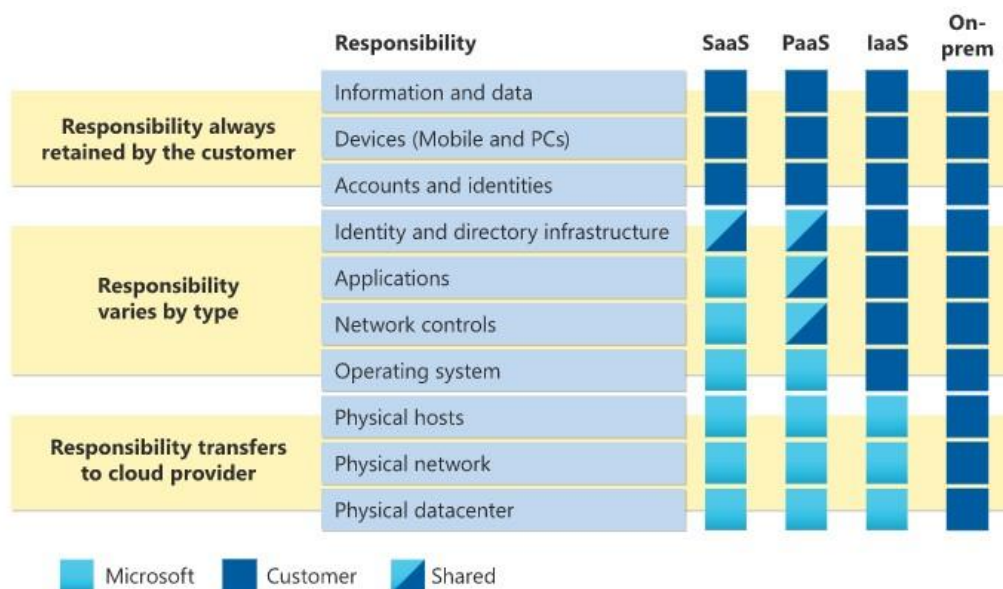


Ilustración 3 - Modelo de responsabilidad compartida (por Microsoft Learn) [1]

En la ilustración 3 se muestran las áreas de responsabilidad entre el cliente y proveedor según el tipo de implementación. Comenzando por la derecha, se observa que mientras que en un centro de datos local, el cliente es responsable de toda la infraestructura y la seguridad de los datos y recursos, al pasar a una infraestructura cloud, algunas responsabilidades se transfieren a Microsoft.

Independientemente del modelo de servicio, el cliente siempre será responsable de los datos, las identidades y la gestión de accesos al igual que el proveedor, siempre se encargará de la seguridad física de la infraestructura como los servidores y las redes. Luego dependiendo del modelo se reparten las responsabilidades de gestionar el sistema operativo, las aplicaciones, los datos y las identidades.

2.4 Principales proveedores de cloud computing

Actualmente son tres empresas las que dominan el mercado del cloud computing, y cada una ofrece una amplia gama de productos y servicios adaptados a diferentes necesidades de proyectos.

Amazon Web Services (AWS):

- Tienes la mayor cuota de mercado y número de características.
- Es ideal para empresas que necesitan una eficacia probada.
- Ofrece gran flexibilidad y facilidad de uso, permitiendo la selección de sistemas operativos, lenguajes de programación, plataformas de aplicaciones web, bases de datos y otros servicios según las necesidades.

Microsoft Azure:

- Posee la segunda mayor cuota de mercado, muy similar a la de AWS.
- Su ventaja principal en la integración y facilidad de uso para organizaciones ya integradas en el ecosistema de Microsoft.

Google Cloud Platform (GCP):

- Tiene menos cuota de mercado en comparación con AWS y Azure, sin embargo, su crecimiento es mayor.
- Destacado por servicios sólidos y rentables en análisis de datos, aprendizaje automático y computación de alto rendimiento.

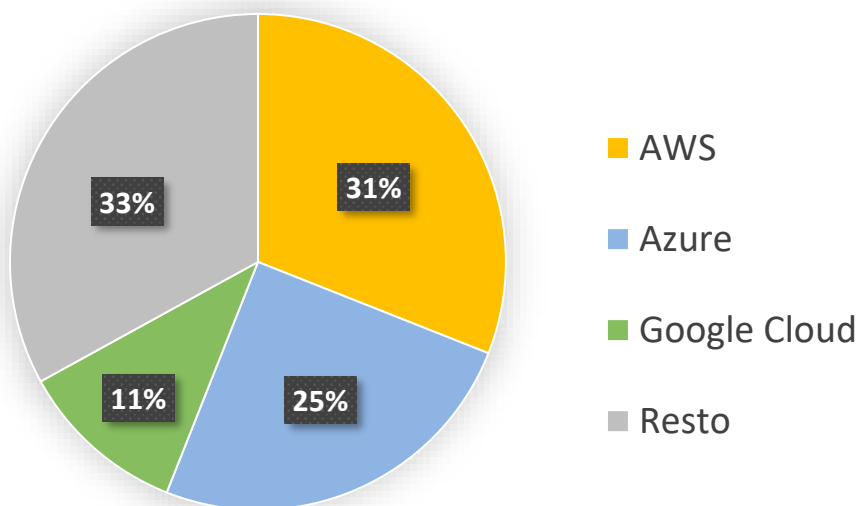


Ilustración 4 - Cuotas de Mercado, 4º trimestre 2023 (datos obtenidos de Statista) [2]

Capítulo 3. Evolución de la gestión de identidades

3.1 Identidad digital

La identidad digital es la representación única de un usuario, entidad o dispositivo en un sistema. Se compone de información y atributos que permiten identificar y autenticar a un sujeto o dispositivo dentro de un sistema, permitiendo que solo las entidades autorizadas tengan acceso a recursos específicos.

Entre estos atributos se pueden incluir el nombre de usuario, dirección de correo electrónico, credenciales de inicio de sesión, claves criptográficas, certificados digitales y otros datos utilizados para verificar la identidad de la entidad. Para cualquier sistema de gestión de identidades es un componente fundamental, ya que sirve como base para la creación de usuarios y la asignación de grupos y roles según sus características.

3.2 Identity Provider (IdP)

El IdP es una entidad que crea, mantiene, almacena y gestiona la información relativa a las identidades de los usuarios y que proporciona servicios de autenticación y autorización. Además se encargan de verificar las identidades de los usuarios mediante distintos factores, no solo verifican a los usuarios humanos, sino que pueden autenticar cualquier entidad conectada a una red o a un sistema, incluyendo ordenadores y otros dispositivos.

La función del IdP es realmente importante puesto que la identidad digital debe ser almacenada de manera segura, especialmente en un entorno cloud donde la identidad del usuario determina si alguien puede o no acceder a datos confidenciales.

Se puede diferenciar entre dos tipos de IdP:

IdP local:

- Active Directory
- RADIUS
- LDAP

IdP cloud:

- Entra ID
- Okta
- Google Workspace

Durante los últimos años, se ha observado una tendencia creciente de llevar la gestión de identidades a entornos cloud puesto que cada vez son más las organizaciones que buscan aprovechar las ventajas ofrecidas por este entorno. Esto provoca que cada vez sea más común entre las organizaciones pasar de IdPs como Active Directory a Entra ID. Muchas organizaciones que se encuentran en medio de la transición de un entorno a otro optan por mantener un modelo híbrido donde conviven ambos IdPs.

3.3 Active Directory

Windows Active Directory (AD) es el IdP en local de Microsoft, esencial para la gestión de identidades en entornos de red basados en Windows. Este sistema organiza y gestiona la información a través de una estructura jerárquica que incluye cuentas de usuarios, equipos y otros recursos, facilitando su acceso y administración.

AD optimiza la gestión de identidades almacenando datos detallados sobre los objetos de la red y proporcionando robustos servicios para su eficiente búsqueda y administración. La seguridad se refuerza mediante la autenticación de inicio de sesión y control de acceso.

Dentro de sus herramientas clave para una gestión eficaz se encuentran:

- **Esquema:** Define las estructuras y reglas para la creación y gestión de objetos dentro del directorio.
- **Catálogo Global:** Centraliza la información de todos los objetos del directorio, facilitando el acceso y la búsqueda a través de diferentes dominios.
- **Mecanismo de Consulta e Índice:** Mejora la visibilidad y el acceso a los recursos de la red mediante un sistema eficiente de publicación y búsqueda.
- **Servicio de Replicación:** Garantiza una copia completa y actualizada de los datos del directorio en todos los controladores de dominio, asegurando la consistencia en toda la organización. [3]

3.4 Entra ID

Microsoft Entra ID puede considerarse como una extensión de AD pero en un entorno cloud. Al igual que AD, ofrece capacidades de administración de identidades y acceso, pero al estar basado en la nube, los usuarios pueden acceder a recursos y servicios desde cualquier lugar y en cualquier momento, sin estar limitados por la infraestructura local de la red.

Una de las ventajas que aporta Entra ID es inicio de sesión único (SSO) que permite a los usuarios utilizar una única identidad para acceder a una amplia gama de recursos, ya sean aplicaciones en la nube, servicios internos o incluso aplicaciones móviles. Con esto se simplifica la experiencia de inicio de sesión y aumenta la productividad a la par que la seguridad, puesto que no es necesario realizar múltiples inicios de sesión y por lo tanto dificulta al atacante el robo de credenciales.

Aparte Entra ID ofrece características avanzadas de seguridad, como la autenticación multifactor (MFA) y el monitoreo de la actividad de inicio de sesión, para proteger los datos y recursos empresariales contra las amenazas cibernéticas. Con estas herramientas, los administradores pueden mantener un control granular sobre quién tiene acceso a qué recursos y aplicar políticas de seguridad consistentes en toda la infraestructura de TI, tanto local como en el cloud.

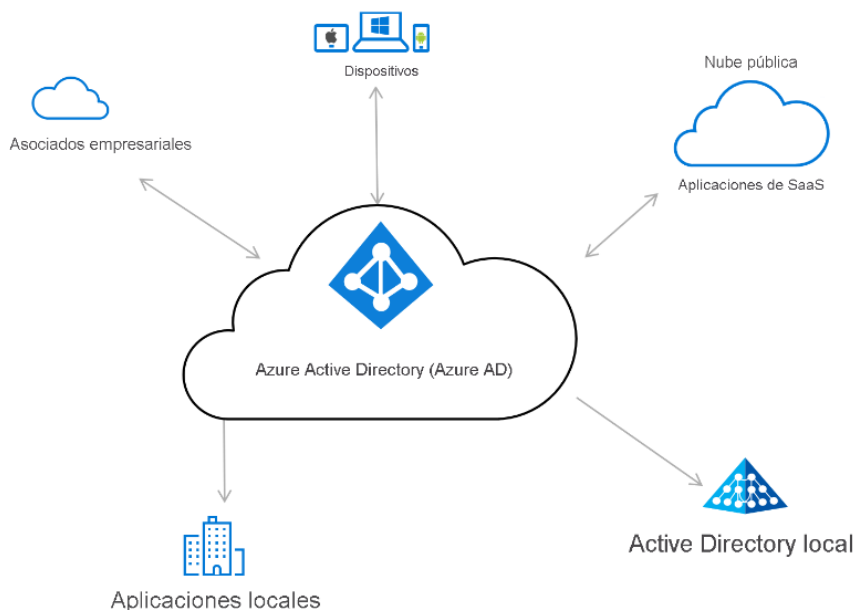


Ilustración 5 - Entorno con Entra ID integrado (por Microsoft Learn 2024)



Tal y como se observa en la ilustración 5, un entorno que incorpora Entra ID como IdP permite integrar aplicaciones tanto locales como SaaS, a las cuales tal y como se ha comentado se accederá mediante un único acceso gracias al SSO. Además, los dispositivos pueden ser registrados e hibridados, permitiendo un control unificado de seguridad y acceso. También es posible establecer un entorno híbrido donde AD se conecte con Entra ID, logrando que los usuarios registrados on-premise se sincronicen automáticamente en el cloud. Con este modelo lo que se consigue es una gestión centralizada de identidades y accesos, mejorando la seguridad y la eficiencia operativa al unificar la administración en un solo punto.

Capítulo 4. Entra ID y Gestión de Identidades en un entorno Cloud

Tras establecer la base sobre la evolución de la gestión de identidades y la transición de AD a Entra ID, es momento de realizar un análisis más profundo de las características, herramientas y funcionalidades que Entra ID ofrece. A continuación se va a comentar desde las licencias necesarias para utilizar ciertas herramientas, la gestión de usuarios y la organización en grupos, hasta la implementación de políticas de seguridad avanzadas y la administración de aplicaciones. El objetivo es mostrar como Entra ID impulsa la seguridad, la eficiencia y la flexibilidad en el entorno empresarial moderno.

4.1 Funcionamiento

Entra ID ofrece una variedad de licencias que amplían las funcionalidades estándar del directorio gratuito y proporcionan acceso a características avanzadas. Lo que se busca es que a través de estas licencias los usuarios puedan beneficiarse del autoservicio, una supervisión mejorada, informes de seguridad y un acceso seguro para los usuarios móviles, entre otras mejoras significativas.

Entre las licencias más relevantes se encuentran:

- **Licencia de Microsoft Entra ID P1 o P2:** Permiten acceder a un conjunto más amplio de características, por ejemplo esta licencia es necesaria para utilizar funcionalidades como Privileged Identity Management (PIM) que se estudiará más adelante.
- **Licencias de Microsoft 365, Office 365 y Windows:** Se asignan a usuarios o grupos de para otorgarles acceso a productos de Office o Windows y es necesaria una por cada usuario que necesite acceso.
- **Licencia de usuario activo mensual (MAU):** Esta licencia se emplea con identidades externas de Entra ID y permite examinar a los usuarios externos que inician sesión y generar informes mensuales para fines de facturación, lo que facilita el seguimiento de los accesos y los costos asociados.

4.2 Identity and Access Management (IAM)

A continuación, se explora en detalle cómo Entra ID aborda la gestión de usuarios, la organización en grupos, la asignación de roles y el registro de dispositivos e IPs, aspectos cruciales para garantizar un buen control de los accesos y proteger la identidad.

4.2.1 Usuarios

En Entra ID, un usuario es una identidad digital creada para representar al individuo en el sistema y gestionar la autenticación y autorización en diversos recursos. Los usuarios pueden ser de 2 tipos:

- **Miembros:** Son usuarios que pertenecen a la misma organización y tienen permisos para acceder a la mayoría de la información del directorio.
- **Invitados:** Son aquellos usuarios que provienen de otros directorios, y son invitados de colaboración B2B (Business to Business). Tienen permisos restringidos en el directorio y suelen ser colaboradores externos o proveedores que necesitan acceso a recursos específicos, pero por seguridad no a toda la infraestructura ni a los aspectos relacionados con la facturación.

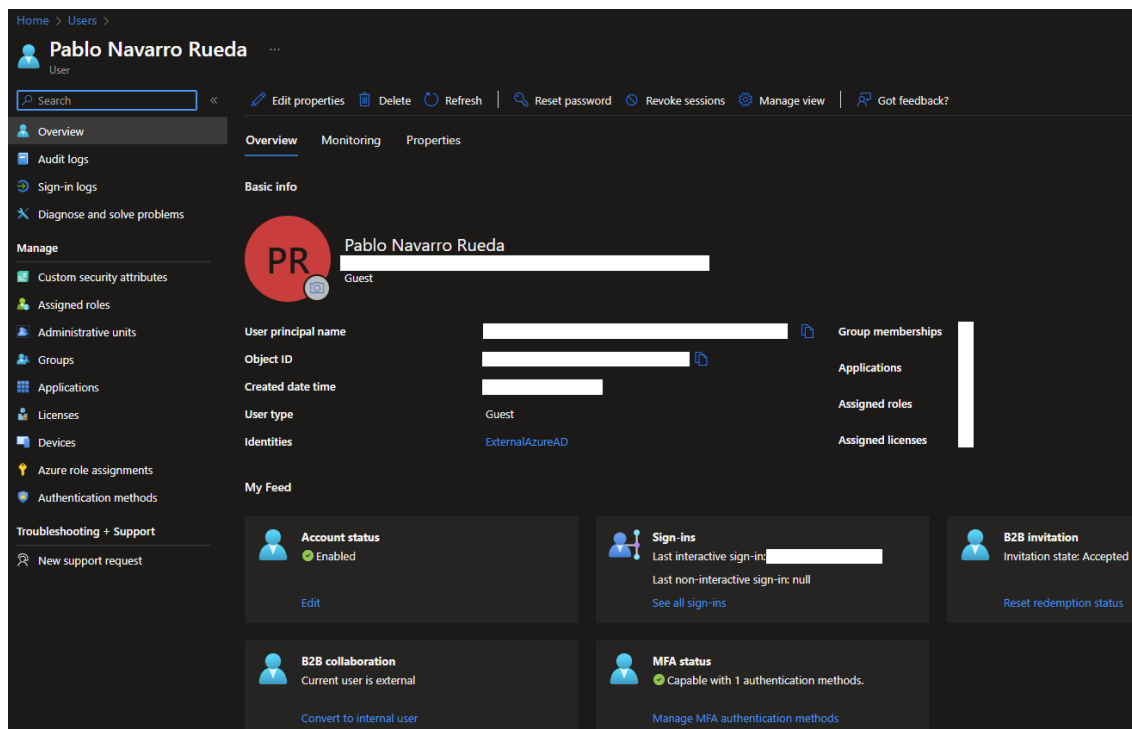


Ilustración 6 - Usuario en Entra ID (Elaboración propia)

4.2.2 Grupos

Los grupos son una herramienta clave para administrar el acceso a recursos, aplicaciones y tareas de manera eficiente. Permiten que en lugar de gestionar individualmente a cada usuario, estos puedan ser añadidos a un grupo y manejar a todos los usuarios que contenga el grupo de manera conjunta.

Según la naturaleza del grupo se pueden distinguir dos tipos:

- **Grupos de seguridad:** Estos grupos pueden estar formados por usuarios, dispositivos, servicios y otros grupos anidados y se utilizan para gestionar el acceso de los miembros a recursos compartidos. Por ejemplo, se puede utilizar un grupo para asignar un conjunto específico de permisos a todos sus miembros.
- **Grupos de Microsoft 365:** Destinados a la colaboración a través de servicios como Outlook, OneDrive, Planner y Teams.

Y de forma independiente a su naturaleza, los grupos pueden ser de dos tipos en base a la forma en que se añade a los usuarios:

- **Grupos estáticos:** Los miembros son asignados manualmente por un administrador y permanecen en el grupo hasta que son eliminados. Son útiles cuando la membresía del grupo no cambia con frecuencia.
- **Grupos dinámicos:** Los miembros son asignados de forma automática por reglas basadas en atributos de los usuarios, como su departamento, ubicación o rol. Cuando alguno de los atributos de un usuario cambia, el sistema evalúa las reglas del grupo para determinar si el usuario debe ser agregado o eliminado automáticamente. Son útiles cuando la membresía del grupo cambia con frecuencia.

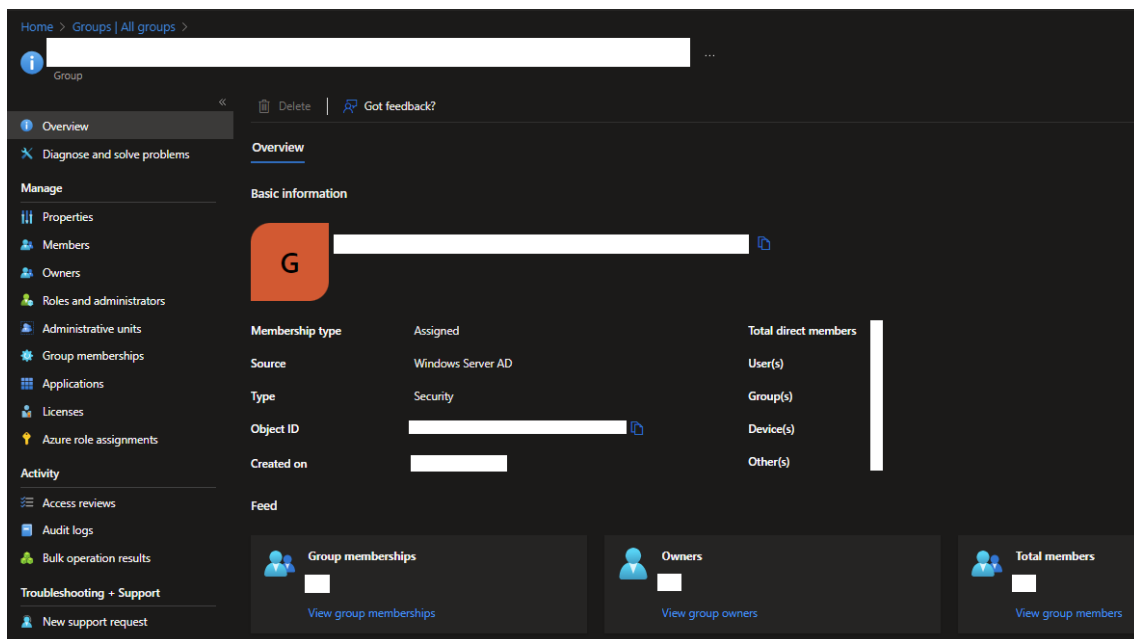


Ilustración 7 - Grupo en Entra ID (Elaboración propia)

4.2.3 Roles

Un rol, representa un conjunto de permisos que determinan las acciones que un usuario puede realizar dentro del entorno de Entra ID. Cuando un usuario necesita interactuar con recursos específicos, se le asigna un rol que define sus capacidades y limitaciones en función de los permisos asociados.

