Universidad Zaragoza
1542

Trabajo Fin de Grado

Ciberseguridad en entornos Cloud: estrategias de protección y transferencia segura de datos

Cybersecurity in Cloud Environments: Protection Strategies and Secure Data Transfer

Autor

Jesús Gabriel García Salomón

Director

Eduardo Peris Millán.

Escuela Universitaria Politécnica La Almunia

Junio 2025

Página intencionadamente en blanco.

Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Ciberseguridad en entornos Cloud: estrategias de protección y transferencia segura de datos

Cybersecurity in Cloud Environments: Protection Strategies and Secure Data Transfer

625.25.51

Autor: Jesús Gabriel García Salomón

Director: Eduardo Peris Millán

Fecha: 04 2025

Página intencionadamente en blanco.

INDICE de contenido Breve

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. RESUMEN

Con el presente Trabajo de Fin de Grado se tiene como principal objetivo el desarrollo del análisis sobre las vulnerabilidades más comunes presentes en entornos cloud debido a una mala configuración de los servicios que estas ofrecen, en esta ocasión estara centrado en la plataforma Amazon Web Services (AWS).Los servicios en la nube ofrecen numerosas ventajas, como son la escalabilidad y la accesibilidad de los datos, pero a su vez abre las puertas a posibles nuevas brechas de seguridad que pueden ser aprovechadas por atacantes si no se configuran adecuadamente estos recursos.

Durante el desarrollo del proyecto se han empleado distintos servicios como EC2, S3, RDS, DynamoDB y VPC para simular una infraestructura real. A través de pruebas controladas y simuladas se han reproducido escenarios con configuraciones deficientes que exponen datos sensibles o permiten accesos no autorizados. Para el análisis de estas vulnerabilidades se han utilizado herramientas como Wireshark, SQLMap, Nmap y HashCat, con estas herramientas se ha podido demostrar diferentes amenazas como son la interceptación de datos, ataques de inyección SQL o la recuperación de contraseñas débiles.

Con los resultados obtenidos en estas pruebas se pondrán de manifiesto la facilidad con la que un atacante puede explotar una mala práctica de configuración de un servicio en la nube, y con ello comprometer tanto la integridad como la confidencialidad de los datos.

Asimismo, se procederá a detallar estrategias necesarias para para para poder mitigar estos riesgos, como es el cifrado de datos tanto en tránsito como en reposo, el uso de políticas de acceso adecuadas, y la monitorización continua de la infraestructura.

En conclusión, el presente Trabajo de Fin de Grado destaca la importancia de una correcta configuración y gestión de los servicios cloud como factor clave para garantizar la seguridad en entornos virtuales, y proporciona un conjunto de buenas prácticas y recomendaciones aplicables en entornos reales.

1.1. PALABRAS CLAVE

Como principales palabras claves de este Trabajo de Fin de Grado se destacan las siguientes:

1. Ciberseguridad
2. Computación en la nube
3. Amazon Web Services (AWS)
4. Configuración segura
5. Transferencia de datos

2. ABSTRACT

This Final Degree Project aims to analyze the most common vulnerabilities that can arise in cloud environments due to misconfigurations of services, with a specific focus on the Amazon Web Services (AWS) platform. While the cloud offers numerous advantages, such as scalability and accessibility, it also introduces new attack vectors if resources are not properly configured.

Throughout the project, various AWS services such as EC2, S3, RDS, DynamoDB, and VPC were used to simulate a real infrastructure. Controlled tests were conducted to recreate scenarios with poor configurations that expose sensitive data or allow unauthorized access. Tools such as Wireshark, SQLMap, Nmap, and Hashcat were used to analyze these vulnerabilities, demonstrating attacks such as data interception, SQL injection, and brute-force password recovery.

The results show how easily an attacker can exploit cloud misconfigurations to compromise data confidentiality and integrity. Additionally, the study outlines the necessary strategies to mitigate these risks, including data encryption (in transit and at rest), proper access control policies, and continuous infrastructure monitoring.

In conclusion, this work emphasizes the importance of proper configuration and management of cloud services as a key factor in ensuring security in virtual environments. It also provides a set of best practices and recommendations that can be applied in real-world scenarios.

2.1. KEY WORDS

The main keywords of this Final Degree Project are as follows:

1. Cybersecurity
2. Cloud computing
3. Amazon Web Services (AWS)
4. Secure configuration
5. Data transfer

3. INTRODUCCIÓN

Con la creciente transformación digital que se ha visto en los últimos años ha llevado a un crecimiento exponencial en el uso de los servicios en la nube por parte de empresas y organizaciones de todos los tamaños. Plataformas como Amazon Web Services (AWS) ofrecen soluciones escalables, flexibles y eficientes, pero esto también conlleva a la introducción de nuevos desafíos en materia de la ciberseguridad. Una configuración incorrecta o una gestión inadecuada de los recursos cloud puede llegar a derivar en vulnerabilidades críticas en todo un sistema, exponiendo datos sensibles y comprometiendo la integridad del sistema.

La elección de este tema surge de mi interés personal por el campo de la ciberseguridad y, en particular, por cómo se aplican sus principios en entornos de computación en la nube. La motivación principal ha sido profundizar en los riesgos asociados a configuraciones inseguras en AWS, demostrar de forma práctica cómo pueden llegar a ser aprovechadas estas vulnerabilidades mediante herramientas ampliamente utilizadas en el ámbito de la seguridad informática, y presentar estrategias efectivas para mitigar estos riesgos. Se pretende así no solo poner en evidencia la importancia de una buena configuración y gestión de los servicios cloud, sino también ofrecer una guía práctica de protección y transferencia segura de datos.

El objetivo principal de este Trabajo de Fin de Grado es, por tanto, analizar las principales vulnerabilidades en entornos cloud originadas por configuraciones deficientes, demostrar sus posibles consecuencias a través de simulaciones controladas, y proponer un conjunto de buenas prácticas aplicables a infraestructuras en AWS

El trabajo se ha estructurado en varias secciones. En primer lugar, se presenta un marco teórico que contextualiza los conceptos clave relacionados con la computación en la nube y la ciberseguridad.

4. DESARROLLO

4.1. DESCRIPCIÓN GENERAL DEL PROYECTO

4.1.1. Objetivo general del desarrollo

En el siguiente Trabajo Final de Grado se procederá a desarrollar diferentes pruebas de pentesting para así poder evaluar las posibles brechas de seguridad que pueden surgir en servicios Cloud, para ello se desarrollará una aplicación web integral que permita realizar auditorías de seguridad automatizadas sobre los servicios de Amazon Web Services (AWS), en este proyecto nos centraremos específicamente en RDS, DynamoDB, EC2 y S3. Esta aplicación está diseñada de manera que podamos identificar vulnerabilidades comunes, configuraciones inseguras realizadas en la creación de los servicios y brechas de seguridad que podrían llegar a comprometer la integridad, confidencialidad y disponibilidad de datos sensibles que se almacenan en estos servicios.

Con este sistema se buscará proporcionar a administradores de sistemas y profesionales de ciberseguridad una herramienta práctica y eficiente para evaluar la seguridad de su infraestructura cloud, con la redacción de informes detallados con las recomendaciones específicas en cada caso para poder mitigar las vulnerabilidades detectadas.

4.1.2. Alcance y funcionalidades

En la aplicación desarrollada se abarcará las siguientes funcionalidades:

4.1.2.1. Auditoria de servicios AWS

- Amazon RDS

  Se realizara un analasis y evaluación de la configuracion de bases de datos, grupos de seguridad, cifrado de la información, deteccion de instancias expuestas públicamente,etc.

- Amazon DynamoDB

  Revisión y analisis de políticas de acceso IAM, configuraciones de sistema de cifrado, análisis de endpoints.

- Amazon S3

  Análisis y detección de buckets públicos, analisis de las politicas de acceso a los buckets, verificacion de configuraciones de cifrado, y auditoria logging.
- Amazon EC2

  Analisis de las configuraciones de instancias, grupos de seguridad, cifrado de volumnes EBS, gestion de claves SSH y deteccion de instancias expuestas públicamente.

4.1.3. Detección de vulnerabilidades

- Se identificaran configuraciones realizadas, por defecto, inseguras.
- Detección de los servicios cloud expuestos innecesariamente a internet.
- Análisis de políticas de acceso demasiado permisivas.
- Verificación de implementación de cifrado y protocolos seguros.
- Evaluación del cumplimiento de buenas practicas relacionadas con el ámbito de la seguridad de la información.

4.1.4. Tecnologías y herramientas empleadas

4.1.4.1. Framework de desarrollo Web

- Flask: es un framework web ligero escrito en Python, seleccionado para este proyecto debido a su ligereza, flexibilidad y facilidad de integración con librerías de terceros. Su diseño modular y su enfoque "micro" lo hacen ideal para construir aplicaciones escalables, permitiendo un flujo eficiente de datos entre la plataforma web y los servicios en la nube (como AWS mediante Boto3)

4.1.4.2. Integración con servicios AWS.

- Boto3: es el Software Development Kit (SDK) oficial de AWS para Python, que permite a los desarrolladores interactuar de manera programática con los servicios de AWS. Proporciona una interfaz intuitiva y segura para gestionar recursos en la nube, facilitando operaciones como el almacenamiento en Amazon S3, el procesamiento con Lambda o el manejo de bases de datos en DynamoDB. En esencia, actúa como un puente entre las

aplicaciones locales y los servicios en la nube de AWS, automatizando y optimizando el flujo de datos.

4.1.4.3. Lenguajes de programación

- Python: es el lenguaje principal del proyecto, seleccionado por su robusto ecosistema de librerías especializadas en ciberseguridad, su sintaxis intuitiva y su capacidad nativa para la automatización de tareas. Estas características lo convierten en una opción ideal para integrar componentes de seguridad, procesamiento de datos e interacción con servicios cloud.
- HTML y CSS: son los estándares empleados para el desarrollo de la interfaz web, garantizando una presentación estructurada y accesible de los resultados. Su uso combinado permite implementar diseños responsive que se adaptan a distintos dispositivos, facilitando la visualización de datos procesados por el backend Python.

4.1.4.4. Herramientas de pentesting integradas

- SQLMap: es un herramienta de penetration testing especializada en la detección y explotación de vulnerabilidades de inyección SQL en bases de datos relacionales.
- Nmap: es un escáner de red de código abierto empleado para el descubrimiento de activos en la infraestructura AWS, fingerprinting de servicios mediante detección de puertos abiertos, y enumeración de versiones de software. Su capacidad para generar mapas de red contribuye al análisis de superficie de ataque.
- Hashcat: es un software de recuperación de contraseñas que permite realizar ataques de fuerza bruta y diccionario para evaluar la fortaleza de credenciales.
- Wireshark: es un analizador de protocolos de red con capacidades de deep packet inspection, permite capturar y analizar tráfico en tiempo real entre servicios AWS. Su uso facilita la detección de anomalías según modelos OSI y el diagnóstico de configuraciones inseguras (TLS/SSL, DNS hijacking).

4.2. ARQUITECTURA DEL SISTEMA

4.2.1. Diseño de entorno de pruebas

El entorno de pruebas implementa una arquitectura web de tres capas desplegada en AWS, compuesta por una aplicación Flask como frontend, servicios de base de datos RDS PostgreSQL y componentes de autenticación y autorización gestionados por IAM.

Esta arquitectura ha sido diseñada deliberadamente con vulnerabilidades conocidas de seguridad para facilitar la evaluación de riesgos en entornos cloud y la validación de técnicas de pentesting específicas para infraestructuras distribuidas. El diseño permite simular escenarios de ataque reales contra aplicaciones web modernas desplegadas en servicios de nube pública, proporcionando un entorno controlado para el análisis de vulnerabilidades y la implementación de medidas de seguridad.

4.2.2. Configuración de Amazon RDS

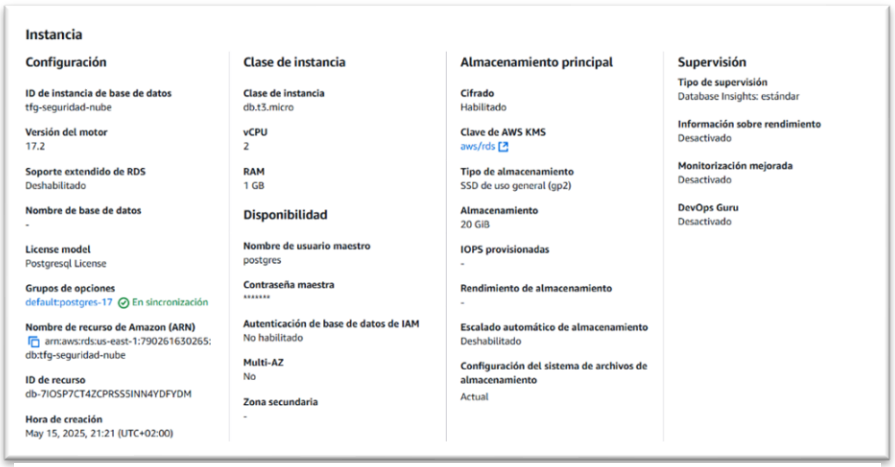


Ilustración 1

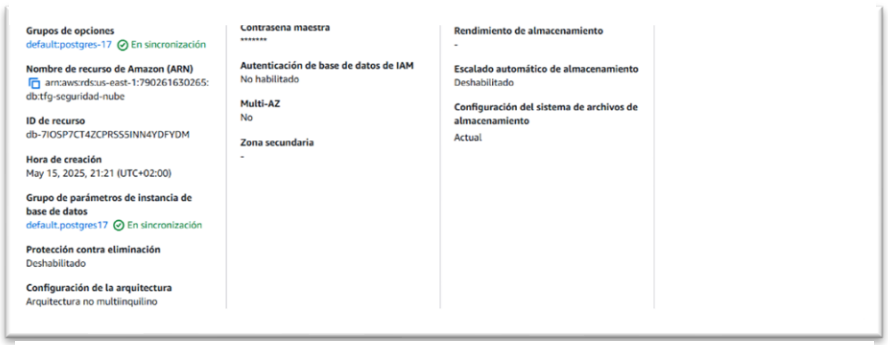


Ilustración 2

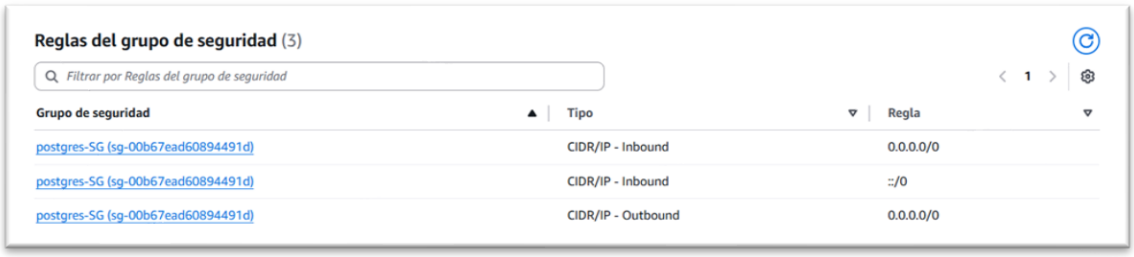
La instancia RDS esta creada para bases de datos PostgreSQL 17.2 configurada, presenta múltiples vulnerabilidades de seguridad críticas. El

identificador "tfg-seguridad-nube" utiliza una clase de instancia db.t3.micro con recursos limitados (2 vCPU, 1 GB RAM) que puede comprometer la disponibilidad del servicio. La configuración de seguridad es especialmente preocupante ya que la autenticación IAM está deshabilitada, forzando el uso exclusivo de credenciales de base de datos tradicionales que son más vulnerables a ataques de fuerza bruta y compromiso de credenciales. La ausencia de Multi-AZ elimina la redundancia geográfica, creando un punto único de fallo que puede resultar en pérdida total de disponibilidad ante fallos de zona. El sistema de copias de seguridad automáticas está completamente deshabilitado, lo que significa que no existe recuperación automática ante corrupción de datos o ataques ransomware. La protección contra eliminación deshabilitada permite que la base de datos sea eliminada accidentalmente o de forma maliciosa sin controles adicionales. La monitorización está completamente desactivada (Performance Insights, monitorización mejorada y DevOps Guru deshabilitados), eliminando la capacidad de detectar actividad sospechosa, consultas maliciosas o patrones de acceso anómalos. Aunque el cifrado está habilitado con AWS KMS, la gestión de claves centralizada puede representar un riesgo si las políticas de acceso no están correctamente configuradas. La configuración actual expone la base de datos a múltiples vectores de ataque incluyendo acceso no autorizado, pérdida de datos, ataques de denegación de servicio y compromiso de integridad sin capacidad de detección o recuperación efectiva.

Ilustración 3

La configuración de conectividad y seguridad de la instancia RDS presenta vulnerabilidades significativas. El punto de enlace público "tfg-seguridad-nube.co7kgm28ubvu.us-east-1.rds.amazonaws.com" en el puerto 5432 expone la base de datos directamente a Internet, permitiendo que cualquier atacante pueda intentar conexiones desde cualquier ubicación global. La configuración "Accesible públicamente:

Sí elimina cualquier barrera de red inicial, convirtiendo la base de datos en un objetivo directo para ataques de fuerza bruta y escaneo de puertos. Aunque existe un grupo de seguridad "postgres-SG" activo, su configuración específica determinará el nivel real de protección. La distribución en múltiples subredes (subnet-044c476235e1da43b, subnet-001b8aeda5c640204, etc.) dentro de la VPC vpc-09fae5da7659def64 sugiere cierta segmentación de red, pero la accesibilidad pública anula estos controles. El certificado de entidad rds-ca-rsa2048-g1 con vencimiento en mayo de 2026 proporciona cifrado en tránsito, pero esto no mitiga el riesgo fundamental de exposición pública. Esta configuración viola principios básicos de defensa en profundidad al eliminar la protección perimetral de red, exponiendo la base de datos a ataques directos desde Internet sin requerir acceso previo a la infraestructura interna.



El grupo de seguridad "postgres-SG" tiene reglas extremadamente permisivas y peligrosas. Las reglas de entrada (Inbound) con CIDR "0.0.0.0/0" y "::/0" permiten conexiones desde cualquier dirección IP en Internet tanto para IPv4 como IPv6, eliminando completamente cualquier restricción de origen. La regla de salida (Outbound) "0.0.0.0/0" permite que la base de datos establezca conexiones hacia cualquier destino en Internet. Esta configuración es equivalente a no tener firewall, ya que acepta todo el tráfico entrante y saliente sin restricciones. Cualquier atacante puede intentar conectarse directamente a la base de datos desde cualquier ubicación mundial, y si logra comprometerla, puede usar la instancia para establecer conexiones hacia otros sistemas o exfiltrar datos sin limitaciones de red.

Una vez iniciada la instancia de Amazon RDS, se procede a conectarla con el motor de pgAdmin para la gestión de la base de datos PostgreSQL. Este proceso requiere el establecimiento de una conexión remota utilizando las credenciales de acceso correspondientes que, por

motivos de seguridad y privacidad, no pueden ser mostradas en el presente documento.