Existen dos categorías principales de roles en Entra ID:

- **Roles integrados:** Son alrededor de 60 roles predefinidos por Microsoft y que no pueden ser modificados. Cada uno de estos roles tiene un conjunto fijo de permisos asignados, lo que permite realizar tareas específicas dentro del entorno.
- **Roles personalizados:** Son roles creados y gestionados por los administradores del entorno. Esto brinda la flexibilidad de definir roles específicos que se ajusten mejor a las necesidades y estructura de la organización, asignando al nuevo rol los permisos exactos que se desean. Su utilidad es evitar que un usuario necesite de varios roles de forma simultánea para realizar sus tareas diarias.

Role	Description	Privileged	Assignments	Type
<input type="checkbox"/> Service Support Administrator	Can read service health information and manage support tickets.			Built-in
<input type="checkbox"/> SharePoint Administrator	Can manage all aspects of the SharePoint service.			Built-in
<input type="checkbox"/> Global Reader	Can read everything that a Global Administrator can, but not update anything.	PRIVILEGED		Built-in
<input type="checkbox"/> Security Reader	Can read security information and reports in Microsoft Entra ID and Microsoft 365.	PRIVILEGED		Built-in
<input type="checkbox"/> Directory Readers	Can read basic directory information. Commonly used to grant directory read access to applications and guests.			Built-in
<input type="checkbox"/> Insights Business Leader	Can view and share dashboards and insights via the M365 Insights app.			Built-in
<input type="checkbox"/> User Administrator	Can manage all aspects of users and groups, including resetting passwords for limited admins.	PRIVILEGED		Built-in
<input type="checkbox"/> Reports Reader	Can read sign-in and audit reports.			Built-in
<input type="checkbox"/> Exchange Administrator	Can manage all aspects of the Exchange product.			Built-in
<input type="checkbox"/> Security reader - Sign In Reports				Custom
<input type="checkbox"/> Groups Administrator	Members of this role can create/manage groups, create/manage groups settings like naming and expiration policies, and view groups activity and audit reports.			Built-in
<input type="checkbox"/> Insights Analyst	Access the analytical capabilities in Microsoft Viva Insights and run custom queries.			Built-in
<input type="checkbox"/> Global Administrator	Can manage all aspects of Microsoft Entra ID and Microsoft services that use Microsoft Entra identities.	PRIVILEGED		Built-in
<input type="checkbox"/> Cloud Device Administrator	Limited access to manage devices in Microsoft Entra ID.	PRIVILEGED		Built-in
<input type="checkbox"/> Intune Administrator	Can manage all aspects of the Intune product.	PRIVILEGED		Built-in
<input type="checkbox"/> Cloud Application Administrator	Can create and manage all aspects of app registrations and enterprise apps except App Proxy.	PRIVILEGED		Built-in

Ilustración 8 - Roles de Entra ID (Elaboración propia)

4.2.4 Registro de dispositivos e IPs

Como ya se ha comentado, Entra ID permite no solo el registro de usuarios sino también de dispositivos. Esto se puede hacer registrándolos directamente en la plataforma o mediante una integración híbrida. Este proceso de registro permite a los dispositivos conectarse y autenticarse de manera segura en el entorno, lo que facilita su administración y control por parte de los administradores.

También se permite registrar lo que se conoce como “named locations” que son grupos de IPs que se establecerán como de confianza, de forma que al analizar los inicios de sesión de los usuarios están serán fácilmente reconocibles.

Mas en adelante se profundizará en como estas dos funcionalidades aportan una mayor seguridad y control en el entorno al funcionar en conjunto con la herramienta de Conditional Access

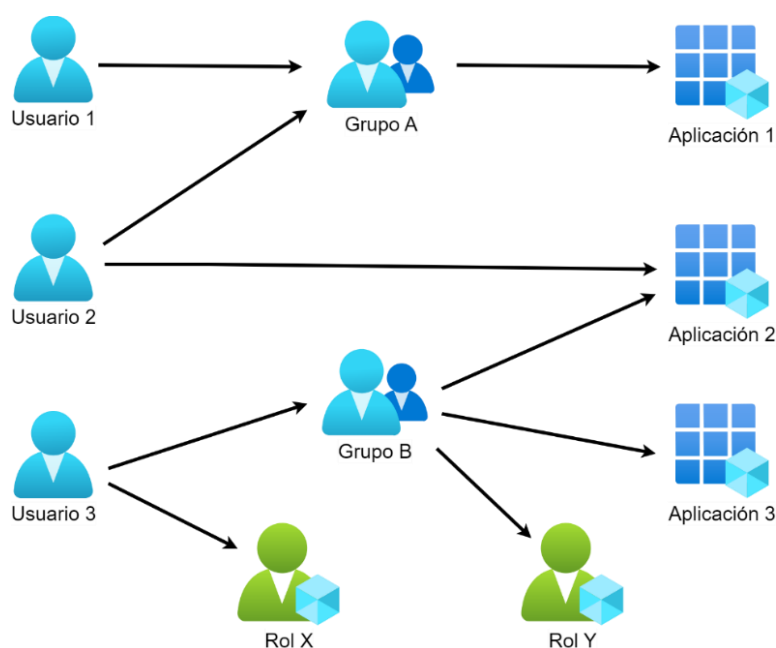


Ilustración 9 - Funcionamiento conjunto de las áreas de la GID (Elaboración propia)

Analizando la ilustración 9 se puede obtener una visión global del funcionamiento de las áreas estudiadas en conjunto. En esta representación, se observa que el Usuario 1 está asignado al Grupo A, lo que le confiere acceso exclusivo a la Aplicación 1, diseñada específicamente para los miembros de dicho grupo. Por otro lado, el Usuario 2, también perteneciente al Grupo A, cuenta además con una asignación individual a la Aplicación 2, lo que le permite acceder tanto a la Aplicación 1 como a la 2. Finalmente, todos los usuarios del Grupo B tienen acceso a las Aplicaciones 2 y 3, y están dotados del Rol Y. En el caso del Usuario 3, quien pertenece al Grupo B, también dispone del Rol X de manera individual, lo que amplía sus permisos de acceso dentro del sistema.

4.3 Gestión de aplicaciones

Al registrar una aplicación, se configura la aplicación para reconocer a Entra ID como su IDP, por lo que autenticación y autorización son delegadas al IdP. Esto permite a la aplicación utilizar los servicios de autenticación y autorización que proporciona Entra ID para gestionar el acceso de los usuarios, por lo que es fundamental establecer una conexión segura y eficiente entre ambos. Además, simplifica la experiencia del usuario al ofrecer un único punto de acceso para todas las aplicaciones registradas.

Una vez registrada y configurada, la aplicación redirige a los usuarios a un portal de inicio de sesión, donde se autentican utilizando sus cuentas dadas de alta en el IdP, en este caso Entra ID. Tras una autenticación exitosa, Entra ID indicará a la aplicación si el usuario autenticado está autorizado para acceder a esos recursos, basándose no solo en la validez de las credenciales, sino también en los permisos y roles asignados al usuario.

4.3.1 App Registrations

App Registration es el proceso de registrar formalmente una aplicación que emplea para la autenticación el protocolo Open ID Connect en un tenant específico de Entra ID, de forma que el registro actúe como una representación de la aplicación dentro del entorno, facilitando la comunicación y conectividad con los servicios proporcionados por este IdP. Una vez creada la App Registration, se asigna un identificador único, conocido como Client ID, exclusivo para el tenant donde se registra la aplicación.

Desde el menú de App Registration el administrador de la aplicación puede gestionar diferentes aspectos claves, como los secretos y certificados que añaden a la aplicación una capa extra de seguridad. También desde aquí se configuran aspectos clave como las URLs de respuesta y de cierre de sesión, los permisos y el acceso a las APIs necesarias. Estos permisos, conocidos como scopes, determinan qué recursos puede solicitar y acceder la aplicación en nombre del usuario, garantizando así un acceso controlado y seguro a los recursos protegidos.

4.3.2 Enterprise Applications

Una Enterprise App es la manifestación operativa de esa aplicación dentro del tenant, que usa SAML como protocolo de autenticación y autorización. La Enterprise App está estrechamente relacionada con un App Registration y un Service Principal, mientras que una App Registration define la aplicación en sí, proporcionando un identificador único (Client ID) y configuraciones de permisos y secretos, la Enterprise App es la manifestación operativa de esa aplicación dentro del tenant.

El Service Principal actúa como la identidad de la Enterprise App dentro del tenant. Define los permisos y roles específicos que la Enterprise App tendrá, permitiendo a los administradores gestionar cómo interactúa la aplicación con otros recursos y servicios.

Desde el menú de Enterprise Apps el administrador de la aplicación puede definir que usuarios tendrán acceso a la app y los roles de los que dispondrán. Además puede acceder a los intentos de inicio de sesión de los usuarios en la aplicación para mantener un control.

4.4 Herramientas de seguridad

A continuación se van a analizar las herramientas de seguridad que incorpora Entra ID, tanto las que se encargan de fortalecer y proteger el entorno como las que permiten a los administradores del entorno investigar las posibles amenazas.

4.4.1 Multi-Factor Authentication (MFA)

El MFA requiere que los usuarios verifiquen su identidad utilizando múltiples métodos de autenticación de forma que añade una capa más de seguridad. Cuando para un acceso se requiere que el usuario realice un MFA, además de ingresar su contraseña, el usuario debe proporcionar una segunda forma de autenticación mediante alguno de los métodos permitidos por Entra ID. De esta forma se hace más difícil para los atacantes obtener acceso a los recursos de la organización, incluso si han comprometido la contraseña de un usuario. Es una medida simple pero efectiva que ayuda a proteger a las organizaciones sobre todo contra ataques de phishing o situaciones donde sus contraseñas pueden ser robadas.

Method	Target	Enabled
▼ Built-In		
FIDO2 security key		
Microsoft Authenticator	All users	
SMS	All users	
Temporary Access Pass	2 groups	
Hardware OATH tokens (Preview)		
Third-party software OATH tokens		
Voice call	All users	
Email OTP		
Certificate-based authentication		

Ilustración 10 - Métodos de autenticación soportados por Entra ID (Elaboración propia)

4.4.2 Privileged Identity Management (PIM)

PIM es un servicio de Entra ID accesible a través de licencias P1 o P2, cuya función principal es mitigar los riesgos asociados con los permisos de acceso excesivos, innecesarios o mal utilizados.

Una de las características principales de PIM es su capacidad para proporcionar activación de roles basada en el tiempo y en aprobaciones. Con esto se consigue que los usuarios solo puedan activar roles privilegiados cuando sea necesario para realizar tareas específicas, y este acceso está sujeto a la aprobación de un administrador. Esta funcionalidad no solo reduce el riesgo de abuso de privilegios, sino que también limita la exposición a posibles amenazas al restringir el acceso privilegiado a momentos y situaciones críticas.

También proporciona capacidades avanzadas de auditoría y supervisión que permiten a las organizaciones realizar un seguimiento detallado de la activación de roles. Los administradores tienen la capacidad de identificar quién activó un rol, cuándo se llevó a cabo la activación y qué acciones se realizaron mientras se tenía acceso a esos recursos. Estos registros detallados se presentan a través de informes y análisis, que permiten detectar anomalías y mejorar continuamente las políticas de acceso.

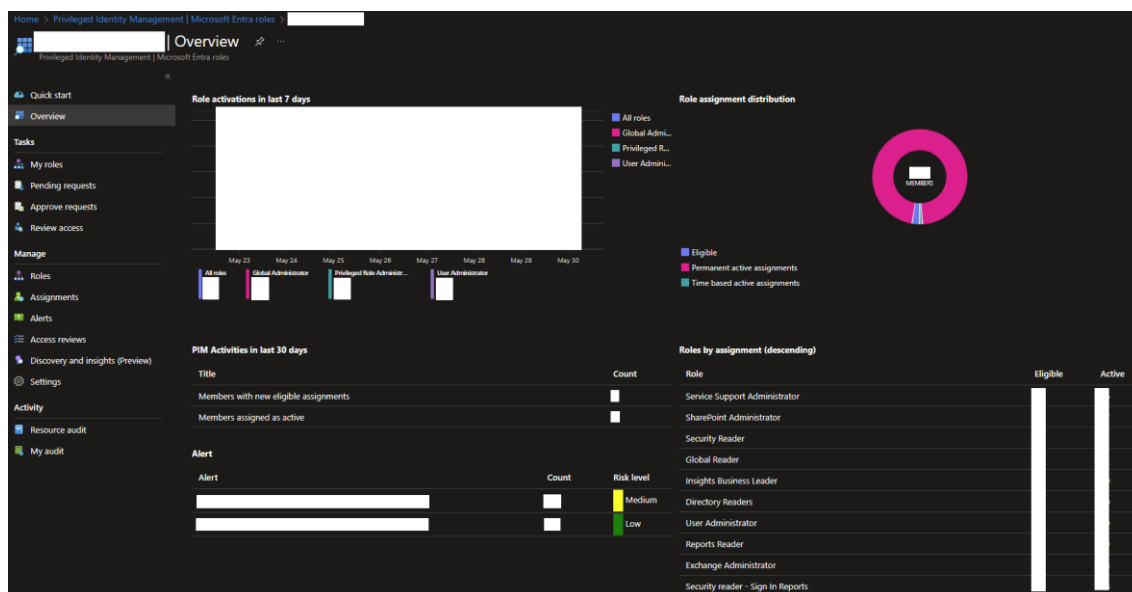


Ilustración 11 - Análisis de PIM (Elaboración propia)

4.4.3 Risk Detections

Son alertas generadas por el sistema cuando se detecta actividad sospechosa relacionada con las cuentas de usuario en el directorio. Las detecciones se basan en un análisis continuo de los registros de actividad de los usuarios, utilizando algoritmos avanzados establecidos por Microsoft para identificar patrones y comportamientos anómalos que podrían indicar una posible actividad maliciosa o compromiso de la cuenta.

Se diferencian dos tipos de riesgos:

- **Risky Sign-Ins:** Se clasifica un inicio de sesión como de riesgo debido a ciertos comportamientos sospechosos, como intentos de inicio de sesión desde direcciones IP anónimas, viajes imposibles a ubicaciones atípicas, inicios de sesión desde dispositivos infectados, inicios de sesión desde direcciones IP con actividad sospechosa e inicios de sesión desde ubicaciones desconocidas.
- **Risky user:** Se clasifica un usuario en riesgo cuando se considera que su cuenta está en riesgo de compromiso. Esto suele ocurrir cuando el mismo usuario tiene varios inicios de sesión clasificados de riesgo o cuando se detectan uno o más riesgos en la cuenta del usuario, como por ejemplo credenciales filtradas.

Tras detectar un riesgo, este es clasificado como low, mid o high según su gravedad y los administradores deben investigarlo utilizando los informes correspondientes y tomar las medidas adecuadas. Estas acciones pueden variar desde descartar el riesgo si se determina que es un falso positivo, solicitar un restablecimiento de contraseña para reforzar la seguridad de la cuenta, hasta incluso llegar a bloquear temporalmente la cuenta del usuario si se considera necesario para proteger los recursos de la organización.

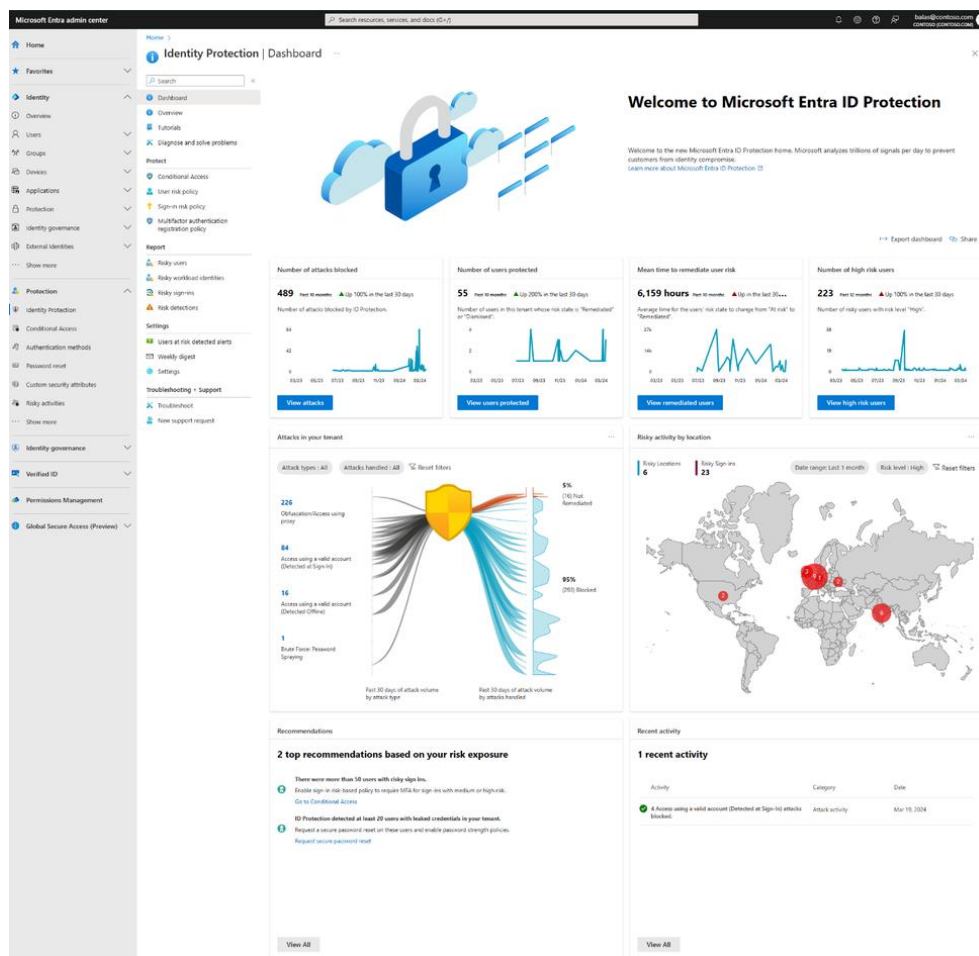


Ilustración 12 - Análisis de los riesgos (por AdminDroid Blog)

4.4.4 Self Service Password Reset (SSPR)

Es una herramienta diseñada para fomentar la autonomía de los usuarios al proporcionarles un proceso guiado y seguro para restablecer sus propias contraseñas utilizando los métodos de autenticación permitidos.

Con esto se consigue reducir la carga de trabajo del equipo de soporte técnico, ya que no será necesario que los técnicos dediquen tiempo a restablecer las contraseñas de los usuarios que lo requieran, como aquellos que han sido clasificados en riesgo. También minimiza el tiempo de inactividad de los usuarios, que ahora podrán resolver sus incidencias rápidamente y retomar sus actividades laborales. Además, el SSPR se puede configurar para adherirse a las políticas de seguridad de la empresa, garantizando que todas las nuevas contraseñas cumplan con los estándares de complejidad y longitud requeridos.

4.4.5 Identity Secure Score

Esta herramienta consiste en una medida numérica que evalúa el nivel de seguridad de un entorno, indica en un porcentaje en qué medida se cumplen las recomendaciones de seguridad de Microsoft. Estas recomendaciones de seguridad se basan en las mejores prácticas y cada una tiene una puntuación máxima, y cuanto más se acerque la configuración del entorno a estas directrices, mayor será la puntuación obtenida.