Una vez que accedemos a la base de datos creada, consultamos las tablas disponibles, en este caso la principal y la que sera objeto de estudio sera la de "users":

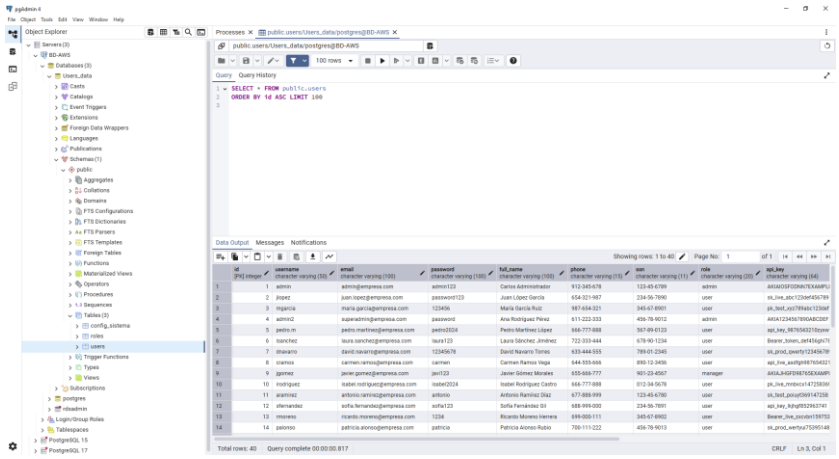


Ilustración 4. Tablas BBDD

4.2.3. Configuración de Amazon DynamoDB

Se ha creado una tabla login_logs de DynamoDB, está diseñada para registrar y auditar los intentos de autenticación en el sistema. Su configuración utiliza una clave de partición denominada username de tipo String, que actúa como identificador principal para distribuir los registros a través de las particiones de la base de datos, y una clave de ordenación llamada timestamp también de tipo String, que permite organizar cronológicamente los registros de cada usuario específico.

Esta estructura de clave compuesta facilita el monitoreo de seguridad y análisis forense, permitiendo rastrear patrones de acceso, detectar intentos de acceso no autorizados, y generar reportes de actividad por usuario. El diseño optimiza consultas como "obtener todos los logins de un usuario en un período específico" o "identificar intentos fallidos consecutivos", siendo fundamental para implementar medidas de seguridad como bloqueo de cuentas por intentos excesivos y análisis de comportamiento de usuarios en el sistema de autenticación.

Ilustración 5 Tabla login_logs

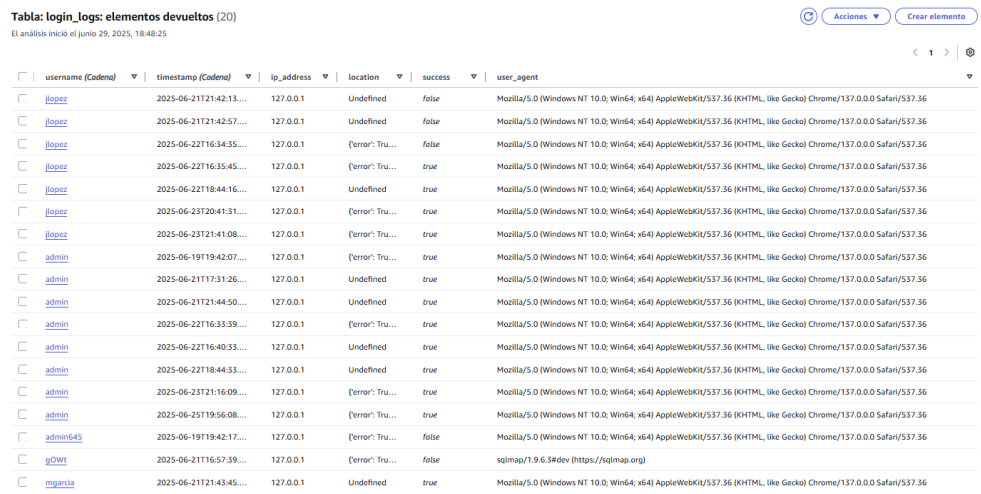


Tabla: login_logs: elementos devueltos (20)

El análisis inició el junio 29, 2025, 18:48:25

Acciones | Crear elemento

username (Cadena)	timestamp (Cadena)	ip_address	location	success	user_agent
jlopez	2025-06-21T21:42:13....	127.0.0.1	Undefined	false	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
jlopez	2025-06-21T21:42:57....	127.0.0.1	Undefined	false	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
jlopez	2025-06-22T16:34:35....	127.0.0.1	{'error': Tru...	false	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
jlopez	2025-06-22T16:35:45....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
jlopez	2025-06-22T18:44:16....	127.0.0.1	Undefined	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
jlopez	2025-06-23T20:41:31....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
jlopez	2025-06-23T21:41:08....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-19T19:42:07....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-21T17:31:26....	127.0.0.1	Undefined	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-21T21:44:50....	127.0.0.1	Undefined	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-22T16:33:39....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-22T16:40:33....	127.0.0.1	Undefined	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-22T18:44:33....	127.0.0.1	Undefined	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-23T21:16:09....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin	2025-06-25T19:56:08....	127.0.0.1	{'error': Tru...	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
admin645	2025-06-19T19:42:17....	127.0.0.1	{'error': Tru...	false	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
gOWt	2025-06-21T16:57:39....	127.0.0.1	{'error': Tru...	false	sqlmap/1.9.6.3#dev (https://sqlmap.org)
mgarcia	2025-06-21T21:43:45....	127.0.0.1	Undefined	true	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36

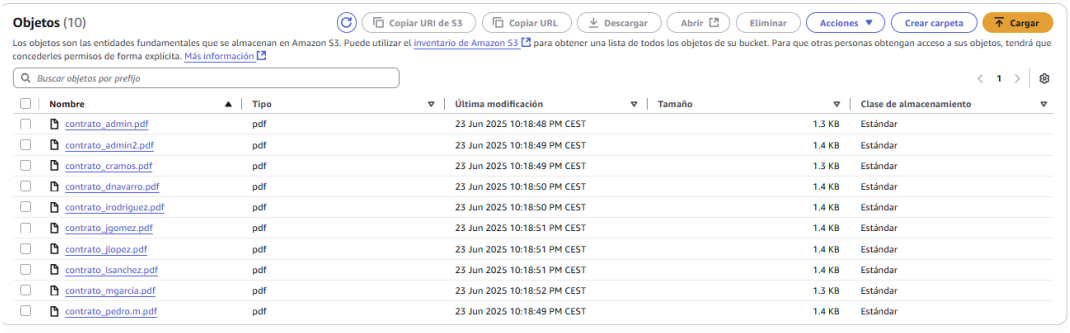
Ilustración 6 Tabla login_logs

4.2.4. Configuración de Bucket S3

Se ha creado un bucket de S3 llamado empresa-informes el cual posee la siguiente estructura de carpetas:

```
empresa-informes/
├── contratos/
├── dni/
├── informes/
└── nominas/
```

En estas carpetas se almacenarán diferentes tipos de ficheros de tipo pdf, xlsx y jpeg.



Objetos (10)

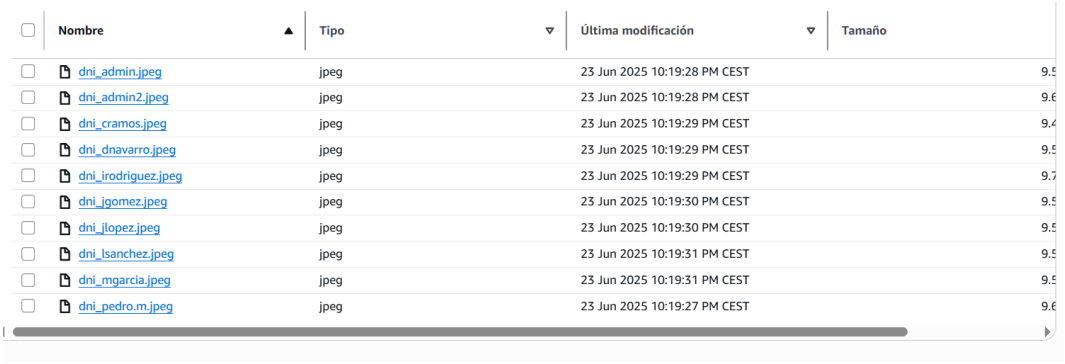
Copiar URI de S3 | Copiar URL | Descargar | Abrir | Eliminar | Acciones | Crear carpeta | Cargar

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el inventario de Amazon S3 para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. Más información

Buscar objetos por prefijo

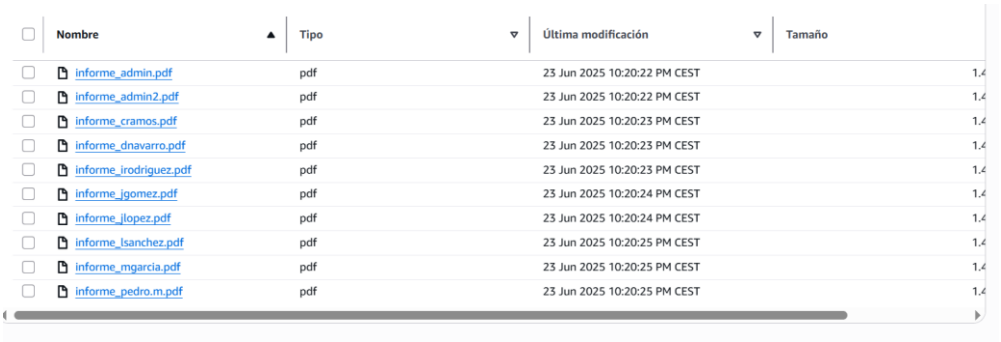
Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
contrato_admin.pdf	pdf	23 Jun 2025 10:18:48 PM CEST	1.3 KB	Estándar
contrato_admin2.pdf	pdf	23 Jun 2025 10:18:49 PM CEST	1.4 KB	Estándar
contrato_cramos.pdf	pdf	23 Jun 2025 10:18:49 PM CEST	1.3 KB	Estándar
contrato_dnavarro.pdf	pdf	23 Jun 2025 10:18:50 PM CEST	1.4 KB	Estándar
contrato_irodriguez.pdf	pdf	23 Jun 2025 10:18:50 PM CEST	1.4 KB	Estándar
contrato_jgomez.pdf	pdf	23 Jun 2025 10:18:51 PM CEST	1.4 KB	Estándar
contrato_jlopez.pdf	pdf	23 Jun 2025 10:18:51 PM CEST	1.4 KB	Estándar
contrato_lsanchez.pdf	pdf	23 Jun 2025 10:18:51 PM CEST	1.4 KB	Estándar
contrato_mgarcia.pdf	pdf	23 Jun 2025 10:18:52 PM CEST	1.3 KB	Estándar
contrato_pedro.m.pdf	pdf	23 Jun 2025 10:18:49 PM CEST	1.4 KB	Estándar

Ilustración 7. Contratos S3



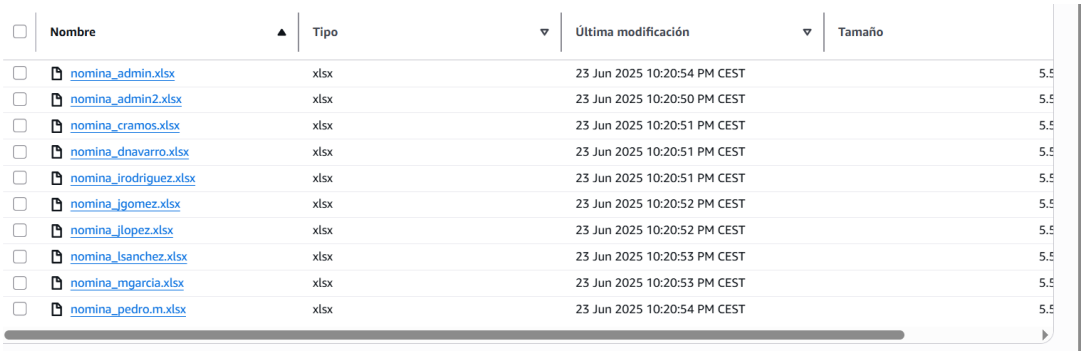
Nombre	Tipo	Última modificación	Tamaño
dni_admin.jpeg	jpeg	23 Jun 2025 10:19:28 PM CEST	9.5
dni_admin2.jpeg	jpeg	23 Jun 2025 10:19:28 PM CEST	9.6
dni_cramos.jpeg	jpeg	23 Jun 2025 10:19:29 PM CEST	9.4
dni_dnavarro.jpeg	jpeg	23 Jun 2025 10:19:29 PM CEST	9.5
dni_irodriguez.jpeg	jpeg	23 Jun 2025 10:19:29 PM CEST	9.7
dni_jgomez.jpeg	jpeg	23 Jun 2025 10:19:30 PM CEST	9.5
dni_jlopez.jpeg	jpeg	23 Jun 2025 10:19:30 PM CEST	9.5
dni_lsanchez.jpeg	jpeg	23 Jun 2025 10:19:31 PM CEST	9.5
dni_mgarcia.jpeg	jpeg	23 Jun 2025 10:19:31 PM CEST	9.5
dni_pedro.m.jpeg	jpeg	23 Jun 2025 10:19:27 PM CEST	9.6

Ilustración 8.Dni S3



Nombre	Tipo	Última modificación	Tamaño
informe_admin.pdf	pdf	23 Jun 2025 10:20:22 PM CEST	1.4
informe_admin2.pdf	pdf	23 Jun 2025 10:20:22 PM CEST	1.4
informe_cramos.pdf	pdf	23 Jun 2025 10:20:23 PM CEST	1.4
informe_dnavarro.pdf	pdf	23 Jun 2025 10:20:23 PM CEST	1.4
informe_irodriguez.pdf	pdf	23 Jun 2025 10:20:23 PM CEST	1.4
informe_jgomez.pdf	pdf	23 Jun 2025 10:20:24 PM CEST	1.4
informe_jlopez.pdf	pdf	23 Jun 2025 10:20:24 PM CEST	1.4
informe_lsanchez.pdf	pdf	23 Jun 2025 10:20:25 PM CEST	1.4
informe_mgarcia.pdf	pdf	23 Jun 2025 10:20:25 PM CEST	1.4
informe_pedro.m.pdf	pdf	23 Jun 2025 10:20:25 PM CEST	1.4

Ilustración 9. Informes S3



Nombre	Tipo	Última modificación	Tamaño
nomina_admin.xlsx	xlsx	23 Jun 2025 10:20:54 PM CEST	5.5
nomina_admin2.xlsx	xlsx	23 Jun 2025 10:20:50 PM CEST	5.5
nomina_cramos.xlsx	xlsx	23 Jun 2025 10:20:51 PM CEST	5.5
nomina_dnavarro.xlsx	xlsx	23 Jun 2025 10:20:51 PM CEST	5.5
nomina_irodriguez.xlsx	xlsx	23 Jun 2025 10:20:51 PM CEST	5.5
nomina_jgomez.xlsx	xlsx	23 Jun 2025 10:20:52 PM CEST	5.5
nomina_jlopez.xlsx	xlsx	23 Jun 2025 10:20:52 PM CEST	5.5
nomina_lsanchez.xlsx	xlsx	23 Jun 2025 10:20:53 PM CEST	5.5
nomina_mgarcia.xlsx	xlsx	23 Jun 2025 10:20:53 PM CEST	5.5
nomina_pedro.m.xlsx	xlsx	23 Jun 2025 10:20:54 PM CEST	5.5

Ilustración 10. Nominas S3

La configuración del bucket S3 empresa-informes tiene desactivado el bloqueo de acceso público, lo que permite el acceso directo a los objetos almacenados sin restricciones de autenticación. Esta configuración facilita el acceso a los documentos desde aplicaciones externas o usuarios sin credenciales AWS, siendo útil para casos donde se requiere compartir información de forma directa mediante URLs públicas.

Sin embargo, esta configuración presenta implicaciones de seguridad significativas, ya que cualquier persona con conocimiento de

las URLs puede acceder a los documentos almacenados. Para un bucket que contiene información sensible como contratos, DNIs y nóminas.

4.2.5. Configuración de instancia EC2

La instancia EC2 ha sido configurada con Ubuntu Server 22.04 LTS (AMI ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20250516) como sistema operativo, optando por una distribución Linux estable y ampliamente soportada para entornos de servidor. La instancia utiliza virtualización HVM con 1 CPU virtual y se ejecuta bajo la especificación de crédito estándar para optimizar el rendimiento según la carga de trabajo.

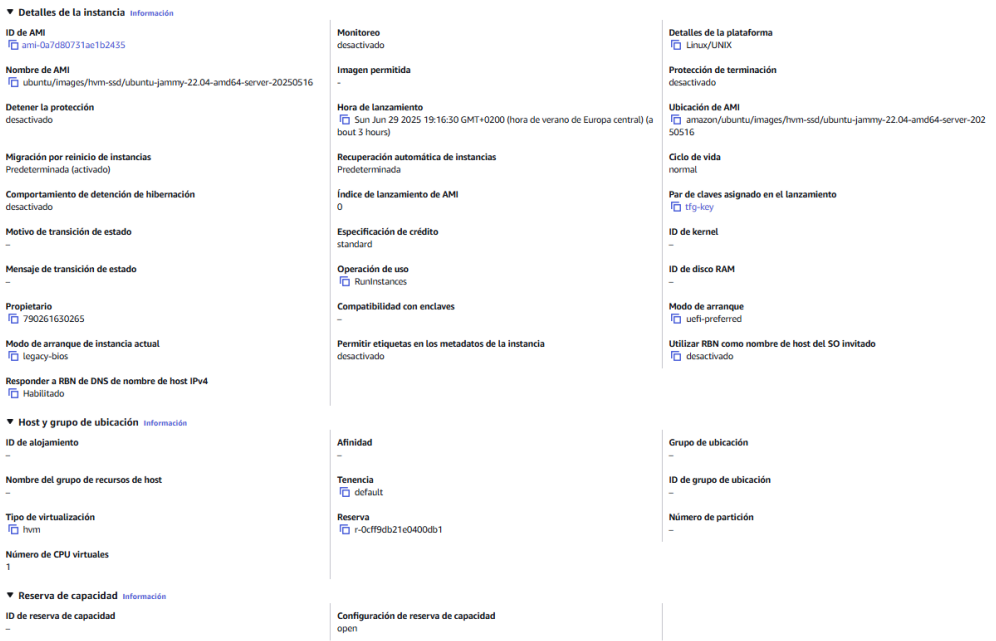


Ilustración 11.Configuracion Instancia EC2

En cuanto a la configuración de red, la instancia se ha desplegado en la zona de disponibilidad por defecto con modo de arranque UEFI-preferred para mayor compatibilidad. El grupo de seguridad launch-wizard-1 controla el acceso mediante reglas específicas: permite conexiones SSH (puerto 22), HTTP (puerto 80), HTTPS (puerto 443) y un puerto personalizado (8080) desde cualquier origen (0.0.0.0/0), mientras que las reglas de salida permiten todo el tráfico hacia cualquier destino.

Nombre	ID de la regla del grupo ...	Intervalo de p...	Protocolo	Origen	Grupos de seguridad
–	sgr-0f901c87edc0523d9	8080	TCP	0.0.0.0/0	launch-wizard-1
–	sgr-077bcdd1d76f657e8	22	TCP	0.0.0.0/0	launch-wizard-1
–	sgr-0badc00fab7f42f27	443	TCP	0.0.0.0/0	launch-wizard-1
–	sgr-0c284c3eb40ef71c2	80	TCP	0.0.0.0/0	launch-wizard-1

Ilustración 12. Configuración Instancia EC2

Durante el proceso de creación, AWS genera automáticamente un par de claves RSA que incluye una clave privada en formato .pem, esencial para establecer conexiones SSH seguras con la instancia. Esta clave debe mantenerse confidencial y es el único método de autenticación para acceder remotamente al servidor, siendo fundamental para la administración y gestión segura de la instancia.