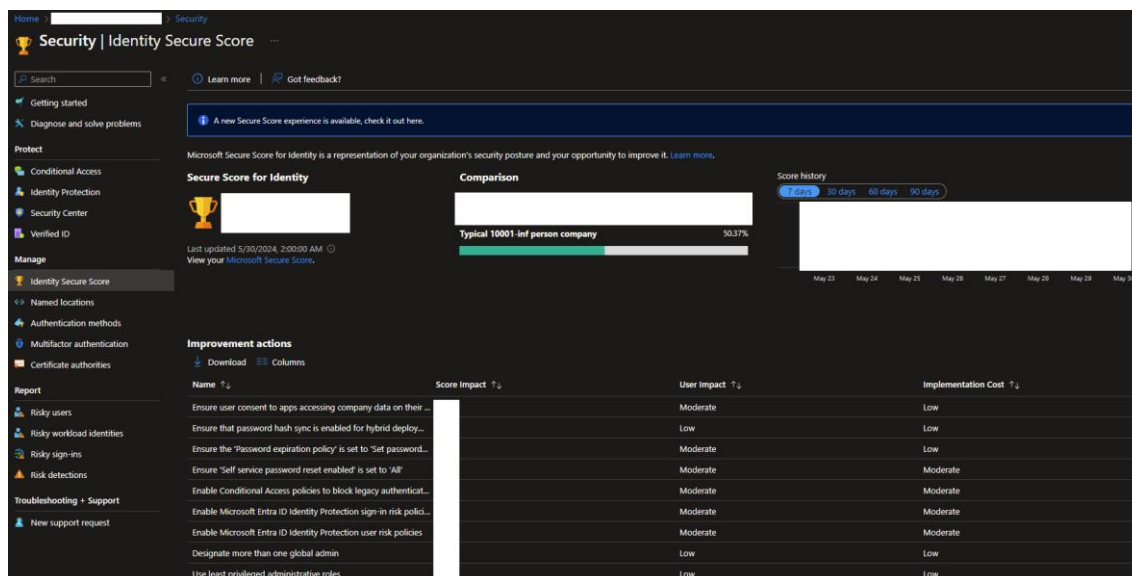


Ilustración 13 - Identity Secure Score (Elaboración propia)

Tal y como se observa en la ilustración 13, con el objetivo de mejorar la seguridad del entorno, Entra ID proporciona en una tabla detallada las recomendaciones de seguridad y el impacto que tendrían si se implementasen.

4.4.6 Herramientas de análisis de logs

Entra ID proporciona varios registros de las acciones que ocurren en el entorno los cuales son de gran utilidad para los administradores pues les permiten detectar aquellas actividades sospechosas de ser maliciosas. Entre los registros de logs destacan dos particularmente:

- **Sign-in logs:** Es un registro de los intentos de inicio de sesión que se producen en el entorno, incluyendo tanto los accesos realizados manualmente por los usuarios como los no interactivos que se realizan en segundo plano.
- **Audit logs:** Es un registro de los cambios realizados en un grupo, usuario, aplicación o incluso políticas de seguridad. Permite a los administradores identificar de forma

proactiva las posibles amenazas, problemas de seguridad y realizar un seguimiento de los cambios para evitar cambios no autorizados.

Date	Request ID	User	Application	Status	IP address	Location	Conditional Access	Authentication req...
5/30/2024, 3:44:59 PM				Success		Madrid, Madrid, ES	Not Applied	Single-factor authenticat...
5/30/2024, 3:44:58 PM				Success		Paris, Paris, FR	Success	Single-factor authenticat...
5/30/2024, 3:44:57 PM				Interrupted			Not Applied	Single-factor authenticat...
5/30/2024, 3:44:55 PM				Success		Barcelona, Barcelona, ES	Success	Single-factor authenticat...
5/30/2024, 3:44:55 PM				Success		Marseille, Bouches-Du-R...	Success	Single-factor authenticat...
5/30/2024, 3:44:54 PM				Success		Marseille, Bouches-Du-R...	Success	Single-factor authenticat...
5/30/2024, 3:44:54 PM				Success		Marseille, Bouches-Du-R...	Success	Single-factor authenticat...
5/30/2024, 3:44:53 PM				Success		Madrid, Madrid, ES	Success	Multifactor authenticat...
5/30/2024, 3:44:50 PM				Success		Marseille, Bouches-Du-R...	Success	Single-factor authenticat...
5/30/2024, 3:44:50 PM				Interrupted		Vilagarcía De Arousa, Pon...	Failure	Multifactor authenticat...

Ilustración 14 - Sign-in logs (Elaboración propia)

En la ilustración 14 se muestra cómo se ven los Sign-in logs del entorno, muy similar a como se muestran los Audit logs. Para ambos registros, los administradores pueden aplicar filtros sobre las columnas que aparecen en la imagen para simplemente analizar los logs de determinada aplicación o en un intervalo temporal determinado...

4.4.7 Conditional Access

Una herramienta clave para proteger los accesos a los recursos del entorno son las políticas de Conditional Access. Estas políticas se basan en sentencias if-then, en las que si se cumple con ciertas condiciones, se permitirá o denegará el acceso, de forma que se consigue controlar y supervisar quien, desde donde y como accede a los recursos de la organización.

Con esta herramienta se sigue el enfoque de seguridad “Zero Trust”, en el cual no se confía implícitamente en ningún usuario o dispositivo, independientemente de si está dentro o fuera de la red corporativa. Cada solicitud de acceso debe ser verificada y cumplir al menos una condición de autenticación en cada paso, garantizando así una protección continua y adaptativa contra posibles amenazas.

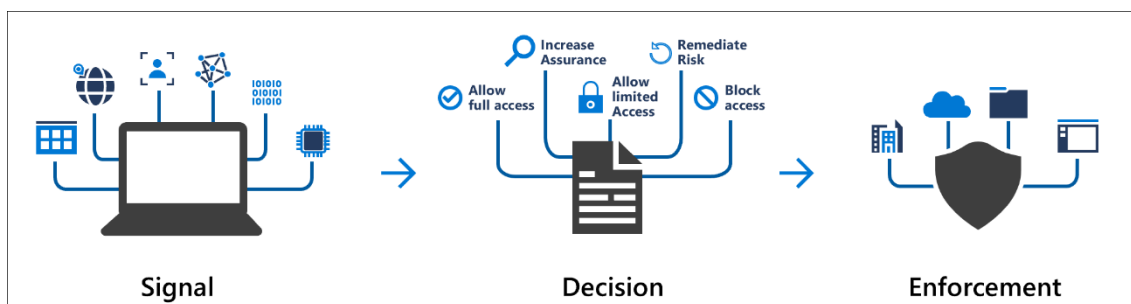


Ilustración 15 - Funcionamiento de Conditional Access (por Microsoft Learn) [4]

Tal y como se observa en la ilustración 15, Entra ID evalúa múltiples factores para tomar una decisión sobre el acceso a los recursos:

- **Contexto del usuario:** Se consideran la pertenencia a grupos, y si los usuarios son internos o invitados externos.
- **Dispositivo:** Se pueden incluir o excluir de la aplicación de las políticas en base a si el dispositivo es de confianza, hibridado, registrado...
- **Ubicación:** Se analiza si la solicitud proviene de una named location, y el administrador puede denegar o permitir el acceso según la ubicación sea o no de confianza.
- **Aplicaciones y recursos:** Las políticas también se pueden configurar para aplicar o no dependiendo de las aplicaciones y recursos a los que se intenta acceder.

- **Riesgo:** Permite aplicar la política a los usuarios en base a su nivel de riesgo, de forma que se pueden crear políticas que directamente bloqueen el acceso a las aplicaciones a los usuarios que se encuentran en riesgo.

Home > Security > Conditional Access > Conditional Access | Policies

Conditional Access policy

Delete View policy information

Control access based on Conditional Access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name *

Assignments

Users [Specific users included and specific users excluded](#)

Target resources [All cloud apps included and 19 apps excluded](#)

Network [NEW](#) [Any network or location and 2 excluded](#)

Conditions [2 conditions selected](#)

Access controls

Grant [Block access](#)

Session [0 controls selected](#)

Control access based on signals from conditions like risk, device platform, location, client apps, or device state. [Learn more](#)

User risk [User risk level is the likelihood that the user account is compromised.](#)
[Not configured](#)

Sign-in risk [Sign-in risk level is the likelihood that the sign-in session is compromised.](#)
[Not configured](#)

Insider risk (Preview) [Insider risk assesses the user's risky data-related activity in Microsoft Purview Insider Risk Management.](#)
[Not configured](#)

Device platforms [Not configured](#)

Locations [Any network or location and 2 excluded](#)

Client apps [Not configured](#)

Filter for devices [Exclude filtered devices](#)

Ilustración 16 - Condiciones de aplicación de una política de Conditional Access (Elaboración propia)

Con toda la información que se evalúa a la hora de acceder a un recurso se determina que políticas deben de aplicarse para ese acceso. Las políticas pueden tomar acciones para bloquear el acceso al recurso, permitirlo directamente o permitirlo pero requiriendo alguna acción extra como un MFA específico.

Home > Security > Conditional Access > Conditional Access | Policies > Grant

Conditional Access policy

Delete View policy information

Control access based on Conditional Access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name *

Assignments

Users [Specific users included and specific users excluded](#)

Target resources [All cloud apps included and 19 apps excluded](#)

Network [NEW](#) [Any network or location and 2 excluded](#)

Conditions [2 conditions selected](#)

Access controls

Grant [Block access](#)

Session [0 controls selected](#)

Control access enforcement to block or grant access. [Learn more](#)

☒ Block access

☐ Grant access

☐ Require multifactor authentication

☐ Require authentication strength

☐ Require device to be marked as compliant

☐ Require Microsoft Entra hybrid joined device

☐ Require approved client app [See list of approved client apps](#)

☐ Require app protection policy [See list of policy protected client apps](#)

☐ Require password change

☐ Código de Conducta Telemático

For multiple controls

☐ Require all the selected controls

☒ Require one of the selected controls

Ilustración 17 - Acciones que puede realizar una política de Conditional Access (Elaboración propia)



Para sacar el máximo beneficio de esta herramientas es muy importante crear en el entorno un conjunto de políticas que cubran todos los casos posibles de acceso para así evitar dejar puntos débiles que aprovechen los atacantes. Las políticas deben de complementarse entre sí y los administradores analizar detenidamente que todo aquello que una regla permite sea valorado por al menos otra regla. Siguiendo estas recomendaciones es cuando se consigue un control completo de las entradas a un sistema, cumpliendo con los requisitos de seguridad y garantizando la protección de los recursos de una empresa.

Capítulo 5. Flujos OAuth 2.0

Tras haber analizado detalladamente cómo funciona y que herramientas incorpora Entra ID es momento de explorar los flujos OAuth 2.0 y entender su utilidad y funcionamiento. Para ello es esencial entender dos conceptos clave:

- **Autenticación:** es el proceso de confirmar que alguien es quien dice ser.
- **Autorización:** es el proceso por el cual se decide si un usuario tiene acceso a ciertos datos

5.1 OAuth 2.0 y OpenID Connect

OAuth 2.0 es un protocolo que se centra en la autorización de aplicaciones para acceder a recursos en nombre de un usuario. El protocolo se encarga de que el acceso sea con un alcance definido ya que permite detallar los permisos de acceso que se le conceden a la app.

OpenID Connect es un protocolo de autenticación construido sobre el protocolo OAuth 2.0, que es comúnmente utilizado para la autorización en aplicaciones web y móviles. El propósito principal de este protocolo es agregar una capa adicional sobre OAuth 2.0 y así entre los dos protocolos manejar la autenticación y la autorización logrando un control total sobre el acceso.

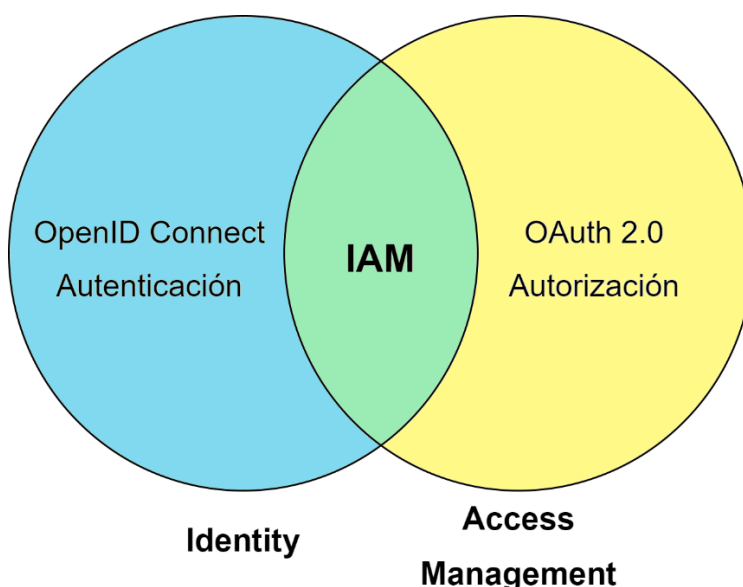


Ilustración 18 - Uso conjunto de OpenID Connect y OAuth 2.0 (Elaboración propia)

Los flujos OAuth que se van a analizar y que más tarde se emplearán en el caso de uso, no solo se encargan de la autorización sino también de la autenticación pues solo de esta manera es como se logra proteger tanto la identidad como los accesos.

5.2 Funcionamiento de un flujo OAuth 2.0

Los flujos OAuth 2.0 funcionan mediante la emisión de tokens de acceso, que son cadenas de texto ilegible con información de la duración del token, los permisos que otorga y su alcance entre otros. Los tokens actúan como llaves digitales que permiten el acceso a unos recursos protegidos sin exponer continuamente las credenciales de los usuarios finales.

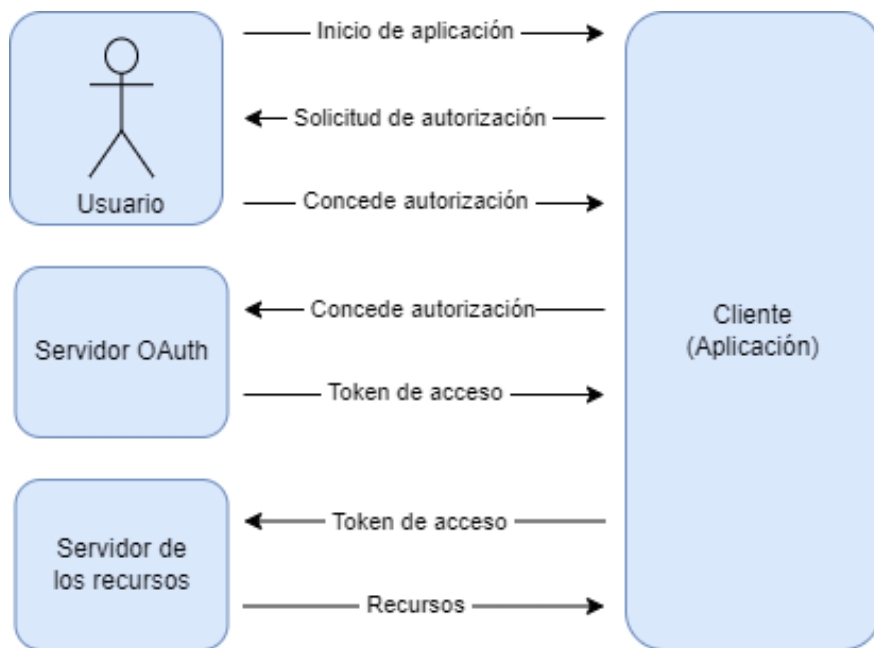


Ilustración 19 - Funcionamiento general de un flujo OAuth 2.0 (Elaboración propia)

La ilustración 19 presenta una representación a rasgos generales de los pasos que se siguen en un flujo. El usuario es quien accede a la aplicación y es reconducido por la aplicación al login, donde el usuario debe realizar la autenticación con sus credenciales. Tras completarse la autenticación la aplicación cliente solicita autorización para acceder a los recursos y una vez concedida la aplicación solicita el token de acceso al servidor OAuth, que es un IdP como Entra ID, quien se encarga de emitir el token. Ya con el token en su poder el cliente puede solicitar acceso a los recursos utilizando el token como llave las veces que sea necesario antes de que caduque. Para el caso de uso que se va a estudiar el servidor OAuth será Entra ID.

5.2.1 Refresh token

Como se ha comentado, cuando el usuario se autentica y autoriza, recibe un access token que podrá usar como llave para acceder a los recursos protegidos. Sin embargo, por motivos de seguridad, el access token tiene una duración limitada y caduca pasado ese periodo, por lo que se requiere repetir todo el proceso para obtener uno nuevo. El refresh token sirve precisamente para evitar esto, en vez de usar este token para conseguir acceso a los recursos, se emplea para renovar el access token cuando ha caducado, sin necesidad de volver a introducir las credenciales del usuario. Este proceso permite a la aplicación continuar accediendo a los recursos de manera ininterrumpida, ya que resulta totalmente transparente.

Como con el refresh token permite obtener un access token, es necesario tomar las medidas de seguridad correspondientes para proteger el entorno. Unas buenas prácticas de seguridad son:

- Almacenar ambos token en el backend de la aplicación ya que así se evita su exposición.
- Ambos token deben de estar encriptados tanto mientras se almacenan en el backend como mientras se transmiten.
- Utilizar la rotación de tokens que consiste en que cada vez que se utiliza el refresh token para obtener un nuevo access token, se emite otro refresh token y el anterior deja de ser valido. De esta forma se limita el impacto en caso de que un refresh token sea comprometido
- Se debe de permitir la revocación de ambos token, de esta forma se puede detener un ataque en caso de que alguno sea comprometido.


```
{
  "access_token": "AYjcyMzY3ZDhiNmJkNTY",
  "refresh_token": "RjY2NjM5NzA2OWJjuE7c",
  "token_type": "bearer",
  "expires": 3600
}
```

Ilustración 20 - Ejemplo de tokens (por Okta) [5]

5.2.2 Solicitud del token

Tras realizar autenticación y autorización, es cuando el cliente solicita el access token al servidor OAuth. Esta solicitud se da a través de una petición GET HTTP que lanza el cliente al servidor, la cual contiene una serie de parámetros que contienen toda la información sobre el alcance del token, su duración requerida, información del solicitante...

A continuación se van a comentar los más relevantes y comunes, ya que no siempre son necesarios los mismos parámetros:

- **response_type**: Este parámetro no indica si se necesita el access token, su función es indicarle al servidor OAuth la respuesta que se espera a la petición. Tal y como se estudiará a continuación, dependiendo del flujo las respuestas que debe dar el servidor OAuth son distintas, por lo que el valor de este parámetro será "code" cuando se espera un código como respuesta y será "token" cuando se espera el access token como respuesta.
- **scope**: Se trata de una lista de los permisos, que pueden ser tanto de lectura como de escritura, que requiere el cliente, y por lo tanto le debe conceder el access token.
- **client_id**: Es un identificador único asignado al cliente por el servidor OAuth, en el momento en que se integró la aplicación en el IdP. Su función es la de identificar al cliente que está iniciando el flujo.
- **redirect_uri**: Este parámetro indica la URL del cliente a la que el servidor OAuth debe redirigir al usuario tras completar el flujo. Como medida de seguridad, el servidor OAuth almacena en el momento de la integración de la aplicación esta URL, de forma que cuando llega la petición compara el valor de este parámetro con el que tiene almacenado para así evitar posibles ataques de redirección.
- **state**: Se trata de un valor único y aleatorio que el cliente utiliza como medida de seguridad. El cliente genera este valor aleatorio y lo incluye en su petición inicial, si en la respuesta el servidor OAuth responde con un valor distinto en este parámetro, el cliente la descarta al considerarla ilegítima.
- **client_secret**: Este parámetro aparece solo en los flujos que emplean client secret y sirve para autenticar al cliente ante el servidor OAuth. Este valor es un secreto compartido entre el cliente y el servidor y se utiliza para proteger la comunicación.

```
https://authorization-server.com/auth?response_type=code
&client_id=29352735982374239857
&redirect_uri=https://example-app.com/callback
&scope=create+delete
&state=xcoivjuywkdkhvususye3kch
```

Ilustración 21 - Petición HTTP (por Okta) [6]

5.2.3 Extensión flows

OAuth es un estándar abierto y flexible que ofrece a los desarrolladores la capacidad de personalizar los flujos de autorización según las necesidades específicas de sus aplicaciones. Esto significa que los desarrolladores pueden crear nuevos flujos modificando los flujos estándar y que se conocen como extension flows. Esto puede incluir la integración con sistemas de autenticación existentes o la implementación de lógicas de autorización personalizadas.

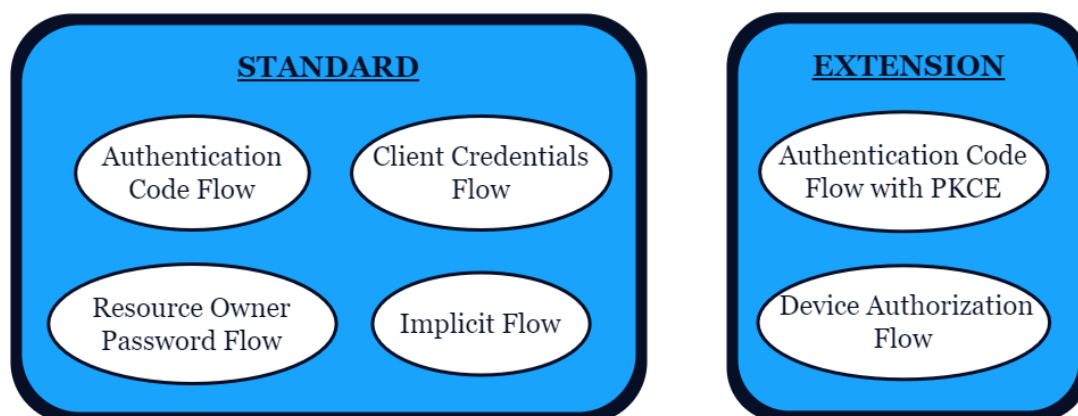


Ilustración 22 - Flujos standard y extension (Elaboración propia)

En la ilustración 22 se muestran los flujos que se van a estudiar a continuación. Basándose en el estándar RFC 6749, que define las especificaciones de OAuth 2.0, se establecen cuatro flujos estándar, a los que se van a añadir dos extension flows muy conocidos en la comunidad. [7]

5.3 Tipos de flujo

Una vez estudiados diferentes aspectos de los flujos, se van a analizar los flujos OAuth 2.0 más conocidos. A pesar de que el objetivo de todos los flujos es que la aplicación consiga el access token para acceder a los recursos, cada flujo representa una forma diferente de hacerlo.

Los desarrolladores de la aplicación son los encargados de seleccionar el flujo que más se ajuste a su aplicación, de forma que se garantice un proceso de obtención del token seguro y eficiente. Siempre se recomienda el uso de los flujos más seguros para aquellas aplicaciones que acceden a los datos más sensibles de la organización.

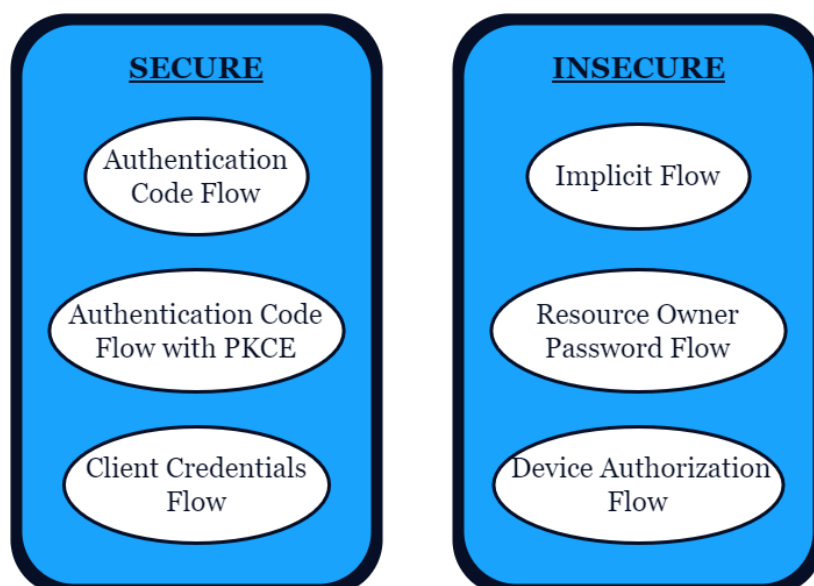


Ilustración 23 - Flujos clasificados según su seguridad (Elaboración propia)

5.3.1 Authorization Code Flow

Este flujo es considerado de los más seguros, pues hace uso de un código de seguridad llamado code y de un client secret, los cuales el servidor OAuth solicita para emitir el token de acceso, y que protegen contra ataques de interceptación. Además la aplicación emplea su backend como almacén seguro para el token y otros datos sensibles, evitando así su exposición en el navegador. Todo esto combinado en todo momento con HTTPS para cifrar los datos durante la transmisión hacen que la captura del token durante el flujo sea prácticamente imposible para el atacante.

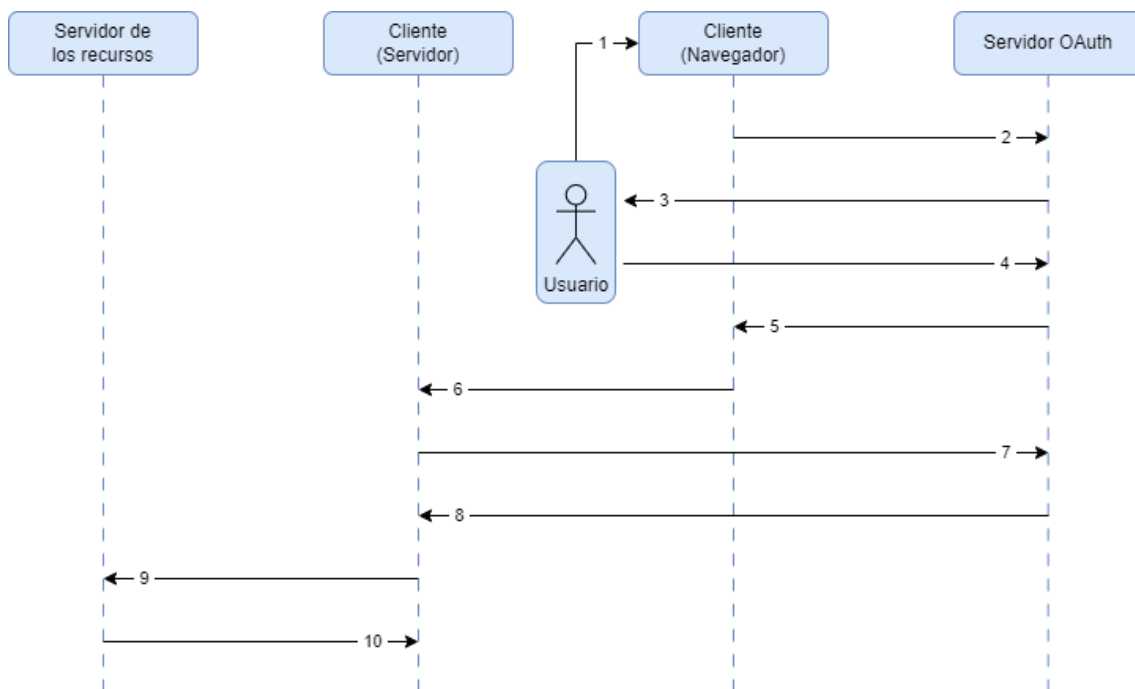


Ilustración 24 - Authorization Code Flow (Elaboración propia)

En la ilustración 24 se observa el funcionamiento detallado del flujo el cual se va a comentar paso por paso:

1. **Inicio de la Aplicación:** El usuario inicia la aplicación en su navegador.
2. **Solicitud de Autorización:** El cliente solicita autorización al servidor OAuth con una petición HTTP que contiene todos los parámetros analizados previamente.
3. **Redireccionamiento al login:** El usuario es redirigido a la página de login del servidor OAuth.
4. **Autenticación y autorización del Usuario:** El usuario inicia sesión en el servidor OAuth y otorga su consentimiento para que la aplicación acceda a los datos.
5. **Emisión del code:** El servidor redirige al usuario de vuelta a la aplicación en el redirect_uri especificado, adjuntando el code. El code es transmitido a través del navegador pero no representa un riesgo de seguridad, ya que sólo puede ser utilizado una vez también se necesita el client_secret para obtener el token de acceso.
6. **Envío del code al backend:** La aplicación envía este código de autorización a su backend mediante un canal seguro.
7. **Intercambio por access token:** El backend del cliente envía una solicitud al servidor OAuth para obtener el access token, donde incluye el client_secret y el code.
8. **Emisión del access token:** El servidor OAuth verifica el client_secret y el code y si todo es correcto, emite el access token (y opcionalmente un refresh token).
9. **Acceso a Recursos:** La aplicación usa el access token para acceder a los recursos.
10. **Validación del token:** El servidor verifica la autenticidad del access token y la aplicación accede al recurso protegido.

5.3.2 Authorization Code Flow with PKCE

Este extended flow consigue mejorar todavía más la seguridad de uno de los flujos más seguros. Al incorporar un secreto dinámico, conocido como PKCE (Proof Key for Code Exchange), este flujo protege contra ataques de interceptación de código, lo que lo convierte en una opción ideal para aplicaciones que no pueden almacenar un client secret de forma segura, como aplicaciones nativas y SPA (Single-Page Apps). Un ejemplo práctico de su utilidad es el proceso de inicio de sesión en una aplicación móvil que utiliza proveedores de identidad como Google o Facebook.

Este nuevo secreto funciona de manera que la aplicación genera un code verifier que es un código criptográficamente aleatorio y a partir de este genera un code challenge. Inicialmente la aplicación compartirá el challenge con el servidor OAuth y le compartirá el verifier solo al final del flujo cuando solicita el token. El servidor OAuth resolverá el challenge obteniendo el verifier y lo comparará con el recibido, solo emitirá el token si ambos coinciden.

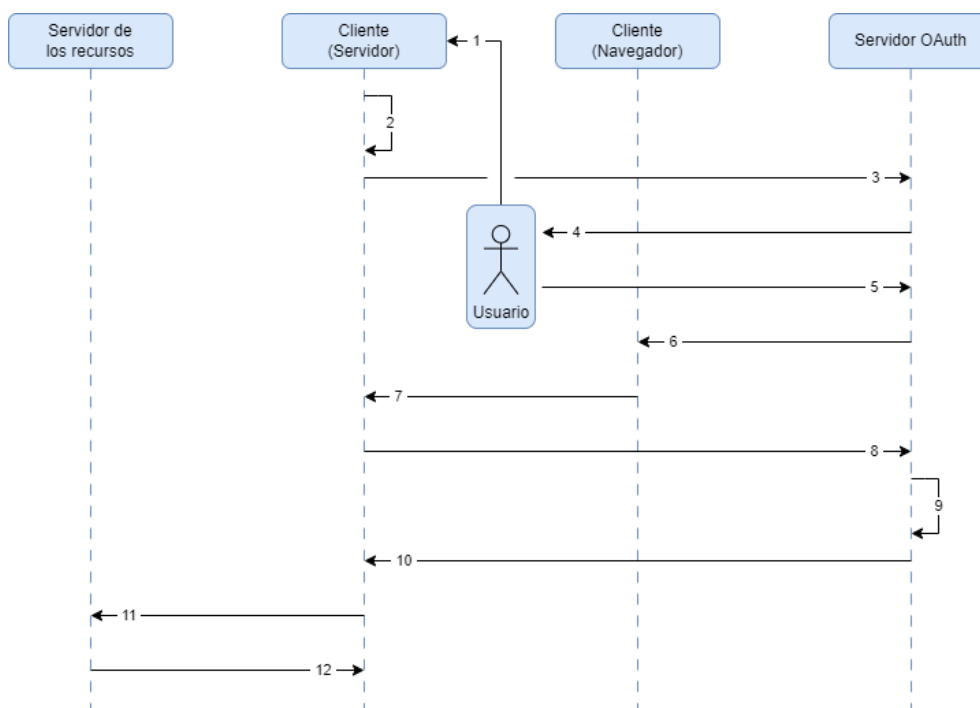


Ilustración 25 - Authorization Code Flow with PKCE (Elaboración propia)

En la ilustración 25 se observa el funcionamiento detallado del flujo el cual se va a comentar paso por paso:

1. **Inicio de la Aplicación:** El usuario inicia la aplicación en su navegador.
2. **Creación del PKCE:** La aplicación crea un code verifier y a partir de este genera el code challenge.
3. **Solicitud de Autorización:** El cliente solicita autorización al servidor OAuth con una petición HTTP que contiene todos los parámetros analizados previamente y ahora también con el challenge.
4. **Redireccionamiento al login:** El usuario es redirigido a la página de login del servidor OAuth.
5. **Autenticación y autorización del Usuario:** El usuario inicia sesión en el servidor OAuth y otorga su consentimiento para que la aplicación acceda a los datos.
6. **Emisión del code:** El servidor OAuth almacena el challenge y redirige al usuario de vuelta a la aplicación adjuntando en la URL el code.
7. **Envío del code al backend:** La aplicación envía este código de autorización a su backend mediante un canal seguro.

8. **Intercambio por access token:** El backend del cliente envía una solicitud al servidor OAuth para obtener el access token, donde incluye el code, el code verifier y el client secret.
9. **Verificación:** El servidor OAuth resuelve el challenge y compara el resultado obtenido con el verifier, también compara el code y el client secret con los que tiene almacenados
10. **Emisión del token de acceso:** Si todo es correcto se emite el token de acceso.
11. **Acceso a Recursos:** La aplicación usa el access token para acceder a los recursos.
12. **Validación del token:** El servidor verifica la autenticidad del access token y la aplicación accede al recurso protegido.

5.3.3 Client Credentials Flow

Este flujo permite que una aplicación se autentique y autorice utilizando sus propias credenciales al llamar a un recurso web, sin ser necesaria la intervención de un usuario. Esta característica lo hace especialmente útil para aplicaciones Machine-to-Machine (M2M), donde las interacciones entre servidores se realizan en segundo plano, sin la participación directa de un usuario. Un ejemplo claro de uso de este flujo es una aplicación de backend que necesita permisos para acceder a una API de terceros. Por el momento este flujo es considerado seguro, siempre y cuando se sigan las correctas prácticas de seguridad, como almacenar el client secret de segura en el backend del cliente.

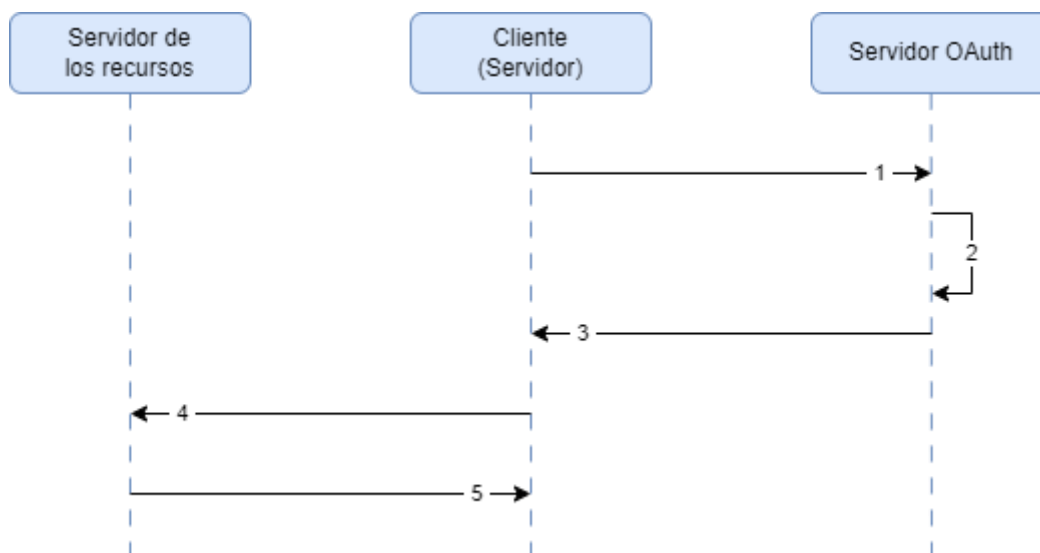


Ilustración 26 - Client Credentials Flow (Elaboración propia)

En la ilustración 26 se observa el funcionamiento detallado del flujo el cual se va a comentar paso por paso:

1. **Solicitud de autorización:** El cliente envía sus credenciales al servidor OAuth.
2. **Validación de credenciales:** El servidor OAuth comprueba que son correctas las credenciales del cliente.
3. **Envío del token:** Si todo es correcto el servidor OAuth emite el access token y lo envía directamente al backend del cliente.
4. **Acceso a Recursos:** La aplicación usa el access token para acceder a los recursos.
5. **Validación del token:** El servidor verifica la autenticidad del access token y la aplicación accede al recurso protegido. [8]

5.3.4 Implicit Flow

Este flujo se diseñó originalmente para aplicaciones basadas en navegadores y aplicaciones nativas, ya que facilitaba la obtención del access token directamente, sin necesidad de un

intercambio adicional de códigos. A diferencia del Authorization Code Flow, donde el cliente tenía que solicitar el access token presentando el code y el client secret, en este flujo el cliente puede obtener el access token sin ser necesario ningún code ni client secret, solo realizando la autenticación con las credenciales.

A la falta de seguridad debido a la ausencia de code y client secret, hay que añadir que el access token se transfiere del servidor OAuth al cliente a través de la URL del navegador, lo que presenta un riesgo frente a ataques de interceptación. Sin embargo, este diseño excesivamente simple presenta varias vulnerabilidades, lo que ha llevado a que su uso ya no sea recomendado por organizaciones como OAuth y Microsoft, cuando se busca seguir las mejores prácticas de seguridad.

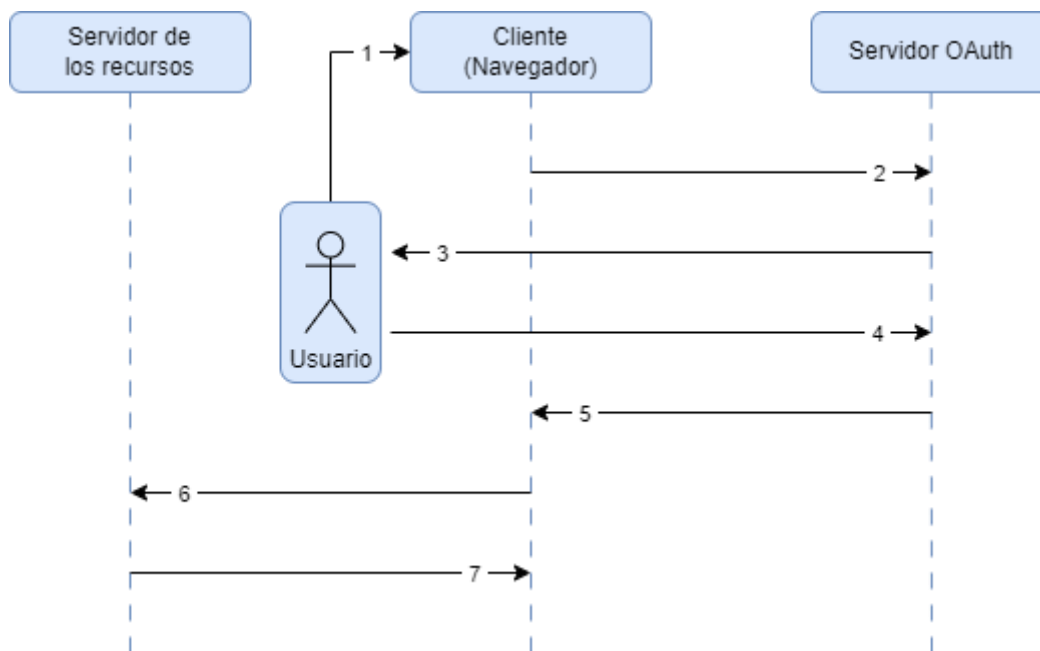


Ilustración 27 - Implicit Flow (Elaboración propia)

En la ilustración 27 se observa el funcionamiento detallado del flujo el cual se va a comentar paso por paso:

1. **Inicio de la Aplicación:** El usuario inicia la aplicación en su navegador.
2. **Solicitud de Autorización:** El cliente solicita autorización al servidor OAuth con una petición HTTP que contiene todos los parámetros analizados previamente.
3. **Redireccionamiento al login:** El usuario es redirigido a la página de login del servidor OAuth.
4. **Autenticación y autorización del Usuario:** El usuario inicia sesión en el servidor OAuth y otorga su consentimiento para que la aplicación acceda a los datos.
5. **Emisión del access token:** El servidor OAuth emite el access token el cual compartirá con el cliente incluyéndolo en la URL que envía al navegador.
6. **Acceso a Recursos:** La aplicación usa el access token para acceder a los recursos.
7. **Validación del token:** El servidor verifica la autenticidad del access token y la aplicación accede al recurso protegido. [9]

5.3.5 Resource Owner Password Flow

En este flujo, es el cliente el que solicita directamente las credenciales del usuario, en lugar de que la introducción de credenciales la gestione el servidor OAuth. La aplicación puede pedir al usuario que ingrese sus credenciales a través de un formulario interactivo y luego enviarlas al backend para su uso futuro antes de intercambiarlas por un token de acceso con el servidor OAuth.

Por motivos de seguridad no se recomienda el flujo y se reserva para aplicaciones de alta confianza que aseguren en todo momento la protección de las credenciales del usuario.