4.2.6. Configuración claves de acceso IAM.

Ilustración 13. Clave pem

La gestión adecuada de las credenciales IAM es fundamental para el acceso seguro a los servicios de AWS como DynamoDB y S3.

Se ha creado los permisos de dos personas: app-logger-dyanmodb y app-logger-s3, con ellas se podrá hacerlos la transferencia y descarga de datos en cada servicio respectivo

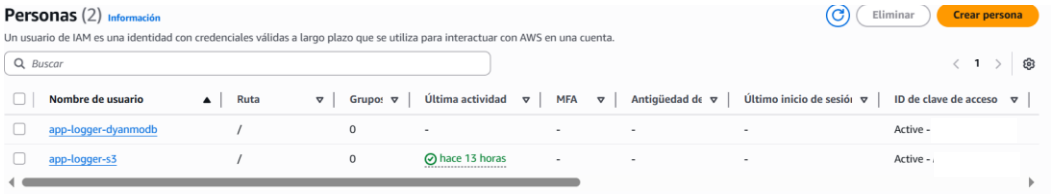


Ilustración 14.Usuarios IAM

4.2.6.1. Configuración de políticas de permisos para DynamoDB

Las políticas de permisos de DynamoDB determinan qué acciones puede realizar un usuario o aplicación sobre las tablas y recursos de DynamoDB. AWS proporciona varias políticas administradas que cubren los casos de uso más comunes, lo que facilita la configuración inicial sin necesidad de crear políticas personalizadas desde cero.

La política **AmazonDynamoDBFullAccess** proporciona acceso completo a todos los recursos de DynamoDB, incluyendo permisos para crear, leer, actualizar y eliminar tablas, así como operaciones de administración como backup y restore. Esta política es útil para entornos de desarrollo o para administradores que necesitan control total sobre el servicio, aunque debe evitarse en producción por razones de seguridad.

Para casos donde solo se requiere consultar información, **AmazonDynamoDBReadOnlyAccess** permite únicamente operaciones

de lectura sobre tablas existentes, incluyendo Query, Scan, GetItem y BatchGetItem, sin permitir modificaciones de datos ni estructura de tablas. Esta política es ideal para aplicaciones que únicamente consultan información almacenada.

Cuando se trabaja con funciones Lambda, **AWSLambdaDynamoDBExecutionRole** está diseñada específicamente para este propósito, combinando permisos básicos de DynamoDB con capacidades de logging en CloudWatch. Para configuraciones donde DynamoDB necesita invocar funciones Lambda, como en el caso de triggers y streams, se utiliza **AWSLambdaInvocation-DynamoDB.**

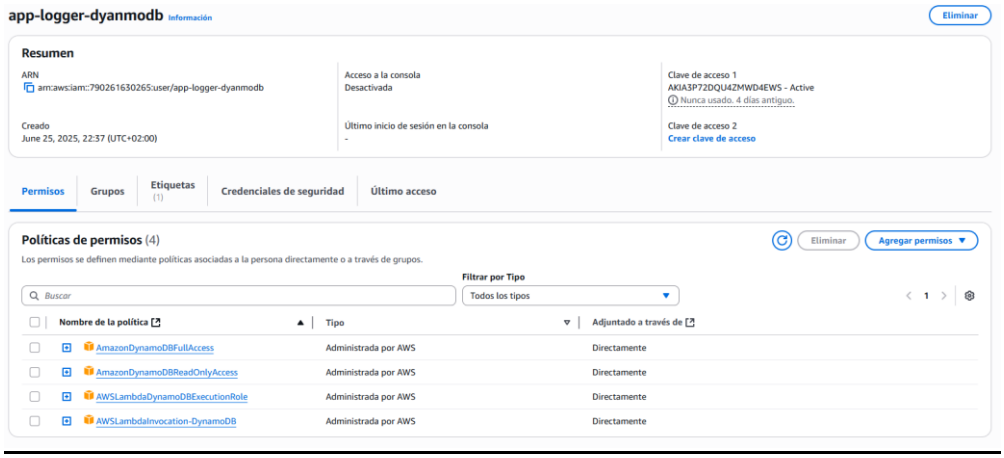


Ilustración 15. Politicas de permisos-Dynamodb

La asignación de estas políticas debe seguir el principio de menor privilegio, seleccionando siempre la política más restrictiva que cubra las necesidades específicas del caso de uso. Es importante verificar los permisos mediante pruebas controladas antes de desplegar en producción y documentar adecuadamente las asignaciones para facilitar futuras auditorías.

En entornos de producción se recomienda evitar el uso de políticas con acceso completo, optando por combinar múltiples políticas específicas cuando sea necesario. El monitoreo continuo del uso de permisos a través de CloudTrail permite identificar posibles problemas de seguridad y optimizar las políticas según los patrones de uso reales.

4.2.7. Configuración de políticas de permisos para S3

Las políticas de permisos de S3 controlan el acceso a los buckets y objetos almacenados en Amazon S3. En la imagen vemos que el usuario tiene asignada la política **AmazonS3ReadOnlyAccess**, que es una de las políticas administradas más utilizadas para acceso controlado a S3.

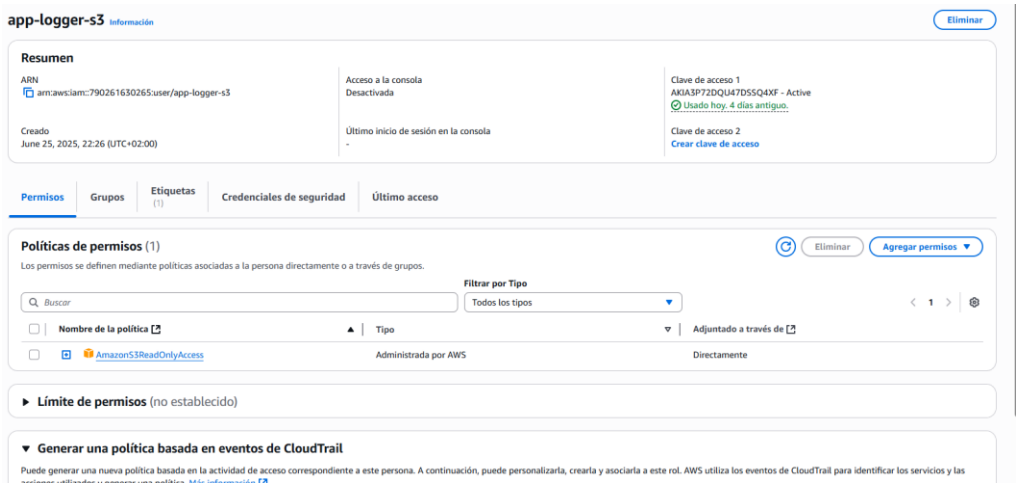


Ilustración 16. Políticas de acceso-S3

La política **AmazonS3ReadOnlyAccess** que aparece asignada al usuario proporciona permisos únicamente de lectura sobre todos los recursos de S3. Esta política permite al usuario listar buckets existentes, obtener objetos y sus metadatos, y descargar contenido, pero no realizar modificaciones, cargas o eliminaciones.

Esta configuración es ideal para aplicaciones que necesitan acceder a archivos almacenados sin riesgo de alterarlos, como sistemas de consulta de documentos, aplicaciones de reporting que acceden a datos históricos, o servicios que procesan información sin modificar el origen. Al ser una política de solo lectura, minimiza significativamente los riesgos de seguridad al prevenir modificaciones accidentales o maliciosas.

4.2.8. Configuración de credenciales en el entorno de desarrollo

Una vez obtenidas las claves de acceso IAM, es necesario configurar las credenciales en el entorno de desarrollo para que las aplicaciones puedan conectarse a los servicios de AWS. Boto3, el SDK de AWS para Python, ofrece varias formas de autenticación que permiten a las aplicaciones acceder de forma segura a servicios como DynamoDB y S3.

Boto3 facilita la conexión a los servicios de AWS mediante la creación de clientes específicos para cada servicio. Para establecer una conexión a S3, se utiliza el método boto3.client() especificando el servicio, la región y las credenciales de acceso. La configuración típica incluye el ID de clave de acceso (aws_access_key_id) y la clave secreta (aws_secret_access_key) obtenidas durante la creación del usuario IAM.

```python
import boto3
from botocore.exceptions import ClientError
import urllib.parse

# Parámetros de conexión (cámbialos por los tuyos)
s3 = boto3.client(
    's3',
    region_name='us-east-1',
    aws_access_key_id=[redacted],
    aws_secret_access_key=[redacted]
)

BUCKET_NAME = 'empresa-informes'
```

Ilustración 18. Conexión S3

```python
import boto3
from boto3.dynamodb.conditions import Key

dynamodb = boto3.resource(
    'dynamodb',
    region_name='us-east-1',
    aws_access_key_id='[redacted]',
    aws_secret_access_key='[redacted]'
)

# Conexión a tabla
logs_table = dynamodb.Table('login_logs')
```

Ilustración 17. Conexión Dynamodb

Por motivos de privacidad y seguridad no se pueden mostrar las credenciales de acceso en la memoria del proyecto.

4.3. DESARROLLO DE LA APLICACIÓN WEB VULNERABLE – CLOUDSECURE ERP

CloudSecure ERP es una aplicación web desarrollada específicamente para este estudio, que simula un sistema de gestión empresarial con vulnerabilidades intencionalmente implementadas para demostrar técnicas de explotación y análisis de seguridad en entornos cloud.