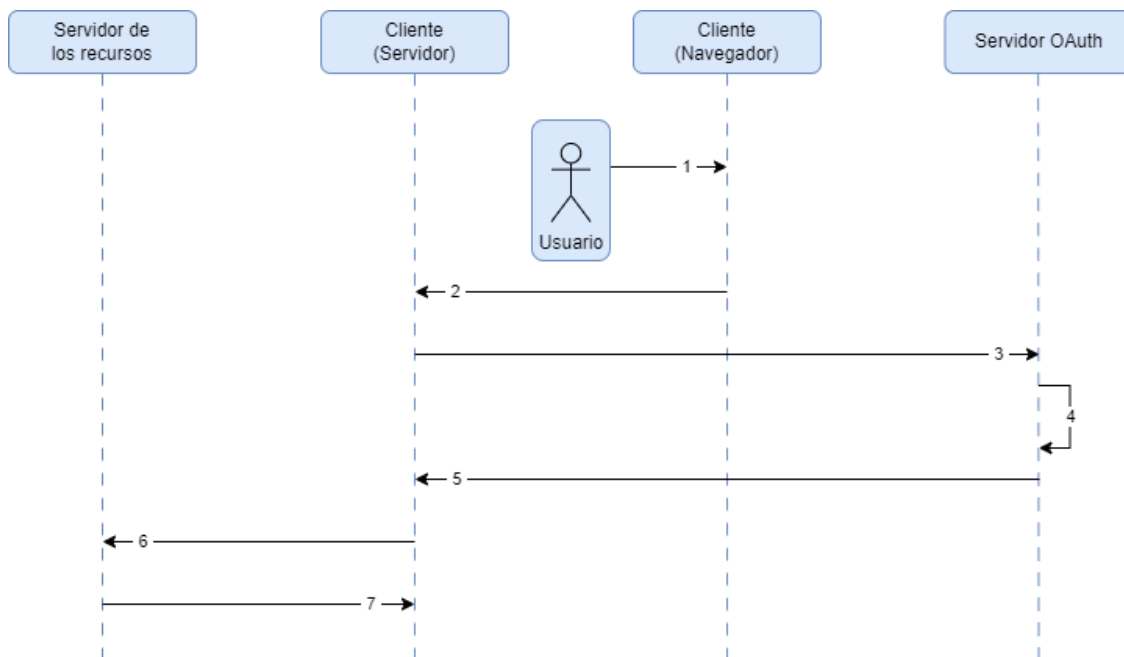


Ilustración 28 - Resource Owner Password Flow (Elaboración propia)

En la ilustración 28 se observa el funcionamiento detallado del flujo el cual se va a comentar paso por paso:

1. **Introducción de credenciales:** El usuario inicia la aplicación en su navegador e introduce sus credenciales en un formulario de login que le muestra el navegador.
2. **Almacenamiento de credenciales:** El navegador envía las credenciales al backend de la aplicación a través de una solicitud HTTP
3. **Envío de credenciales:** El backend recibe las credenciales y las valida. Luego, envía las credenciales al servidor OAuth para la autenticación y autorización.
4. **Validación:** El servidor OAuth valida que las credenciales son correctas.
5. **Emisión del access token:** El servidor genera el token y lo transmite al cliente para que lo almacene en su backend.
6. **Acceso a Recursos:** La aplicación usa el access token para acceder a los recursos.
7. **Validación del token:** El servidor verifica la autenticidad del access token y la aplicación accede al recurso protegido. [10]

5.3.6 Device Authorization Flow

Este flujo está diseñado para dispositivos que carecen de un navegador o tienen restricciones de entrada, como las Smart TVs o los dispositivos IoT. Funciona de manera que el dispositivo sin navegador muestra un código al usuario, quien luego debe visitar una URL en otro dispositivo, como un teléfono móvil o una PC, para completar el flujo y poder obtener el access token en el dispositivo principal

Sin embargo, este método conlleva riesgos de seguridad, incluida la posible exposición del código de autorización durante el proceso y la vulnerabilidad a ataques de intermediarios, como los ataques "Man-In-The-Middle" (MITM).

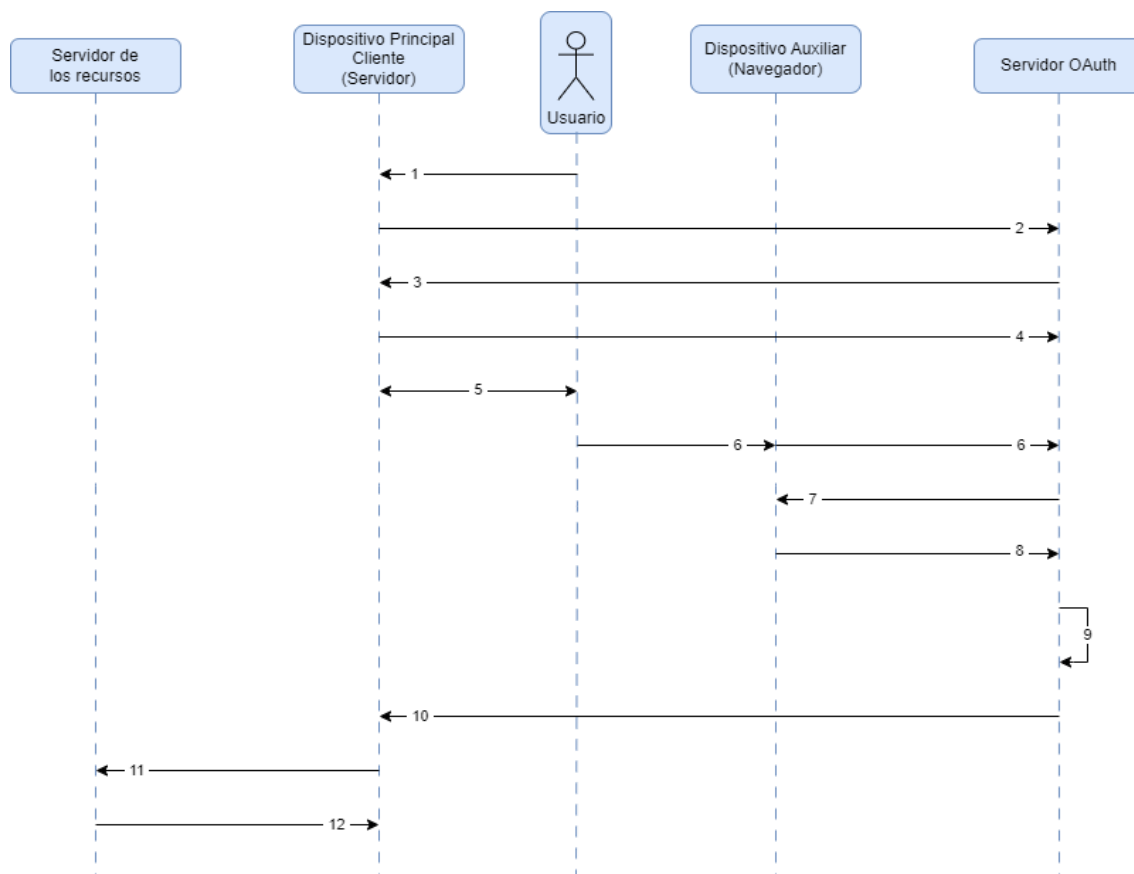


Ilustración 29 - Device Authorization Flow (Elaboración propia)

En la ilustración 29 se observa el funcionamiento detallado del flujo el cual se va a comentar paso por paso:

1. **Inicio de la Aplicación:** El usuario inicia la aplicación en el dispositivo principal.
2. **Solicitud de Autorización:** El cliente solicita autorización al servidor OAuth
3. **URL de verificación:** El Servidor OAuth responde con la URI de verificación y una serie de códigos como el device code, user code, sus tiempos de expiración y el intervalo de espera entre solicitudes de polling.
4. **Polling (Sondeo):** El cliente comienza a hacer solicitudes periódicas al servidor OAuth para obtener el access token que no cesarán hasta que se complete la autorización en el dispositivo secundario o que expire el temporizador de los codes.
5. **Acceso a la URL:** El usuario accede a la URL introduciéndola en el dispositivo auxiliar o bien escaneándola en caso de que venga en un QR.
6. **Verificación:** El usuario ingresa el user code que ha recibido en el dispositivo principal (Si ha accedido a la URL mediante QR se hace automáticamente).
7. **Redirección a login:** El Servidor OAuth redirige al usuario a la página de inicio de sesión y posteriormente a la página de consentimiento.
8. **Autenticación y consentimiento:** El usuario inicia sesión en el servidor OAuth y a continuación otorga su consentimiento para que la aplicación acceda a sus datos.
9. **Autorización al cliente:** Finaliza la función del dispositivo auxiliar al completar exitosamente el proceso.
10. **Emisión del access token:** El servidor genera el token y lo transmite al cliente para que lo almacene en su backend.
11. **Acceso a Recursos:** La aplicación usa el access token para acceder a los recursos.
12. **Validación del token:** El servidor verifica la autenticidad del access token y la aplicación accede al recurso protegido. [11]

Capítulo 6. Caso de uso

En este apartado se lleva a cabo la integración de una aplicación web con Entra ID para que actúe como su IdP. Este proceso permitirá examinar de manera detallada las acciones y configuraciones necesarias desde ambos lados, tanto desde la perspectiva de la aplicación como desde el entorno de Entra ID, para lograr un funcionamiento correcto y seguro.

La aplicación tiene un diseño y funcionamiento sencillo, puesto que el objetivo principal es mostrar cómo conectar la aplicación con Entra ID y lograr que el funcionamiento de la app no suponga un riesgo para los datos que accede. Funcionará de manera que el usuario accederá utilizando sus credenciales de Microsoft y se le pedirá que conceda permisos a la aplicación. Tras esto es redirigido a una página de bienvenida, desde la cual podrá utilizar la API Graph, una herramienta que permite acceder y manipular datos de Microsoft 365 y otros servicios. Al llamar a la API, en otra página se le mostrarán los datos de su perfil obtenidos a través de una consulta.

Inicialmente el flujo que empleará la aplicación será Implicit Flow debido a la simplicidad de su proceso. El objetivo es, una vez la aplicación sea funcional, examinar las vulnerabilidades que presenta y como un atacante puede capturar el access token.

Una vez mostrado el problema, se realizarán las acciones necesarias para lograr securizar el proceso, comenzando por cambiar el flujo por Authorization Code Flow y después se hará uso de algunas de las herramientas de Entra ID para bloquear los accesos no deseados.

6.1 Integración con Implicit Flow

Para el diseño e integración de la aplicación se va a comenzar realizando las acciones desde la parte de Entra ID y después se diseñará toda la parte del código. Lo siguiente será mostrar el funcionamiento de aplicación con la configuración actual para acabar enseñando las vulnerabilidades que hay con este diseño.

6.1.1 Creación de la aplicación en Entra ID

El primer paso de la integración es crear la aplicación en Entra ID, dado que la aplicación va a funcionar mediante autenticación y autorización por OAuth, se debe ir a la sección de App Registration y seleccionar la opción “New registration” donde se debe introducir el nombre para la aplicación y marcar la opción de “permitir el acceso solo a las cuentas de este tenant” para mayor seguridad.

[Home](#) > [App registrations](#) >

Register an application ...

* Name

The user-facing display name for this application (this can be changed later).

TFG - APP

Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (NTTdataAzureCyber only - Single tenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform

e.g. <https://example.com/auth>

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

Ilustración 30 - Creación de la App registration (Elaboración propia)

Una vez creada la aplicación, se deben realizar ciertas configuraciones desde el apartado “Authentication”. Aquí, se debe definir la Reply URL, que es la dirección a la que se enviará el token después de completar el flujo, y además hay que activar unas opciones para permitir que Entra ID funcione con Implicit Flow, ya que por defecto no lo permite, pues como se ha comentado anteriormente Microsoft ya no recomienda el uso de este flujo.

[Home](#) > [App registrations](#) > [TFG - APP](#)

TFG - APP | Authentication

Search

Got feedback?

- Overview
- Quickstart
- Integration assistant
- Manage
 - Branding & properties
 - Authentication
 - Certificates & secrets
 - Token configuration
 - API permissions
 - Expose an API
 - App roles
 - Owners
 - Roles and administrators
 - Manifest
- Support + Troubleshooting
 - Troubleshooting
 - New support request

Web

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URIs. [Learn more about Redirect URIs and their restrictions](#)

This app has implicit grant settings enabled. If you are using any of these URIs in a SPA with MSAL.js 2.0, you should migrate URIs.

<http://localhost:5000/getAToken>

[Add URI](#)

Front-channel logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

e.g. <https://example.com/logout>

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint. If the application has a single-page architecture (SPA) and doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. [Learn more about tokens](#)

Select the tokens you would like to be issued by the authorization endpoint:

- ☒ Access tokens (used for implicit flows)
- ☒ ID tokens (used for implicit and hybrid flows)

Ilustración 31 - Configuración de la app registration para funcionar con Implicit Flow (Elaboración propia)

Finalmente, se deben asignar a la aplicación los permisos que la aplicación solicitará al usuario una vez se haya autenticado. Para esta aplicación tan solo se requieren permisos de lectura, por lo que, siguiendo el principio de Least Privilege Access, se le otorgará únicamente el permiso “user.read”. Este principio busca otorgar los permisos mínimos necesarios para que el usuario pueda ejecutar correctamente su trabajo, minimizando así los riesgos.

Para asignar un permiso, se debe hacer desde la sección “API permissions”, y seleccionar “Add a permission”, donde se debe elegir la API requerida, que en este caso es Microsoft Graph. Para finalizar hay que establecer que estos permisos son Delegated, ya que la aplicación solo los necesita cuando el usuario inicia sesión, y buscar y marcar el permiso “user.read” para acabar de añadirlo.

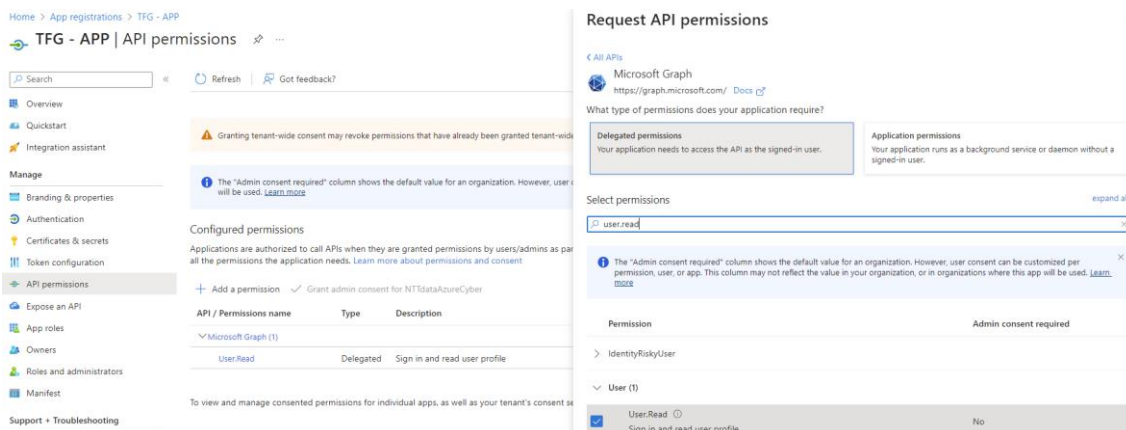


Ilustración 32 - Configuración de los permisos de la API (Elaboración propia)

6.1.2 Código

Lo primero que se debe hacer es crear un archivo “.env” donde se van a almacenar las variables de entorno, en este caso almacenaremos el “CLIENT_ID” de la aplicación que se obtiene desde Entra ID y una variable “AUTHORITY” formada por la URL <https://login.microsoftonline.com/> seguida del Tenant ID que también se obtiene desde Entra ID, ambos desde el apartado properties de la App registration.

1. CLIENT_ID=3dc9137e-a8cb-4263-b278-382fe7612a4b
2. AUTHORITY=https://login.microsoftonline.com/1e2fadca-913b-4243-91b7-e4f0cf339ab5

A continuación hay que crear un archivo “app_config.py” que es donde se van a configurar los parámetros necesarios para que la aplicación conecte con Entra ID. Primero se crearán de nuevos las variables “AUTHORITY” y “CLIENT_ID” que extraerán el valor de las variables de entorno definidas en “.env”, y también se definirán la ruta de redirección, el endpoint de la API Graph, los permisos requeridos y el tipo de sesión que se utilizará.

```

1. import os
2.
3. AUTHORITY = os.getenv("AUTHORITY")
4. CLIENT_ID = os.getenv("CLIENT_ID")
5.
6. # Ruta de redirección después del login
7. REDIRECT_PATH = "/getAToken"
8.
9. # Endpoint de la API
10. ENDPOINT = 'https://graph.microsoft.com/v1.0/me'
11.
12. # Permisos requeridos
13. SCOPE = ["User.Read"]
14.
15. # Establece almacenar las sesiones en el sistema de archivos
16. SESSION_TYPE = "filesystem"

```

Con estos dos archivos lo que se consigue es separar estos parámetros críticos del código de la aplicación, lo cual aporta una capa de seguridad, puesto que si un atacante consigue acceso al código no podrá ver estos valores.

Una vez creados los archivos de configuración, se procede a implementar el código principal en un archivo llamado “app.py”, donde primero se importan las siguientes librerías:

- **requests**: Utilizada para realizar solicitudes HTTP.
- **flask**: Importa las funcionalidades básicas de Flask para la creación de la aplicación web.
- **flask_session**: Utilizada para gestionar las sesiones en la aplicación Flask.

Además, se importa las configuraciones definidas en app_config y se define la versión de la aplicación como 1.0.0.

```
1. import requests
2.
3. from flask import Flask, redirect, render_template, request, session, url_for
4. from flask_session import Session
5.
6. import app_config
7. __version__ = "1.0.0"
```

Lo siguiente es crear la instancia de Flask y cargar la configuración de la aplicación desde el objeto “app_config”. Se realiza una comprobación para asegurarse de que la variable “REDIRECT_PATH” no esté configurada como la raíz ("/"), ya que esto podría causar problemas de seguridad. Finalmente, se inicializa la gestión de sesiones permitiendo que la aplicación mantenga el estado de la sesión del usuario a lo largo de su interacción.

```
8. app = Flask(__name__, static_url_path='', static_folder='static')
9. app.config.from_object(app_config)
10. assert app.config["REDIRECT_PATH"] != "/", "REDIRECT_PATH must not be /"
11. Session(app)
```

Como la app se ejecuta en localhost, se utiliza ProxyFix de middleware para corregir los encabezados de solicitud.

```
12. from werkzeug.middleware.proxy_fix import ProxyFix
13. app.wsgi_app = ProxyFix(app.wsgi_app, x_proto=1, x_host=1)
```

Ahora hay que definir las rutas para manejar las solicitudes, comenzando por /login. En la función login, se construye la URL para redirigir al usuario al login de Microsoft utilizando los parámetros de configuración de la aplicación, incluyendo el client_id, el tipo de respuesta, que al trabajar con Implicit Flow debe ser “token”, la URL de redirección, los permisos necesarios que vienen del scope, y un prompt para seleccionar la cuenta. Finalmente, la función redirige al usuario a esta URL de autorización.

```
14. @app.route("/login") # Define la ruta para manejar las solicitudes a /login
15. def login():
16.     auth_url = (
17.         f"{app.config['AUTHORITY']}/oauth2/v2.0/authorize"
18.         f"?client_id={app.config['CLIENT_ID']}"
19.         f"&response_type=token"
20.         f"&redirect_uri={url_for('auth_response', _external=True)}"
21.         f"&scope={' '.join(app_config.SCOPE)}"
22.         f"&prompt=select_account"
23.     )
24.     return redirect(auth_url)
```

También hay que definir la ruta para manejar la respuesta de autenticación. Para ello se comprobaba si se ha recibido el access token a través de un formulario POST. Si se ha recibido se almacena en la sesión y se redirige al usuario a la página principal (index). Si no se encuentra, se

redirige al usuario de nuevo a la página de login. Si la solicitud es de tipo GET, se renderiza una plantilla `auth_response.html`.

```
25. @app.route(app_config.REDIRECT_PATH, methods=["GET", "POST"])
26. def auth_response():
27.     if request.method == "POST":
28.         access_token = request.form.get("access_token")
29.         if access_token:
30.             session["access_token"] = access_token
31.             return redirect(url_for("index"))
32.         else:
33.             return redirect(url_for("login"))
34.     return render_template("auth_response.html")
```

Cuando llega una solicitud a la ruta principal, la aplicación ha de verificar primero si hay “CLIENT_ID” establecido y en caso de no haberlo carga una página de error. Luego comprueba si el usuario está logeado, buscando si en la sesión hay un access token y en caso de no haberlo se vuelve a la página de login. Si todo es correcto se renderiza la página principal indicando el nombre del usuario logeado.

```
39. @app.route("/")
40. def index():
41.     if not app.config["CLIENT_ID"]:
42.         return render_template('config_error.html')
43.     if "access_token" not in session:
44.         return redirect(url_for("login"))
45.     return render_template('index.html', user={"name": "User"}, version=__version__)
```

Para cuando se haga una solicitud a logout se debe de eliminar el access token de la sesión para asegurarse de que el usuario ya no esté autenticado, para ello se emplea “sesión.pop”. Después de esto, se redirige al usuario a la página principal, que como se acaba de explicar, al no detectar token en la sesión llevará al usuario de vuelta al login.

```
35. @app.route("/logout") # Define la ruta para manejar las solicitudes a /logout
36. def logout():
37.     session.pop("access_token", None)
38.     return redirect(url_for("index"))
```

La última ruta por definir es la que maneja las llamadas a la API Graph. Primero se ha de comprobar si el access token está presente en la sesión, si no lo está el usuario es redirigido a la página de login. Si el token está presente, se realiza una solicitud GET a la API, pasando el token en la cabecera de autorización. El resultado de la API se obtiene en formato JSON y se redirige a la plantilla `display.html` para su mostrarlo por pantalla.

```
39. # Define la ruta que maneja solicitudes a /call_downstream_api
40. @app.route("/call_downstream_api")
41. def call_downstream_api():
42.     if "access_token" not in session:
43.         return redirect(url_for("login"))
44.     api_result = requests.get(
45.         app_config.ENDPOINT,
46.         headers={'Authorization': 'Bearer ' + session["access_token"]},
47.         timeout=30,
48.     ).json()
49.     return render_template('display.html', result=api_result)
```

Finalmente, para ejecutar la aplicación se utiliza `app.run()`, que inicia el servidor web de desarrollo de Flask, permitiendo que la aplicación acepte conexiones.

```
50. if __name__ == "__main__":
51.     app.run()
```

El diseño de las diferentes páginas de la aplicación se realiza con HTML, utilizando un archivo CSS para mejorar la presentación gráfica. En particular hay un HTML de gran importancia, ya que cuando se carga tras completar la autenticación y autorización ejecuta un script que gestiona la extracción del token de la URL.

Lo primero que hace el script es emplear varias funciones para encontrar el access token en la URL y almacenarlo en la variable token.

```
1. <script type="text/javascript">
2. function getTokenFromUrl() {
3.     const hash = window.location.hash.substring(1);
4.     const params = new URLSearchParams(hash);
5.     const token = params.get("access_token");
```

A continuación comprueba si se ha encontrado el token. Si es así, crea un formulario HTML dinámico para enviar el token al backend mediante una solicitud POST.

```
6. if (token) {
7.     const form = document.createElement("form");
8.     form.method = "POST";
9.     form.action = "{{ url_for('auth_response') }}";
10.
11.     const hiddenField = document.createElement("input");
12.     hiddenField.type = "hidden";
13.     hiddenField.name = "access_token";
14.     hiddenField.value = token;
15.
16.     form.appendChild(hiddenField);
17.     document.body.appendChild(form);
18.     form.submit();
```

Y en caso de no encontrar token, redirige al usuario a la página de inicio de sesión para que inicie el proceso de autenticación nuevamente.

```
19. } else {
20.     window.location.href = "{{ url_for('login') }}";
21. }
```

6.1.3 Funcionamiento

Con todo lo anterior realizado, la aplicación ya es funcional, por lo que solo queda verificar que funciona de manera correcta. Para ello, la aplicación se ejecutará en localhost empleando el puerto 5000, ejecutando el comando: `flask run --host=localhost --port=5000`

El usuario accede a la aplicación a través de la URL `http://localhost:5000` y es redirigido inmediatamente al login de Microsoft donde debe iniciar sesión utilizando una cuenta del tenant donde se ha registrado la aplicación.

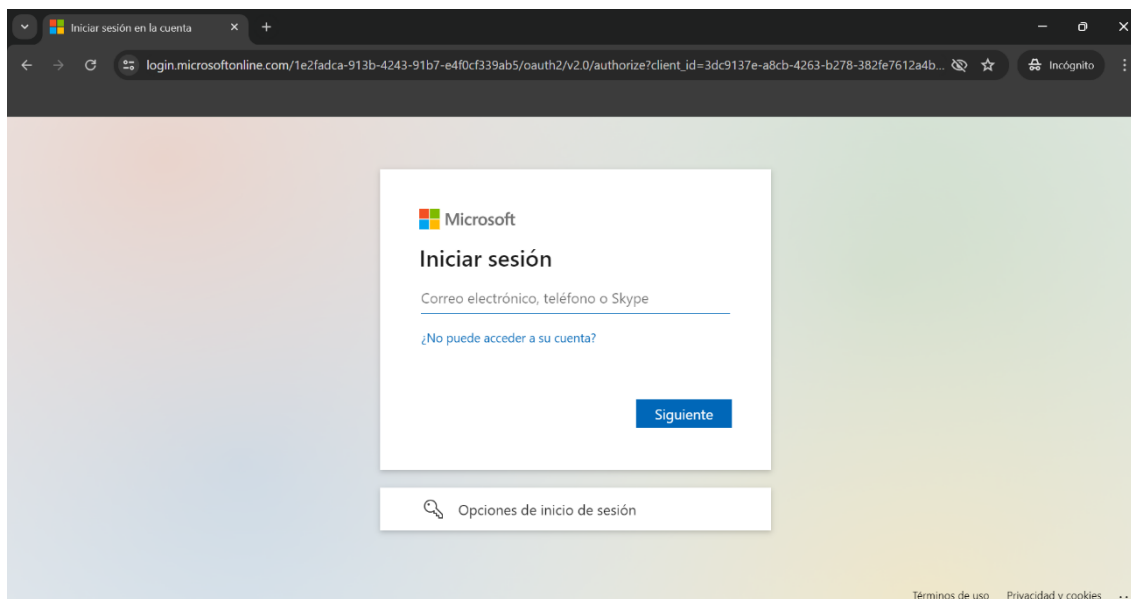


Ilustración 33 - Página de login de Microsoft (Elaboración propia)

Al acceder correctamente a la aplicación, inmediatamente se le presentará al usuario un pop-up donde la aplicación informa de los permisos que requiere para funcionar y pide al usuario que le autorice.

En este punto, el flujo se ha completado y Entra ID ha emitido el token de acceso, que ya está en posesión del cliente. El usuario es redirigido a la página principal, donde se le presentan dos opciones: hacer logout o llamar a la API Graph.

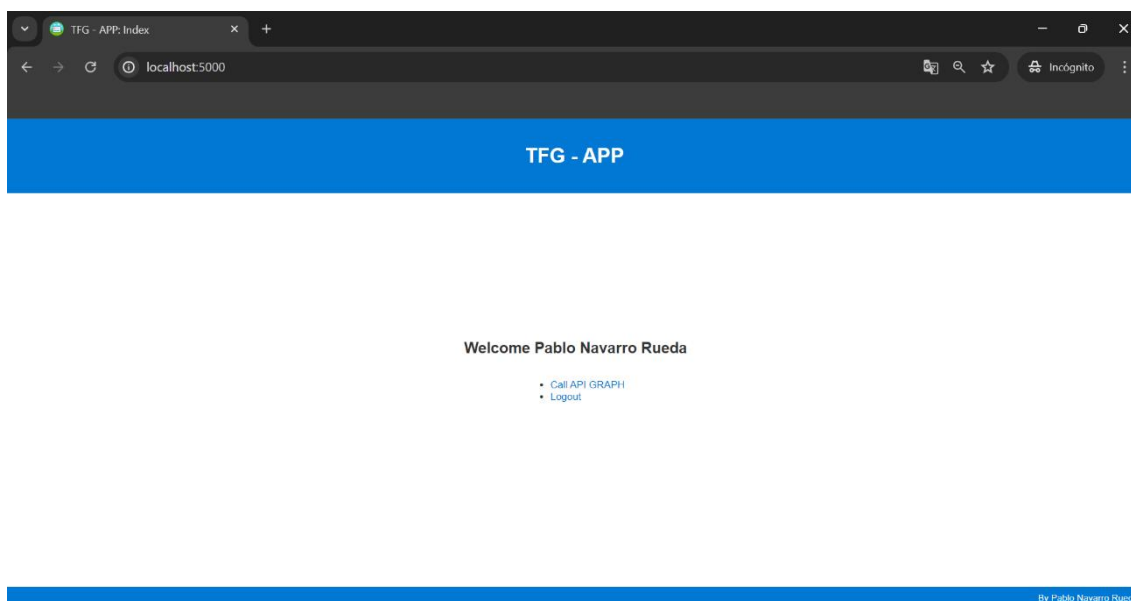
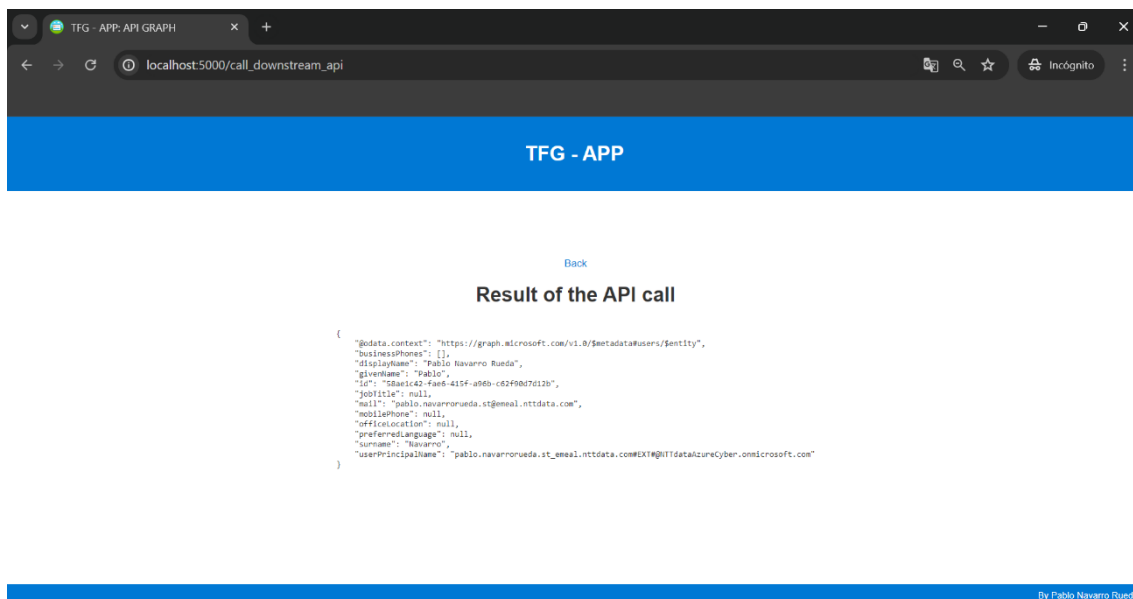


Ilustración 34 - Página principal (Elaboración propia)

Si se selecciona la opción de logout, el usuario volverá a la página del login de Microsoft y tendrá que volver a iniciar sesión. Al seleccionar la opción de llamar a Graph, la aplicación utiliza el access token que acaba de almacenar para así poder acceder a los datos del usuario y se redirige al usuario al HTML Display donde se muestran los datos del usuario obtenidos a través de una consulta a Graph.

**Ilustración 35 - Llamada a Graph (Elaboración propia)**

6.1.4 Problemas de la implementación

Aunque como se ha comprobado el funcionamiento de la aplicación es correcto, la configuración actual presenta varios riesgos de seguridad, principalmente debido al uso de Implicit Flow. Uno de los principales riesgos de este flujo es la inclusión del access token en la URL de redirección para que el cliente pueda extraerlo.

A continuación se va a simular un entorno en el cual el atacante consigue superar el cifrado de HTTPS, consiguiendo conectarse a la misma red que el usuario de la aplicación. En una situación así el atacante podría realizar un ataque Man-in-the-Middle interceptando el tráfico y logrando obtener el access token.

Para mostrar como el atacante puede lograr esto se ha utilizado Wireshark, una herramienta que permite analizar el tráfico de red. Dado que el backend de la aplicación se ejecuta en localhost, Wireshark se ha de configurar para escuchar en el canal de loopback, situándose así entre el navegador del cliente y su servidor backend.

Primero, se inicia una captura y se accede a la aplicación tal y como se ha hecho anteriormente. Al hacerlo, los distintos paquetes de tráfico capturados van apareciendo en Wireshark. Aplicando el filtro "http" para ver solo los paquetes relevantes, se puede identificar uno especialmente significativo.

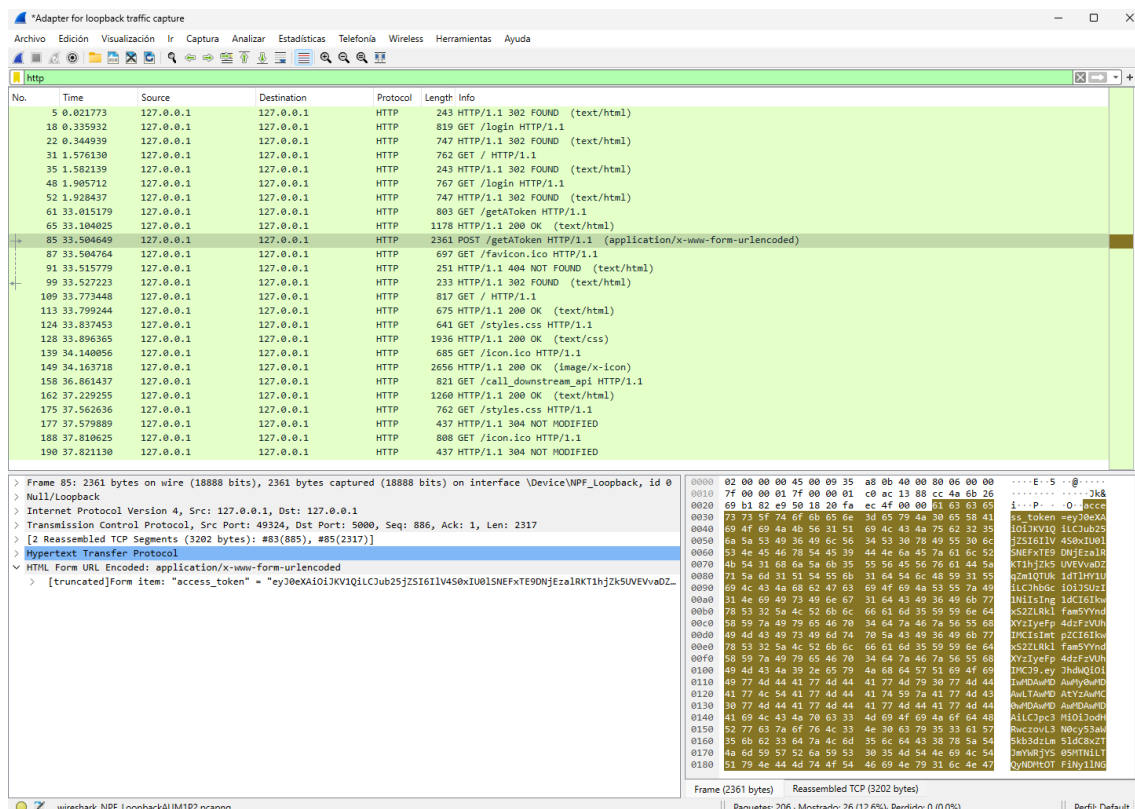


Ilustración 36 - Captura con Wireshark del tráfico con Implicit Flow (Elaboración propia)

La petición mostrada en detalle en la imagen anterior es una petición POST que el navegador del cliente realiza a su backend. Esto se sabe ya que el origen de esta es el puerto 49324 (navegador) y el destino es el puerto 5000 (backend) que ha sido asignado previamente con el comando de ejecución. El motivo de esta petición es que, una vez se ha extraído el access token de la URL de redirección, el navegador debe enviar este token a su servidor para almacenarlo de forma segura y poder utilizarlo cuando sea necesario sin la necesidad de obtener uno nuevo. Abajo a la derecha de la ilustración, se puede ver claramente cómo el contenido de la petición incluye el access token completo “access_token=...”.

6.1.5 Análisis del token

Que el atacante consiga obtener el access token es algo crítico por la información que contiene. Para ver la información que se puede extraer del token capturado se ha realizado un análisis de este en una aplicación web de Okta que analiza JSON Web Token (JWT).

```
HEADER:
{
  "typ": "JWT",
  "nonce": "UxKLHSIR4AqLOC613jTJ0XcFNTTEoh6jfmPMI5u9Gcl",
  "alg": "HS512",
  "x5t": "L1KfKFI_jnXbwWc22xZxw1sUHH0",
  "kid": "L1KfKFI_jnXbwWc22xZxw1sUHH0"
}
```

En el encabezado del token se encuentra información sobre el tipo de token y el algoritmo empleado para la firma. Como se puede observar el access token es de tipo JWT y usa un algoritmo HS512 para la firma que utiliza la función hash SHA-512. También se encuentra el nonce, un valor único cuya función es evitar ataques de repetición, x5t que es la clave pública del certificado x.509 que se ha utilizado para firmar el token, y el kid que es el identificador de clave y sirve para saber cual es la firma correspondiente cuando hay varias disponibles.

PAYLOAD:

```
{
  "aud": "00000003-0000-0000-c000-000000000000",
  "iss": "https://sts.windows.net/1e2fadca-913b-4243-91b7-e4f0cf339ab5/",
  "iat": 1718643360,
  "nbf": 1718643360,
  "exp": 1718649050,
  "acct": 1,
  "acr": "1",
  "acrs": [
    "urn:user:registersecurityinfo"
  ],
  "aio":
    "AWQAm/8XAAAAQ4yc5Z2R7GpUj2NV9uvSf8StDeAdBsJAKEijXVBb16WVh1nXoQ8IqTCmEZPEM8sBBab2DsHPFI6VsVFO4V1Q/AxPL/LuTULbySyV3ayDjraHD/HnM7Vi3Br1XybBeyCp",
  "altsecid": "5::1003200357669F25",
  "amr": [
    "pwd"
  ],
  "app_displayname": "TFG - APP",
  "appid": "3dc9137e-a8cb-4263-b278-382fe7612a4b",
  "appidacr": "0",
  "email": "pablo.navarrorueda.st@emeal.nttdata.com",
  "family_name": "Navarro",
  "given_name": "Pablo",
  "haswids": "true",
  "idp": "https://sts.windows.net/3048dc87-43f0-4100-9acb-ae1971c79395/",
  "idtyp": "user",
  "ipaddr": "84.120.67.242",
  "name": "Pablo Navarro Rueda",
  "oid": "58ae1c42-fae6-415f-a96b-c62f90d7d12b",
  "platf": "3",
  "puid": "100320035F3C3317",
  "rh": "0.AU4Ayq0vHjuRQ0KRt-TwzzOatQMAAAAAAAAAAwAAAAAAAAAOAV0.",
  "scp": "User.Read profile openid email",
  "signin_state": [
    "kmsi"
  ],
  "sub": "s0zmD20506SqeJdtFiLnMkrA2Ne000fVkt6tiHSxOQY",
  "tenant_region_scope": "EU",
  "tid": "1e2fadca-913b-4243-91b7-e4f0cf339ab5",
  "unique_name": "pablo.navarrorueda.st@emeal.nttdata.com",
  "uti": "iskJYtMBIESuknTlXriAAA",
  "ver": "1.0",
  "xms_idrel": "5 16",
  "xms_st": {
    "sub": "ytJBjmYR2YhgXjPPfF2o_v223KWL7fd8FRllyyF8NNmQ"
  },
  "xms_tcdt": 1634716040,
  "xms_tdbr": "EU"
}
```

En cuanto a la información que el token contiene en el payload, puede clasificarse en tres categorías principales que pueden ser explotadas por un atacante:

- **Datos personales:** es toda la información que permite identificar directamente al usuario, como el campo email, donde aparece el correo del usuario; los campos name, given_name y family_name, donde aparece el nombre completo del usuario; el campo unique_name, donde se muestra el User Principal Name (UPN) que el usuario tiene en Entra ID; y el campo ipaddr, que contiene la dirección IP del usuario en el momento en que se emitió el token.
- **Datos de sesión:** son aquellos datos que incluyen información sobre la autenticación y el estado de la sesión, y que pueden ser utilizados para ataques de repetición o secuestro de sesión. El campo iat indica el timestamp de emisión del token; nbf y exp indican el inicio y fin de la validez del token; acr es el nivel de robustez de la autenticación del usuario;

amr muestra el método de autenticación utilizado, que en este caso es contraseña; aio y rh son cadenas con información específica de la sesión; y sub es el identificador único del usuario en el contexto del token.

- **Contexto de aplicación:** son los identificadores y permisos que indican cómo y dónde se utiliza el token, proporcionando información sobre el entorno y posibles vulnerabilidades a explotar. El parámetro iss indica la autoridad que emite el token; idp muestra el proveedor de identidad del usuario; appid y app_displayname contienen el application id y el nombre de la aplicación en el tenant; scp es el scope, los permisos que otorga el token; y en cuanto al tenant, tenant_region_scope y tid indican la región, en este caso Europa, y el identificador de este.

Con toda esta información en poder del atacante puede llevar a cabo todo tipo de ataques como de suplantación de identidad, phishing, de repetición o de secuestro de sesión, que pueden llevar al acceso no autorizado a recursos sensibles o al robo de datos. Para evitarlo, se van a realizar una serie de medidas con el fin de proteger el token y los accesos.