4.3.1. Arquitectura y diseño del sistema

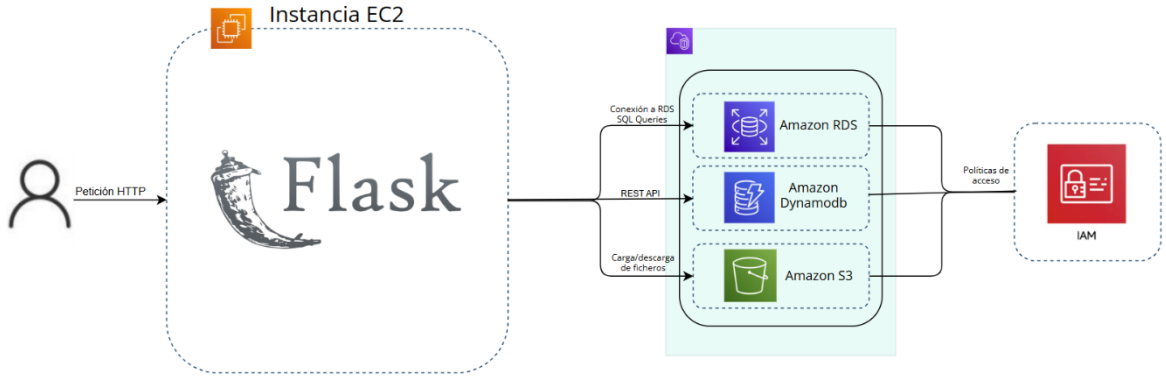


Ilustración 19.Arquitectura Aplicación web

Con este diagrama se muestra el funcionamiento de la aplicación web construida con Flask la cual se ejecuta en una instancia EC2 de AWS y utiliza varios servicios de Amazon para gestionar datos y seguridad.

4.3.1.1. Flujo de funcionamiento:

1. El usuario realiza una petición HTTP desde su navegador o aplicación hacia la aplicación Flask.
2. Flask recibe la petición y actúa como el núcleo de la aplicación, procesando las solicitudes y coordinando las respuestas.
3. Según el tipo de operación, Flask se comunica con diferentes servicios de AWS a través de su REST API.
4. IAM (Identity and Access Management) controla las políticas de acceso, asegurando que solo los usuarios y servicios autorizados puedan acceder a los recursos correspondientes.
5. Flask devuelve la respuesta al usuario con los datos solicitados, ya sean consultas de base de datos, archivos descargados o confirmaciones de operaciones realizadas.

Esta arquitectura aprovecha la escalabilidad y confiabilidad de los servicios gestionados de AWS mientras mantiene Flask como el controlador central de la lógica de aplicación.

4.3.2. Estructura del Proyecto

La aplicación se organizó siguiendo buenas prácticas de desarrollo, con una separación clara de responsabilidades y componentes modulares:

- db/: Módulos especializados para cada servicio de base de datos (DynamoDB, RDS, S3).
- static/css/: Archivos de estilos organizados por funcionalidad (admin, dashboard, login).
- templates/: Plantillas HTML para cada interfaz del sistema.
- uploads/: Directorio para gestión de archivos subidos.
- utils/: Utilidades y archivo principal de la aplicación.

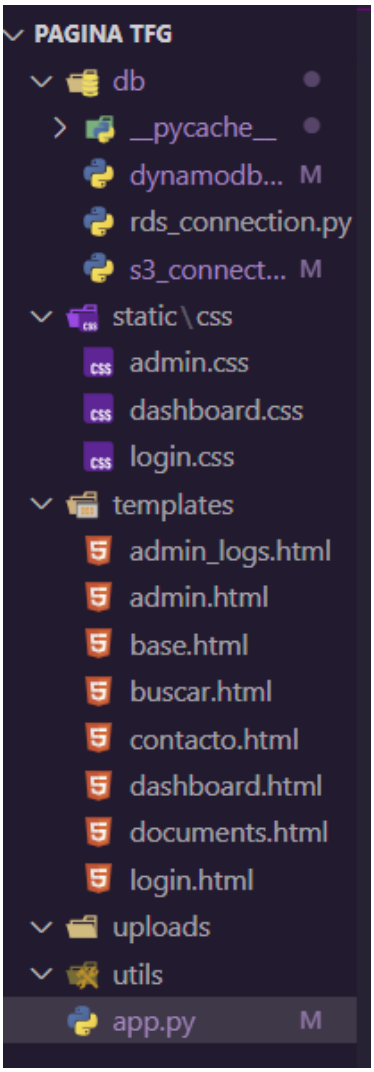


Ilustración 20. Estructura de archivos

4.3.3. Componentes Principales del Sistema

4.3.3.1. Módulos de Conexión a Bases de Datos:

- DynamoDB_connection.py: Gestiona operaciones CRUD con la base de datos NoSQL, optimizada para consultas rápidas y almacenamiento de logs de auditoría
- RDS_Connection.py: Maneja todas las operaciones SQL con la base de datos relacional PostgreSQL, incluyendo autenticación de usuarios y consultas de datos estructurados

- S3_Connection.py: Controla la gestión completa de archivos (subida, descarga, listado) con generación de URLs firmadas para acceso seguro.

4.3.3.2. Interfaces de Usuario

- Login: Sistema de autenticación con validación y registro automático de intentos.
- Dashboard: Panel personalizado para usuarios regulares con acceso a sus datos personales.
- Panel de Administración: Interfaz avanzada para administradores con funcionalidades de gestión y monitoreo.
- Gestión de Documentos: Sistema completo para visualizar y descargar archivos personales.