6.2 Transición a Authorization Code Flow

Una vez demostrado el riesgo de seguridad que supone que la aplicación funcione con Implicit Flow, se realizarán los cambios necesarios para que la aplicación ahora funcione con Authorization Code Flow. Como se ha visto anteriormente, se añadirá un client secret y Entra ID no enviará directamente el access token, sino que emitirá un code que el cliente deberá almacenar. Para solicitar el token, el cliente necesitará enviar una solicitud con el client secret y el code, de forma que el atacante solo podría obtenerlo si consigue estos dos valores.

6.2.1 Cambios en Entra ID

Para configurar el nuevo flujo, es necesario realizar varios ajustes desde el lado de Entra ID, comenzando por la creación de un client secret. Esto se hace desde el menú de App Registrations, donde se debe buscar la aplicación y acceder a su configuración. Una vez dentro, en el apartado “Certificates & Secrets”, se debe acceder a la sección “Client secrets” y seleccionar la opción “New client secret”. Simplemente se tiene que añadir, de forma opcional, una descripción de la utilidad del secret y establecer su duración. Por motivos de seguridad, es recomendable que la duración sea lo más breve posible y que se vaya renovando, por lo que basta con 6 meses.

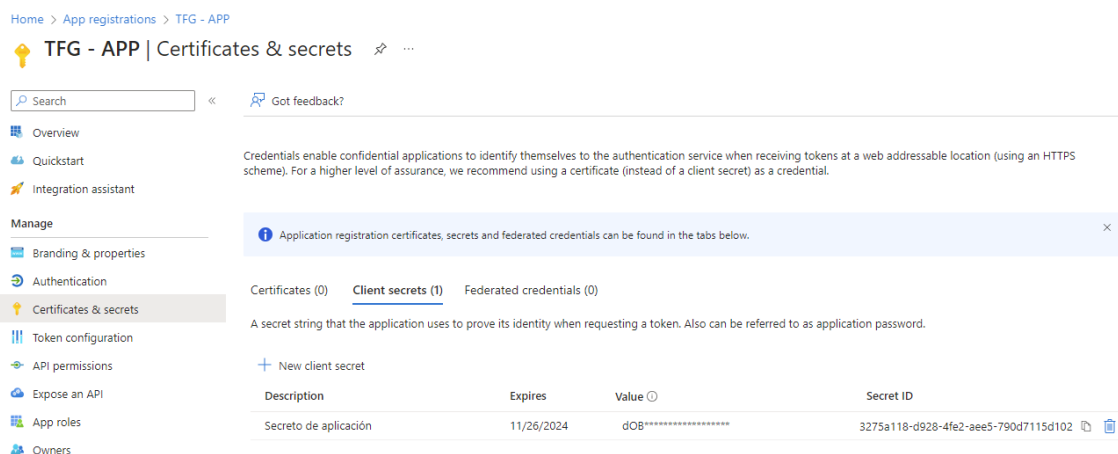


Ilustración 37 - Creación de un client secret desde Entra ID (Elaboración propia)

Es muy importante guardar el valor del secret de inmediato tras crearlo, ya que este valor no puede ser visto nuevamente después de su creación, ni siquiera por un administrador. Este valor del secreto se añade al archivo “.env”, ya que este valor será necesario para algunas de las peticiones durante el flujo.

Por último, se ha de visitar el apartado “Authentication” para desactivar las dos opciones previamente activadas para que la aplicación funcionara con Implicit Flow, ya que para que la app funcione con Authorization code flow no deben estar marcadas.

Home > App registrations > TFG - APP

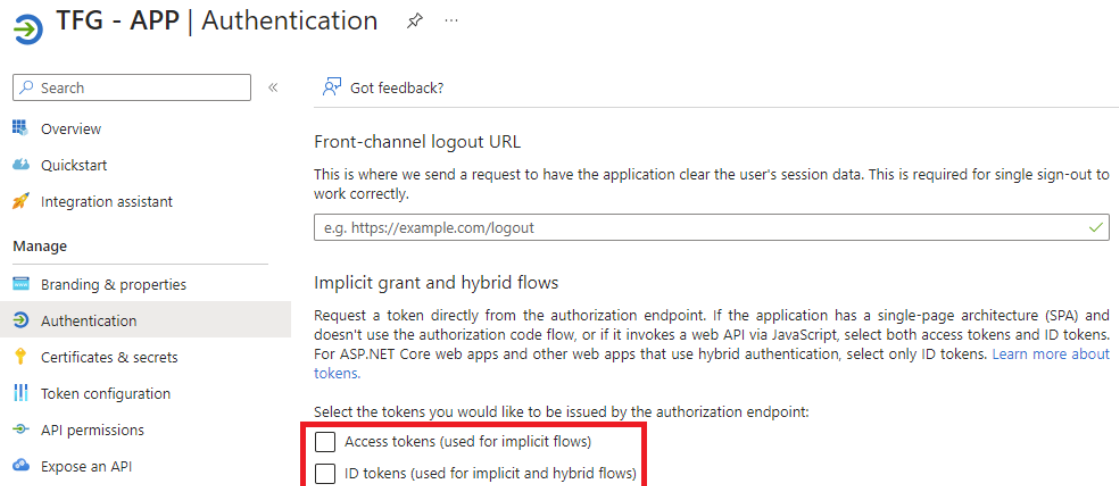


Ilustración 38 - Configuración para Implicit Flow (Elaboración propia)

6.2.2 Cambios en el código

Por la parte del código, hay que introducir el valor del client secret en el archivo “.env” para que la aplicación pueda acceder a él.

```
1. CLIENT_ID=3dc9137e-a8cb-4263-b278-382fe7612a4b
2. AUTHORITY=https://login.microsoftonline.com/1e2fadca-913b-4243-91b7-e4f0cf339ab5
3. CLIENT_SECRET=d0B8Q~44-E2b05jYRHD5d60sQIdkZjgIN~Zyhcv.
```

También es necesario crear una variable CLIENT_SECRET en el archivo app_config.py que obtenga su valor extrayéndolo del archivo “.env”

```
1. CLIENT_SECRET = os.getenv("CLIENT_SECRET")
```

En cuanto al código principal de app.py, se deben realizar varios cambios importantes. Primero, importar la librería identity.web que permitirá manejar la autenticación y autorización de manera segura.

```
1. import identity.web
```

Con esta librería, se crea el objeto “auth” para gestionar la sesión del usuario, su función es centralizar y simplificar la autenticación, manejando el token de forma segura.

```
1. app.jinja_env.globals.update(Auth=identity.web.Auth) #Agrega variable global
2. #Iniciación del Objeto de Autenticación
3. auth = identity.web.Auth(
4.     session=session,
5.     authority=app.config["AUTHORITY"],
6.     client_id=app.config["CLIENT_ID"],
7.     client_credential=app.config["CLIENT_SECRET"],
8. )
```

La función “login” cambia completamente. Ahora, la URL de autorización se construye a través del método “auth.log_in()”, que asegura que se sigan correctamente los parámetros requeridos y recomendados por Microsoft, a la hora de implementar Authorization Code Flow.

```
1. def login():
2.     return render_template("login.html", version=__version__, **auth.log_in(
```

```
3.         scopes=app_config.SCOPE,  
4.         redirect_uri=url_for("auth_response", _external=True),  
5.         prompt="select_account",  
6.     ))
```

El manejo de la respuesta de autenticación también cambia. Ahora se maneja a través del método “auth.complete_log_in(request.args)”, que gestiona la recepción del code, su almacenamiento en el servidor, la petición del token de acceso usando el code y el client secret, y la recepción segura del token directamente en el backend.

```
1. @app.route(app_config.REDIRECT_PATH)  
2. def auth_response():  
3.     result = auth.complete_log_in(request.args)  
4.     if "error" in result:  
5.         return render_template("auth_error.html", result=result)  
6.     return redirect(url_for("index"))
```

La comprobación en la página principal para ver si el usuario está logeado correctamente ahora también verifica que haya “CLIENT_SECRET” además de “CLIENT_ID”. Además, se usa el método “auth.get_user()” para comprobar que el usuario está logeado correctamente, redirigiendo al login en caso de error.

```
1. @app.route("/")  
2. def index():  
3.     if not (app.config["CLIENT_ID"] and app.config["CLIENT_SECRET"]):  
4.         return render_template('config_error.html')  
5.     if not auth.get_user():  
6.         return redirect(url_for("login"))  
7.     return render_template('index.html', user=auth.get_user(), version=__version__)
```

Por último el proceso de logout también cambia, en lugar de eliminar el access token de la sesión y redirigir al usuario, ahora se usa el método “auth.log_out()”, que se encarga de todo el proceso con mayor seguridad.

```
1. @app.route("/logout")  
2. def logout():  
3.     return redirect(auth.log_out(url_for("index", _external=True)))
```

Con el cambio a Authorization Code Flow, ya no es necesario el HTML “auth_response” ni el script para extraer el token de la URL. Ahora, la recepción del token se gestiona directamente en el método “auth_response” en el backend tal y como se ha comentado.

6.2.3 Funcionamiento

El cambio en el flujo se realiza de forma transparente para el usuario por lo que no hay ningún cambio en el funcionamiento de la aplicación. Únicamente se ha añadido una página de login desde la que clicando en el botón se redirige al usuario al login de Microsoft. Es un cambio simplemente a nivel visual para tener una página de bienvenida a la app, ya que antes era imposible al ser necesario para el flujo la redirección inmediata al portal de inicio de sesión de Microsoft.

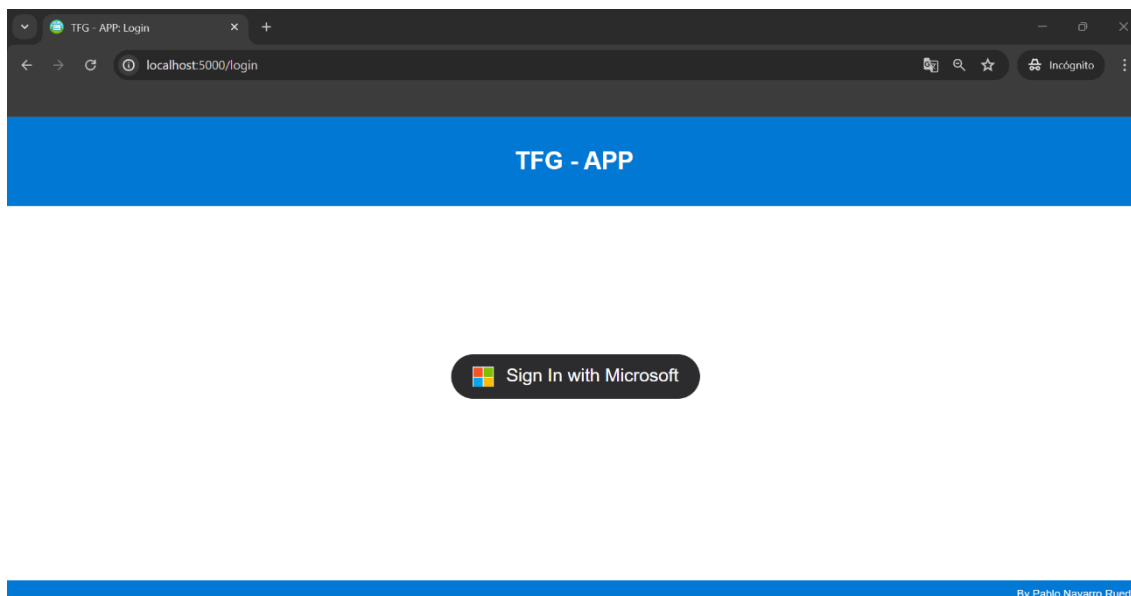


Ilustración 39 - Página de inicio nueva (Elaboración propia)

6.2.4 Análisis de la securización

Como se hizo para el Implicit Flow, se va a intentar capturar el token de acceso utilizando Wireshark para interceptarlo. La configuración de Wireshark y los pasos para seguir son exactamente los mismos. Tras iniciar la captura y acceder a la aplicación, los paquetes interceptados vuelven a aparecer, pero esta vez, el token no aparece en ninguna de las peticiones que se han capturado.

Esto se debe a que, a diferencia del Implicit Flow, en el que el token se enviaba al navegador del cliente y permitía al atacante interceptarlo cuando se intentaba pasar del navegador al servidor para almacenarlo, con el Authorization Code Flow el token viaja directamente de Entra ID al backend del cliente. Esto demuestra que el token deja de estar expuesto en el navegador y ya no basta con interponerse entre el navegador y el servidor para capturarlo.

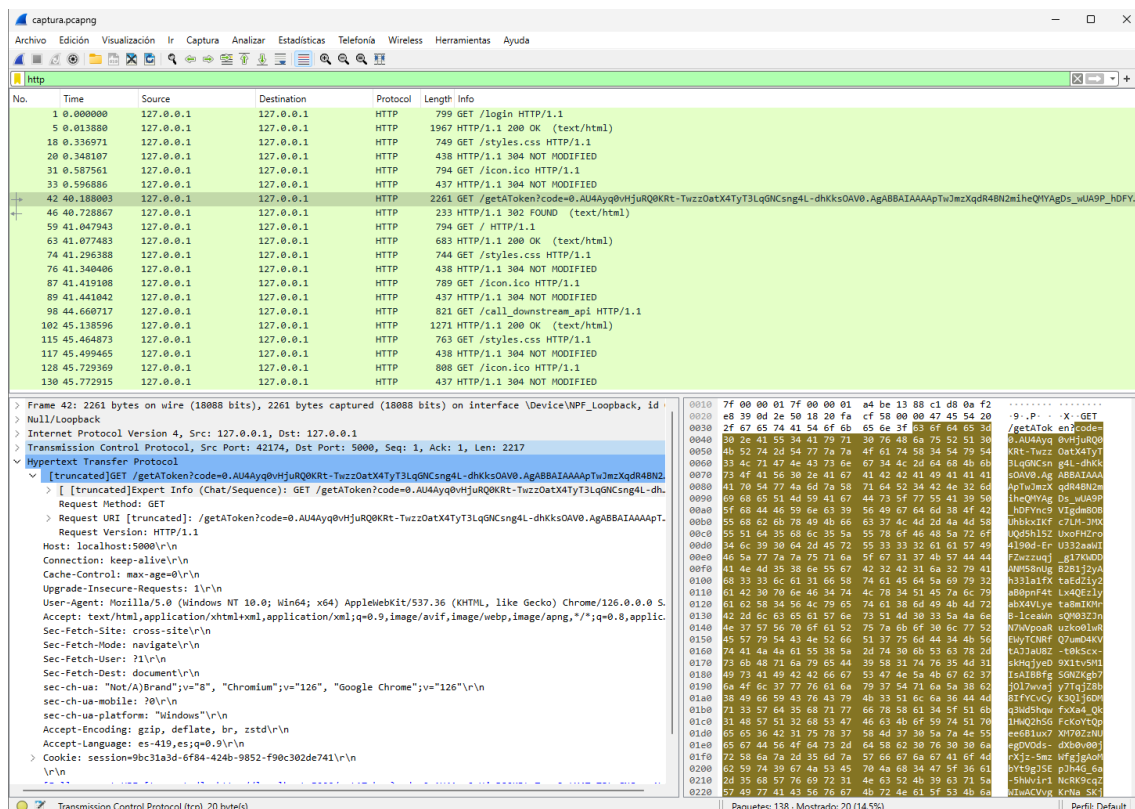


Ilustración 40 - Captura de Wireshark con Authorization Code Flow (Elaboración propia)

Aun así, se logra capturar algo interesante para el atacante, es el code en el momento que el cliente lo envía a su backend para almacenarlo, tal y como se había capturado anteriormente el access token con Implicit Flow. Sin embargo, esto ya no es tan sensible porque, para obtener el access token, no es suficiente con el code sino que el atacante también deberá disponer del client secret. Tener uno sin el otro no aporta nada, y además, el code es de un solo uso por lo que no presenta un riesgo significativo. De esta forma, se demuestra que con el cambio realizado se ha reducido en gran medida la vulnerabilidad a ataques Man-in-the-Middle.

6.3 Proteger la identidad y los accesos a la aplicación

Aunque ya se ha mejorado notablemente la seguridad, se pueden aprovechar las diversas herramientas y funcionalidades que proporciona Entra ID para seguir mejorándola. A continuación se va a restringir el acceso a la app, se van a implementar políticas de acceso condicional y se va a requerir un MFA para acceder a la aplicación.

6.3.1 Bloquear acceso salvo a usuarios asignados

Un buen comienzo para proteger la aplicación de accesos no deseados es restringir la aplicación salvo a usuarios específicos. Desde el menú de “Enterprise apps” se ha de buscar la aplicación, acceder y dirigirse a “Properties”. Ahí hay que cambia el “Assignment required” a “yes”, de forma que solo los usuarios asignados podrán acceder a la aplicación.

Una buena práctica que permite controlar mejor quien puede acceder es crear un grupo que incluya a todos los usuarios autorizados, ya que así se simplifica la administración. El grupo se llama “Access TFG-APP” y desde el menú de grupos se puede añadir los usuarios que necesiten acceso. Para la asignación del grupo a la app se hace desde “Enterprise apps”, accediendo a la aplicación y, en “Users and groups”, seleccionando “Add user/group” para buscar y añadir el grupo que se ha creado.

Ahora cuando un usuario que no pertenece al grupo intenta acceder a la app se le bloquea el acceso con el siguiente mensaje.

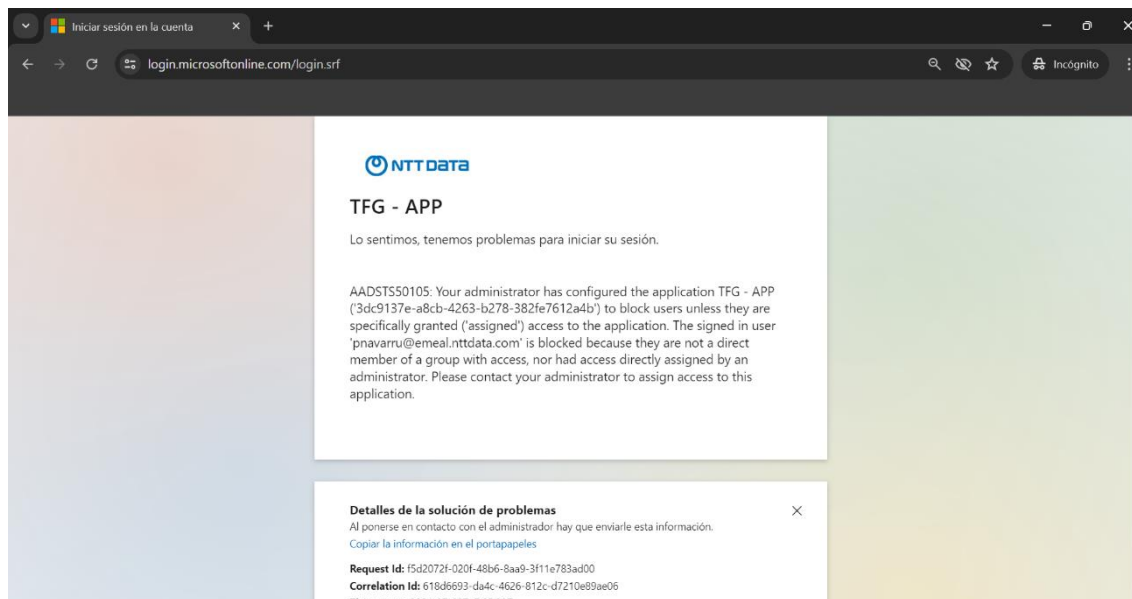


Ilustración 41 - Bloqueo a usuario no asignado a la app (Elaboración propia)

6.3.2 Definir política de Conditional Access para el bloqueo

Para lograr un mayor grado de seguridad en la aplicación, se definirá una política de Conditional Access que denegará el acceso a menos que se cumplan ciertos requisitos. El objetivo es que si un atacante consigue las credenciales de un usuario, con eso no será suficiente para lograr acceder ya que también debería cumplir con las condiciones. Primero, es necesario identificar los casos específicos que se quieren bloquear, para esta aplicación solo se permitirá el acceso si el usuario cumple una de las siguientes condiciones: acceder desde una IP de confianza o hacerlo a través de un dispositivo de confianza.

Para configurar esto, primero se debe crear una "named location" que incluya todas las IPs consideradas de confianza y desde las cuales se va a permitir a los usuarios acceder, como pueden ser las IPs de la organización que necesita acceso a la aplicación. En cuanto a los dispositivos de confianza, se requiere que la empresa proporcione dispositivos a sus empleados y que estos estén marcados como de confianza (compliant), solo se permitirá el acceso desde estos dispositivos corporativos para evitar que se acceda desde un dispositivo personal que pueda estar comprometido.

[Home](#) > [Conditional Access | Policies](#) >

New

Conditional Access policy

Control access based on Conditional Access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name *

CA01-TFG APP-Block ✓

Assignments

Users

All users

Target resources

1 app included

Network **NEW**

Any network or location and 1 excluded

Conditions

2 conditions selected

Access controls

Grant

Block access

Session

0 controls selected

Control access based on signals from conditions like risk, device platform, location, client apps, or device state. [Learn more](#)

Device platforms

Not configured

Locations

Any network or location and 1 excluded

Client apps

Not configured

Filter for devices

Exclude filtered devices

Authentication flows (Preview)

Not configured

Enable policy

Report-only On Off

Create

Ilustración 42 - Creación de la política de bloqueo (Elaboración propia)

En la Ilustración 42 se muestra la configuración de la política, que en la sección "Grant" se ve como está configurada para bloquear el acceso a todos los usuarios que intenten acceder a la aplicación (se selecciona la app en "Target resources"). Sin embargo, para evitar el bloqueo cuando se cumplan las condiciones establecidas, se ha excluido la "named location" creada, de manera que si el acceso proviene de una de esas IPs, la política no se aplicará. Además, se han excluido los dispositivos marcados compliant, asegurando que solo los accesos legítimos sean permitidos.



You cannot access this right now

Your sign-in was successful but does not meet the criteria to access this resource. For example, you might be signing in from a browser, app, or location that is restricted by your admin.

[Sign out and sign in with a different account](#)

[More details](#)

Ilustración 43 - Mensaje de bloqueo por política de Conditional Access (Elaboración propia)

Ahora cuando un usuario intenta acceder y es bloqueado por esta política recibirá el error de la ilustración 43 tras completar el login, en el que se explica que a pesar de que las credenciales que ha introducido si son correctas, no cumple las condiciones para acceder al recurso.

6.3.3 Definir política de Conditional Access para solicitar MFA

Después de establecer una política de bloqueo general, es una buena práctica proteger todos los accesos con MFA. Para ello se va a crear una política para solicitar MFA a todos los usuarios que quieran acceder. De esta manera, si un atacante logra obtener las credenciales de un usuario y accede mediante un dispositivo compliant o a través de las IPs de confianza, no podrá acceder si no completa el MFA requerido.

Home > Conditional Access | Policies >

New

Conditional Access policy

Control access based on Conditional Access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name *

CA02-TFG APP-Require MFA ✓

Assignments

Users ⓘ

All users

Target resources ⓘ

1 app included

Network NEW ⓘ

Not configured

Conditions ⓘ

0 conditions selected

Access controls

Grant ⓘ

1 control selected

Session ⓘ

0 controls selected

Enable policy

Report-only On Off

Create

Grant

Control access enforcement to block or grant access. [Learn more](#)

☐ Block access

☒ Grant access

☐ Require multifactor authentication ⓘ

⚠ "Require authentication strength" cannot be used with "Require multifactor authentication". [Learn more](#)

☒ Require authentication strength ⓘ

Authenticator App

ℹ To enable all authentication strengths, configure cross-tenant access settings to accept claims coming from Microsoft Entra tenants for external users. Authentication strengths will only configure second factor authentication for external users. [Learn more](#)

☐ Require device to be marked as compliant ⓘ

☐ Require Microsoft Entra hybrid joined device ⓘ

☐ Require approved client app ⓘ

[See list of approved client apps](#)

Select

Ilustración 44 - Creación de política para requerir MFA (Elaboración propia)

En la Ilustración 44 se muestra la configuración de la política. Al igual que en la anterior, hay que establecer que esta política se aplique específicamente a la app y a cualquier usuario que intente acceder. En la sección "Grant", ahora no se configura para bloquear si no para permitir el acceso siempre que se cumpla con la condición establecida. En lugar de seleccionar "Require multifactor authentication", se recomienda usar "Require authentication strength" para mayor seguridad ya que con esta opción se limitan los métodos de MFA permitidos. En este caso, solo se permitirá la autenticación mediante la app Authenticator, ya que se considera uno de los métodos más seguros.

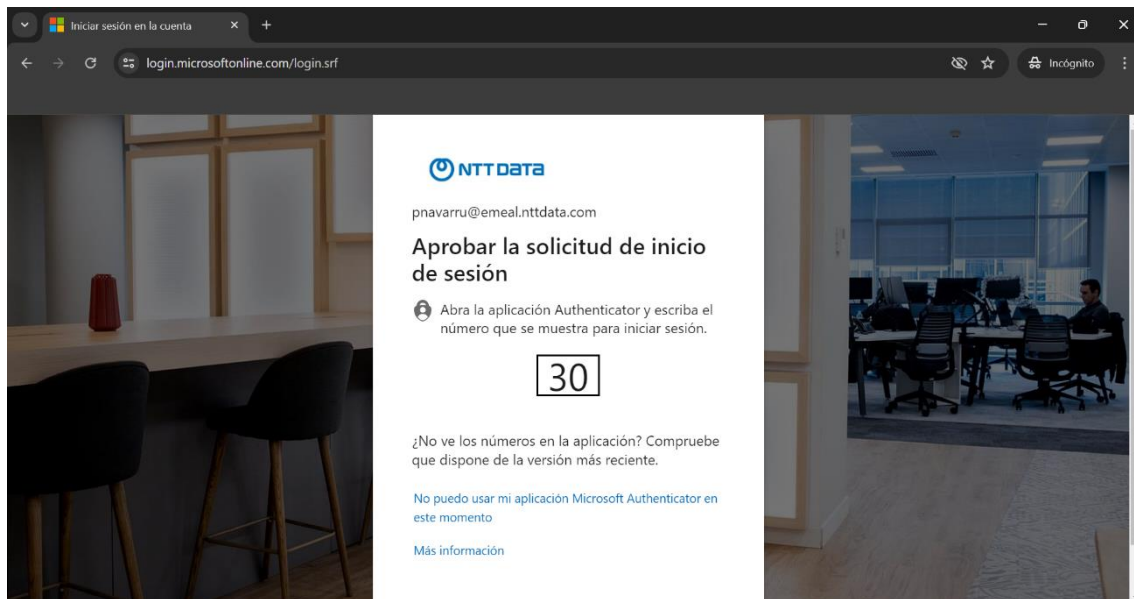


Ilustración 45 - MFA de Authenticator App requerido (Elaboración propia)

Como se observa en la ilustración 45, cada vez que un usuario quiera acceder deberá completar el desafío del MFA, introduciendo el numero mostrado en la pantalla del navegador en la aplicación móvil.

Capítulo 7. Conclusiones

7.1 Conclusiones y trabajos futuros

Este TFG ha explorado la autenticación y autorización en aplicaciones web mediante la integración con Entra ID, demostrando cómo una correcta implementación de flujos OAuth, como el Authorization Code Flow, mejora significativamente la seguridad. La transición del Implicit Flow al Authorization Code Flow ha sido efectiva en la mitigación de vulnerabilidades, reduciendo el riesgo de ataques Man-in-the-Middle y asegurando que el access token no se exponga en el navegador del cliente. Este trabajo resalta la importancia de prácticas de seguridad adecuadas para proteger las aplicaciones web en un entorno cloud.

En cuanto a trabajos futuros, la Gestión de Identidades Digitales (GID) seguirá siendo un componente crucial en la seguridad de los activos digitales de una compañía. A medida que más aplicaciones migren sus procesos de autenticación y autorización a la nube, aprovechando sus ventajas, será esencial mantener y mejorar las prácticas de seguridad.

Una forma de ampliar este trabajo es el estudio detallado de ataques más avanzados a aplicaciones integradas con Entra ID, tratando de explotar vulnerabilidades dentro de los flujos más seguros. Además, se podría considerar la integración de mecanismos adicionales de seguridad, como el uso de tokens de refresco, la adopción de tecnologías para autenticación sin contraseña y la implementación de políticas de acceso más robustas.

Otra área de interés es el uso de inteligencia artificial para la detección y prevención de ataques. Entra ID ha incorporado recientemente "Identity protection", una herramienta basada en IA que mejora la capacidad de detectar y mitigar amenazas en tiempo real analizando el comportamiento del usuario durante el proceso de autenticación. Aplicar soluciones que incorporen Inteligencia Artificial en la GID abriría nuevas vías para aumentar la seguridad de los activos en la nube.

7.2 Relación con los estudios cursados

En el último año del grado, en la mención de telemática, he cursado asignaturas sobre seguridad y desarrollo de aplicaciones web. Este Trabajo de Fin de Grado explora en profundidad la autenticación y autorización en una aplicación web, integrándola con Entra ID como proveedor de identidad. El proyecto aborda cómo Entra ID maneja estos procesos cruciales y destaca la importancia de utilizar el flujo OAuth adecuado cuando se busca la mayor seguridad posible para los datos que maneja la aplicación.

A lo largo del TFG, se ha demostrado cómo asegurar que los datos manejados por la aplicación permanezcan protegidos mediante la correcta implementación de prácticas de seguridad recomendadas. Esto incluye la configuración y manejo seguro de tokens de acceso, la protección contra ataques Man-in-the-Middle, y la correcta gestión de las sesiones del usuario. De esta manera, se aplican y consolidan los conocimientos adquiridos en las asignaturas de seguridad y desarrollo web, proporcionando una comprensión práctica y aplicada de estos conceptos en un entorno real.



Agradecimientos

Con este proyecto concluye una etapa de cuatro años, un logro alcanzado gracias a la ayuda de muchas personas que me han acompañado a lo largo del camino.

Quiero comenzar agradeciendo a mi tutor, José Enrique López Patiño, por enseñarme las bases de la ciberseguridad y guiarme en el descubrimiento de este fascinante mundo.

Agradezco a NTT DATA, no solo por brindarme la oportunidad de desarrollarme profesionalmente, sino también por el buen ambiente que se respira en el equipo de Ciberseguridad. Especialmente, quiero agradecer a mis compañeros de equipo, Dani, Mario y Alberto, por conseguir que el día a día trabajando haya sido totalmente disfrutable a la par que enriquecedor para mi formación como profesional. Sin duda, este equipo es el mejor recuerdo que guardaré de mis prácticas.

Por último, quiero agradecer a mis amigos, familia y a mi pareja, quienes me han acompañado a lo largo de todo este camino. Ellos son mi principal punto de apoyo y me han mantenido fuerte en los momentos más complicados.

Bibliografía

- [*] Okta, «Client Credentials Flow,» [En línea]. Available: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/client-credentials-flow>. [Último acceso: 15 Mayo 2024].
- [*] Okta, «Implicit Flow with Form Post,» [En línea]. Available: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/implicit-flow-with-form-post>. [Último acceso: 11 Mayo 2024].
- [*] Okta, «Resource Owner Password Flow,» [En línea]. Available: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/resource-owner-password-flow>. [Último acceso: 15 Mayo 2024].
- [*] Okta, «Device Authorization Flow,» [En línea]. Available: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/device-authorization-flow>. [Último acceso: 16 Mayo 2024].
- [*] Cloudflare, «¿Qué es un proveedor de identidad (IdP)?,» [En línea]. Available: <https://www.cloudflare.com/es-es/learning/access-management/what-is-an-identity-provider/>. [Último acceso: 23 Abril 2024].
- [*] R. Neto, «Flujos de autorización con OAuth 2.0 y OpenID Connect,» 28 Febrero 2023. [En línea]. Available: <https://rafaelneto.dev/blog/flujos-autorizacion-oauth-2-0-openid-connect/>. [Último acceso: 30 Abril 2024].
- [*] ICHI.PRO, «La guía completa de los protocolos OAuth 2.0 y OpenID Connect,» [En línea]. Available: <https://ichi.pro/es/la-guia-completa-de-los-protocolos-oauth-2-0-y-openid-connect-59286684942618>. [Último acceso: 30 Abril 2024].
- [*] Okta, «Authorization Code Flow with Proof Key for Code Exchange (PKCE),» [En línea]. Available: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow-with-pkce>. [Último acceso: 10 Mayo 2024].
- [*] Okta, «OAuth 2.0 Authorization Code Flow,» [En línea]. Available: <https://www.oauth.com/playground/authorization-code.html>. [Último acceso: 10 Mayo 2024].
- [*] Microsoft, «What is Microsoft Entra Privileged Identity Management?,» 27 Octubre 2023. [En línea]. Available: <https://learn.microsoft.com/en-us/entra/id-governance/privileged-identity-management/pim-configure>. [Último acceso: 27 Abril 2024].
- [*] Microsoft, «Define identity licensing,» [En línea]. Available: <https://learn.microsoft.com/en-us/training/modules/explore-identity-azure-active-directory/11-define-identity-licensing>. [Último acceso: 8 Mayo 2024].

Referencias

- [1] Microsoft, «Shared responsibility in the cloud,» 29 Septiembre 2023. [En línea]. Available: <https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility>. [Último acceso: 10 Abril 2024].
- [2] F. Richter, «Amazon Maintains Cloud Lead as Microsoft Edges Closer,» Statista, 2 Mayo 2024. [En línea]. Available: <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>. [Último acceso: 24 Mayo 2024].
- [3] Microsoft, «Active Directory Domain Services Overview,» 17 Agosto 2022. [En línea]. Available: <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview>. [Último acceso: 25 Abril 2024].
- [4] Microsoft, «What is Conditional Access?,» 19 Marzo 2024. [En línea]. Available: <https://learn.microsoft.com/en-us/entra/identity/conditional-access/overview>. [Último acceso: 22 Mayo 2024].
- [5] Okta, «Refresh Tokens,» [En línea]. Available: <https://www.oauth.com/oauth2-servers/making-authenticated-requests/refreshing-an-access-token/>. [Último acceso: 10 Mayo 2024].
- [6] Okta, «The Authorization Request,» [En línea]. Available: <https://www.oauth.com/oauth2-servers/authorization/the-authorization-request/>. [Último acceso: 10 Mayo 2024].
- [7] IETF, «The OAuth 2.0 Authorization Framework,» Octubre 2012. [En línea]. Available: <https://www.rfc-editor.org/rfc/rfc6749.html>. [Último acceso: 2 Mayo 2024].

Anexos

Anexo I – Glosario de términos

Cloud: Entorno informático basado en la entrega de servicios de computación, almacenamiento y aplicaciones a través de internet, permitiendo el acceso remoto a recursos y la escalabilidad dinámica.

On-premise: Infraestructura y recursos de TI ubicados físicamente dentro de las instalaciones de una organización, gestionados y mantenidos internamente.

Azure: Plataforma de servicios en la nube de Microsoft que proporciona una amplia gama de servicios para computación, almacenamiento, bases de datos, redes y análisis.

Identity Access Management (IAM): Conjunto de políticas y tecnologías para asegurar y gestionar identidades y accesos en una organización.

Identity Provider (IdP): Entidad que proporciona autenticación y gestión de identidades, permitiendo a los usuarios acceder a múltiples servicios con una única identidad.

Active Directory (AD): Servicio de directorio de Microsoft para gestionar identidades y recursos en una red de Windows.

Microsoft Entra ID: Plataforma de gestión de identidades y accesos en la nube de Microsoft, anteriormente conocida como Azure Active Directory.

Tenant: Instancia dedicada de Entra ID que una organización utiliza para gestionar sus identidades y accesos.

Enterprise Applications: Aplicaciones configuradas en Entra ID para la autenticación y autorización, incluyendo aplicaciones SaaS y aplicaciones desarrolladas internamente.

App Registrations: Registro formal de una aplicación en Entra ID, configurando su acceso, permisos y otros aspectos necesarios para su integración con Entra ID.

Service Principal: Identidad creada en Entra ID para una aplicación, permitiendo que interactúe con otros recursos y servicios dentro del directorio.

Named location: Ubicación definida en base a direcciones IPs que se utilizan en políticas de acceso para prohibir o permitir el acceso.

Autenticación Multifactor (MFA): Método de autenticación que requiere más de una forma de verificación para probar la identidad de un usuario.

Privileged Identity Management (PIM): Herramienta que ayuda a gestionar, controlar y supervisar el acceso a roles privilegiados en Microsoft Entra ID.

Risk Detections: Funcionalidad en Microsoft Entra ID que identifica y notifica sobre actividades de inicio de sesión sospechosas.

Conditional Access: Herramienta de seguridad que permite establecer políticas para controlar el acceso a los recursos en función de condiciones específicas

Zero Trust: Modelo de seguridad que no confía implícitamente en ninguna entidad, verificando continuamente la identidad y contexto de cada acceso a recursos.

Autenticación: Proceso de verificar la identidad de un usuario o sistema, asegura que la entidad que intenta acceder a un recurso es quien dice ser.

Autorización: Proceso posterior a la autenticación para determinar si un usuario o sistema tiene permiso para acceder a un recurso o realizar una acción específica.

OAuth 2.0: Protocolo de autorización que permite a las aplicaciones obtener acceso limitado a los recursos de usuario en un servidor.

OpenID Connect (OIDC): Protocolo de autenticación basado en OAuth 2.0 que permite la verificación de identidades basándose en la autenticación realizada por un servidor.

Cliente: Es la aplicación que interactúa con el usuario y con Entra ID.

Navegador (Frontend): Programa que permite a los usuarios acceder y visualizar contenido web, interpretando y ejecutando código HTML, CSS y JavaScript.

Backend: Es el servidor de la aplicación donde se procesan las solicitudes.

Authorization Code Flow: Flujo de OAuth 2.0 que permite a las aplicaciones obtener tokens de acceso mediante un código de autorización intercambiado por un token.

PKCE (Proof Key for Code Exchange): Extensión de OAuth 2.0 que mejora la seguridad del Authorization Code Flow al mitigar ataques de interceptación de códigos.

Client Credentials Flow: Flujo de OAuth 2.0 en el cual una aplicación obtiene un token de acceso usando sus propias credenciales sin intervención del usuario.

Implicit Flow: Flujo de OAuth 2.0 en el que las aplicaciones reciben tokens directamente en el navegador sin un código de autorización intermedio, para aplicaciones de una sola página.

Resource Owner Password Flow: Flujo de OAuth 2.0 en el cual una aplicación gestiona directamente las credenciales del usuario para obtener los token del servidor de autorización.

Device Authorization Flow: Flujo de OAuth 2.0 diseñado para dispositivos que no tienen una interfaz de usuario, como dispositivos IoT, para obtener tokens de acceso.

URL: Dirección específica que se utiliza para acceder a recursos en Internet indicando la ubicación de un recurso y el protocolo para recuperarlo.

HTML: Lenguaje de marcado estándar para crear y diseñar páginas web definiendo la estructura y el contenido de una página mediante etiquetas y atributos.

Script: Conjunto de instrucciones de programación que se ejecutan automáticamente dentro de un entorno, como una página web o una aplicación

Python: Lenguaje de programación interpretado y de alto nivel conocido por su legibilidad y simplicidad.

Flask: Microframework de desarrollo web en Python que permite crear aplicaciones web rápidamente.

Microsoft Graph: API para acceder a datos de Microsoft 365 y otros servicios de Microsoft, permitiendo la integración y manipulación de estos datos en aplicaciones.

Client secret: Clave utilizada por aplicaciones para autenticar su identidad con el proveedor de servicios y solicitar tokens de acceso de manera segura.

Wireshark: Herramienta de análisis de red que permite capturar y examinar los datos en tiempo real, utilizada para diagnosticar problemas y detectar intrusiones en la red.

Loopback: Interfaz de red virtual que los sistemas utilizan para enviar y recibir datos a sí mismos. La dirección IP 127.0.0.1 es comúnmente utilizada para pruebas de red locales.

Man-in-the-Middle (MitM): Tipo de ataque donde un tercero intercepta y altera la comunicación entre dos partes sin su conocimiento. El atacante se posiciona entre el emisor y el receptor, pudiendo robar datos o inyectar información maliciosa.

JSON Web Token (JWT): Un estándar abierto de token compacto y autocontenido utilizado para transmitir información de forma segura entre dos partes como un objeto JSON.

Anexo II – Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.				X
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.	X			
ODS 9. Industria, innovación e infraestructuras.		X		
ODS 10. Reducción de las desigualdades.				X
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.			X	
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.		X		
ODS 17. Alianzas para lograr objetivos.				X

ODS 8. Trabajo decente y crecimiento económico: Mejorar la seguridad en las aplicaciones web permite aumentar la productividad en el entorno laboral, promoviendo así el crecimiento económico.

ODS 9. Industria, innovación e infraestructuras: La migración de organizaciones a una gestión de identidades en entornos cloud mejora la eficiencia operativa, fomenta la innovación y contribuye al desarrollo de infraestructuras modernas y resilientes.

ODS 13. Acción por el clima: Una de las ventajas de los entornos cloud es la optimización de recursos que permite reducir el consumo energético y, por lo tanto, logra una reducción de la huella de carbono, ayudando en la lucha contra el cambio climático.

ODS 16. Paz, justicia e instituciones sólidas: Al securizar las aplicaciones web y crear un entorno de gestión de identidades sólido y seguro, se fortalecen las instituciones, promoviendo justicia y confianza en el ámbito tecnológico.