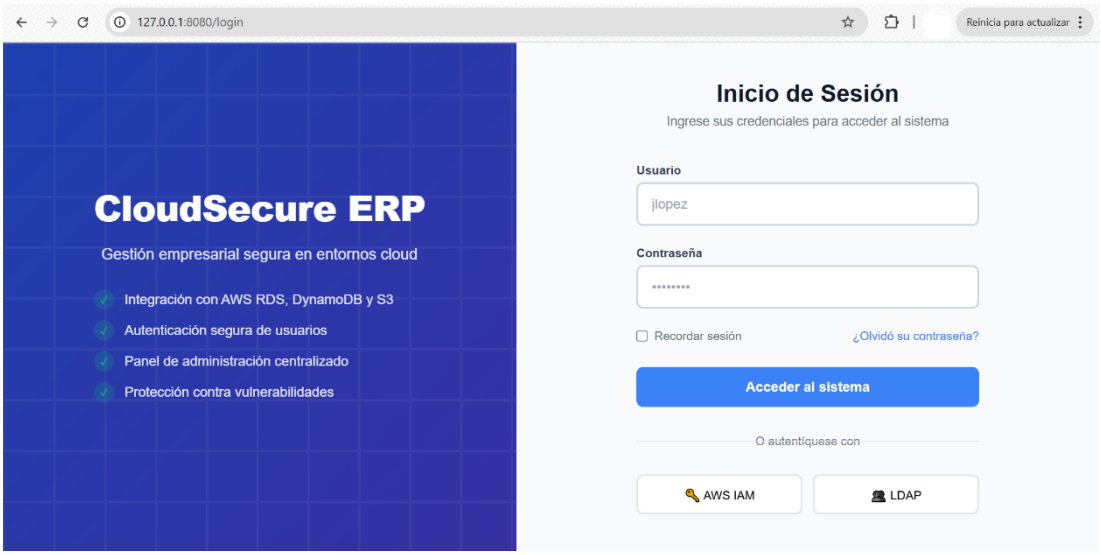


Ilustración 21. Página login

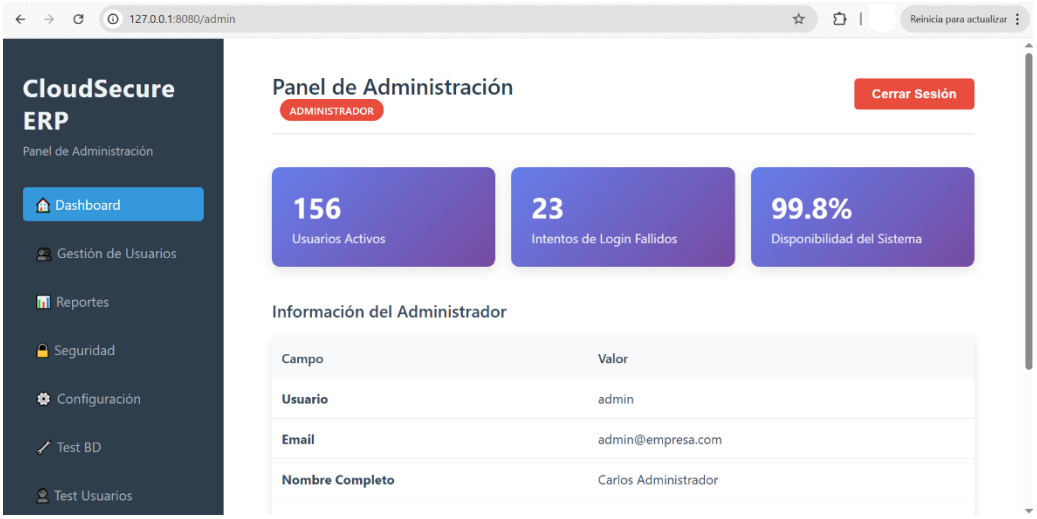


Ilustración 22. Panel de Administración

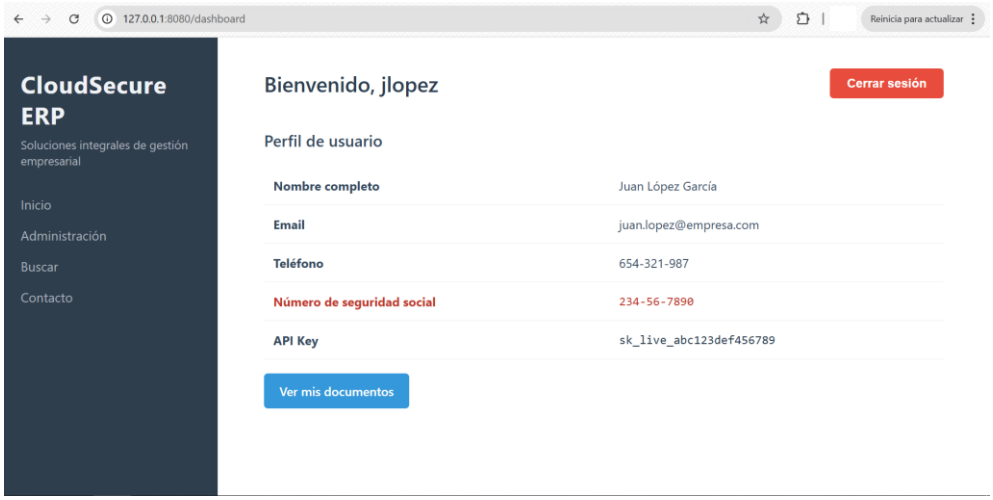


Ilustración 23. Menú Dashboard

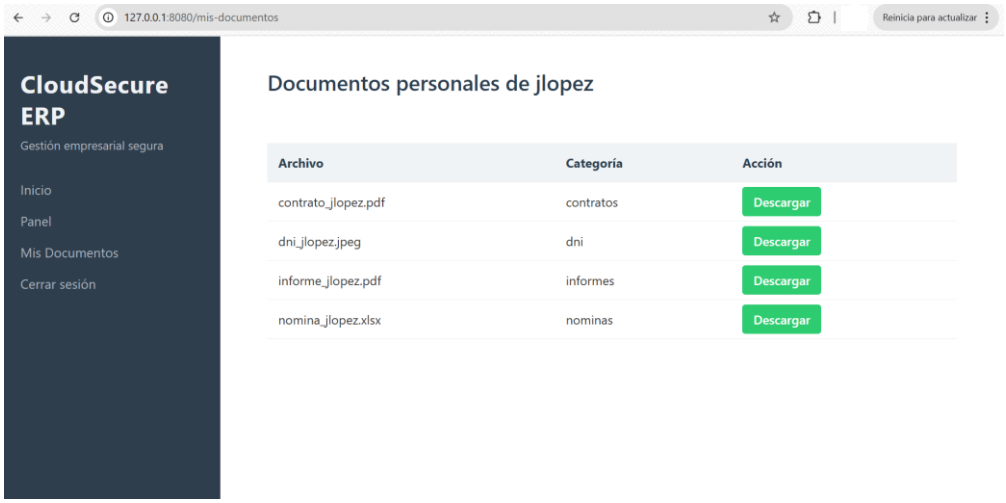


Ilustración 24. Descargar documentos

4.3.3.3. Despliegue en AWS EC2

4.3.3.3.1. Transferencia de Archivos con PSCP

Para poder transferir la aplicación a la instancia de EC2 se empleara protocolo PSCP el nos permite transferir archivos de forma segura entre el sistema local y la instancia de EC2.

Para usarlo, el comando necesita la llave de acceso en formato .ppk, esto se puede conseguir con el software de Puttygen que nos permite convertir nuestra llave original .pem en .ppk:

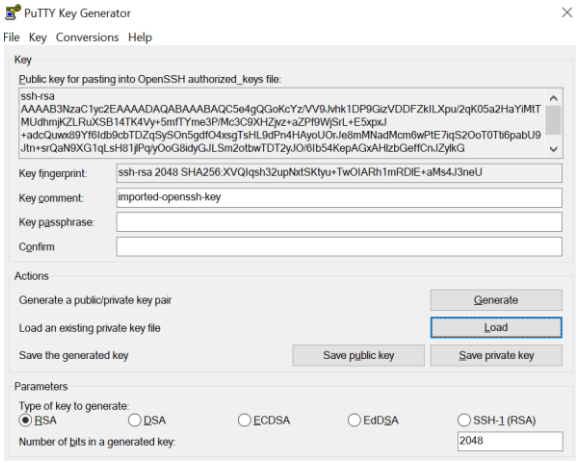


Ilustración 25. Clave ppk

Una vez obtenida la clave, transferiremos los archivos a la instancia con el siguiente comando:

```
pscp -r -i "C:\Users\gabau\OneDrive\Escritorio\Documentacion Proyecto TFG\tfg-key.ppk" "C:\Users\gabau\OneDrive\Escritorio\pagina TFG" ubuntu@54.152.149.75:/home/ubuntu/proyectos/
```

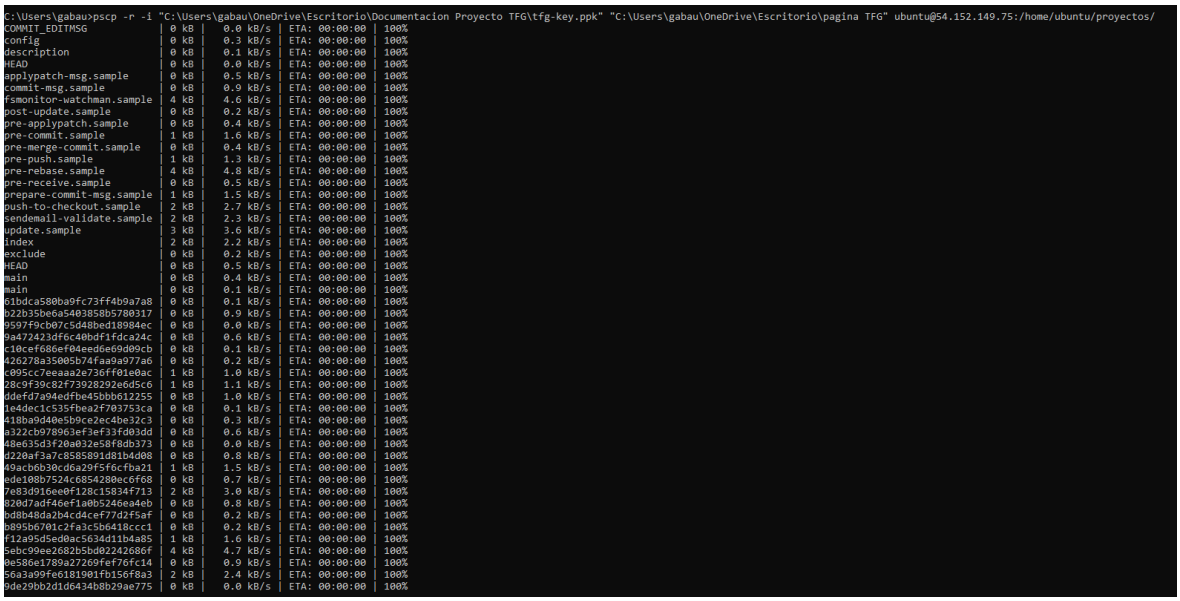


Ilustración 26.Transferencia PSCP

Una vez que se haya transferido todos los archivos, vamos a la instancia y listamos los ficheros con "ls" para ver si se ha completado:

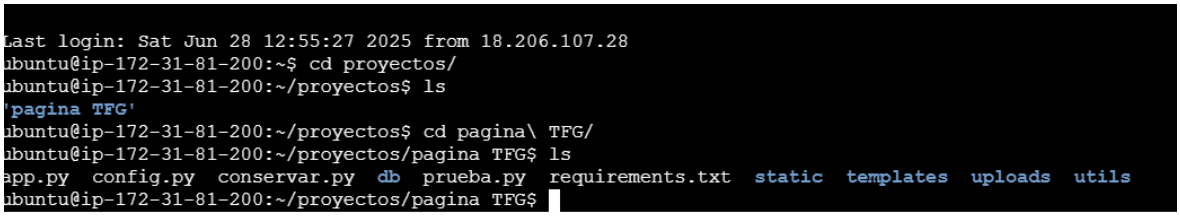


Ilustración 27. Instancia EC2

Como podemos observar se ha completado la transferencia, para ejecutar la aplicación web desde la instancia se empleara la orden python3 app.py

Esta nos devolverá la url hacia la pagina web con la ip privada, pero para poder acceder desde fuera hay que hacerlo con la ip pública ya que la privada solo es accesible en local.

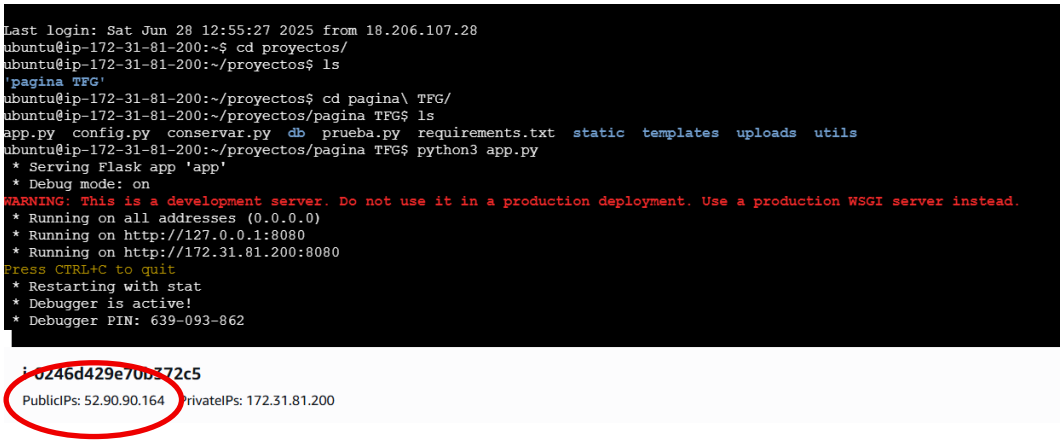


Ilustración 28. Despligue app web en instancia

Como vemos se ha lanzado, y cada vez que hay una nueva entrada al sistema, la instancia lo registra en la consola todas las actividades que se realizan.

Cuando accedemos con el navegador a la dirección url http://52.90.90.164:8080/ no sale esto en consola:

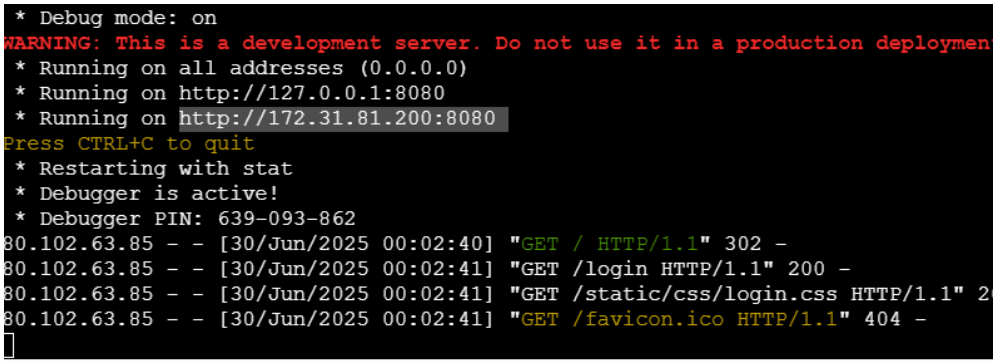


Ilustración 29. Registros en EC2

Hay que tener en cuenta que por configuraciones de AWS la instancia cada día cambia de ip pública, por lo que cuando queramos acceder de nuevo a la instancia tendremos que verificar cual es la ip pública en ese momento.

4.4. ANÁLISIS AUTOMATIZADO CON HERRAMIENTAS DE AUDITORÍA

4.4.1. Escaneo de vulnerabilidades con Nmap

4.4.1.1. Reconocimiento de servicios y versiones

El primer paso para realizar esta auditoria fue realizar un escaneo básico de puertos con Nmap para obtener información sobre las diferentes conexiones que se realizaban a través de la instancia que alojaba el sitio web.

Para poder realizar este escaneo necesitamos de la ip publica de la instancia que alojaba este servicio, en este experimento estaba construida de esta forma: http:// 34.239.0.29:8080/login

Se utilizó el siguiente comando para identificar los puertos abiertos por defecto en la instancia:

```
nmap 34.239.0.29
```

Este escaneo nos revela que el host que alojaba el sitio web estaba activo y tenia los puertos 22 (SSH) y 8080 (HTTP-proxy), mientras que los puertos 80 y 443 aparecen cerrados:

```
C:\Users\gabau>nmap 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 19:31 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.12s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   open   ssh
80/tcp   closed http
443/tcp  closed https
8080/tcp open   http-proxy
```

Ilustración 30 Nmap-Reconocimiento de servicios y versiones

Teniendo esta información pasamos a realizar el escaneo del puerto 8080 para obtener más información del host:

```
nmap -p 8080 34.239.0.29
```

Con este comando nmap realiza un escaneo de red dirigida al ip pública de la instancia EC2 en el puerto 8080.

```
C:\Users\gabau>nmap -p 8080 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 19:32 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.10s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.75 seconds
```

Ilustración 31.Nmap-Reconocimiento de servicios y versiones

Con este comando nos ha dado como resultado que el puerto donde se accede a la web está abierto y que esta aloja un servicio http-proxy. También obtenemos que que la ip 34.239.0.29 proviene de una instancia de AWS y que esta se encuentra activa.

Para obtener información detallada sobre todos los puertos abiertos posibles, servicios en ejecución e identificación del sistema operativo de tu instancia EC2, utilizaremos nmap con parámetros específicos que nos permitan realizar un análisis exhaustivo de la superficie de ataque.

El siguiente comando combina técnicas de escaneo completo de puertos, detección de servicios y fingerprinting del sistema operativo:

nmap -Pn -T4 -p- -sV 34.239.0.29

```
C:\Users\gabau>nmap -Pn -T4 -p- -sV 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 20:53 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.12s latency).
Not shown: 65531 filtered tcp ports (no-response)
PORT     STATE  SERVICE    VERSION
22/tcp   open   ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu0.13 (Ubuntu Linux; protocol 2.0)
80/tcp   closed http
443/tcp  closed https
8080/tcp closed http-proxy
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 145.50 seconds
```

Ilustración 32. Nmap-Reconocimiento de servicios y versiones

Con los parámetros -Pn se omite el ping inicial para que el escaneo sea más silencioso, con -T4 aumentamos la velocidad del escaneo, pero sin que este sea muy ruidoso en la red, con -p- indicamos que escanee todos los puertos posibles en el rango de red (de 1 al 65535) y por último con -sV nmap activara una función que detectara versiones de servicios del host escaneado

4.4.1.2. Análisis de configuraciones HTTP

En este apartado vamos a analizar las configuraciones del servicio HTTP expuesto en el puerto 8080, esto tendrá como objetivo identificar cabeceras de seguridad implementadas, tecnologías utilizadas y posibles fugas de información que pueden llegar a ser aprovechadas por un atacante que encontrase estas vulnerabilidades.

Para obtener más informaciones específicas sobre el servicio web que se encuentra en ejecución en el puerto 8080 de la instancia del objetivo, emplearemos scripts específicos en nmap del Nmap Scripting Engine (NSE) que extraen metadatos del servidor HTTP:

C:\Users\gabau>nmap --script http-headers,http-title,http-server-header -p 8080 34.239.0.29

```
C:\Users\gabau>nmap --script http-headers,http-title,http-server-header -p 8080 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 22:08 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.11s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
|_http-server-header: Werkzeug/3.1.3 Python/3.10.12
| http-title: CloudSecure ERP - Portal de Gesti\xC3\xB3n
|_Requested resource was /login
| http-headers:
|   Server: Werkzeug/3.1.3 Python/3.10.12
|   Date: Thu, 26 Jun 2025 20:08:40 GMT
|   Content-Type: text/html; charset=utf-8
|   Content-Length: 2417
|   Connection: close
|
|_  (Request type: HEAD)

Nmap done: 1 IP address (1 host up) scanned in 1.68 seconds
```

Ilustración 33. Nmap-Análisis de configuraciones HTTP

Con el escaneo con scripts NSE se ha podido revelar información crítica sobre el servicio web alojado en el puerto 8080, como podemos observar logra detectar que la instancia en la que esta alojado el sitio web esta ejecutando una aplicación web ERP que fue desarrollada en Python en la versión 3.10.12 utilizando Werkzeug/3.1.3 (típicamente conocido como Flask) y que esta posee un sistema de autenticación que requiere de login, cabe destacar que ha detectado el nombre de la web "CloudSecure ERP" lo cual demuestra toda la información sensible que queda expuesta a través de la ip.

Con la exposición de esta información se presentan múltiples vectores de ataque críticos. El uso de Werkzeug en el proceso de producción resulta altamente peligroso ya que este servidor puede exponer funcionalidades de debugging que permiten ejecución remota de código y revelación de información sensible del sistema. Además, las versiones específicas reveladas (Werkzeug 3.1.3, Python 3.10.12) facilitan a los atacantes identificar vulnerabilidades conocidas para realizar ataques dirigidos, mientras que el contexto de la aplicación ERP y su portal de login proporcionan información valiosa para orientar ataques de fuerza bruta y reconocimiento adicional de endpoints críticos que manejan datos empresariales sensibles.

Ahora pasaremos a analizar las medidas de protección implementadas a nivel de aplicación web, para ello emplearemos el script NSE especializado en la auditoria de cabeceras de seguridad HTTP. Con este análisis nos permitirá identificar la ausencia o configuración inadecuada de controles de seguridad fundamentales:

```
C:\Users\gabau>nmap --script http-security-headers -p 8080 34.239.0.29
```

```
C:\Users\gabau>nmap --script http-security-headers -p 8080 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 22:10 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.10s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.66 seconds
```

Ilustración 34. Nmap-Análisis de configuraciones HTTP

Tras el escaneo en búsqueda de cabeceras de seguridad, el análisis nos confirma la ausencia total de estas en la aplicación de ERP. Con la falta de output especifico en el script, nos indica que el servidor no implementa ningún tipo de cabecera de protección estándar.

Esta carencia representa una vulnerabilidad bastante crítica que expone la aplicación ERP a múltiples vectores de ataques web. Con laa ausencia de estas protecciones, combinada con el uso de Werkzeug en producción, confirma una configuración de seguridad deficiente que facilita la explotación exitosa de vulnerabilidades de aplicación web y compromete la integridad de los datos empresariales gestionados por el sistema.

4.4.1.3. Detección de vulnerabilidades conocidas

Para poder identificar vulnerabilidades específicas documentadas en bases de datos públicas como CVE (Common Vulnerabilities and Exposures), utilizaremos el motor de scripts de Nmap especializado en detección automatizada de exploits conocidos. Este análisis exhaustivo permite descubrir vulnerabilidades críticas que podrían ser explotadas por atacantes utilizando herramientas automatizadas o exploits públicamente disponibles.

```
C:\Users\gabau>nmap --script vuln -p 8080 34.239.0.29
```

El script vuln ejecuta una batería completa de pruebas que incluyen verificación de CVEs específicos para las versiones de software identificadas, detección de configuraciones inseguras conocidas, pruebas de vectores de ataque documentados y evaluación de vulnerabilidades de denegación de servicio. Esta metodología sistemática es fundamental en auditorías de seguridad profesionales, ya que las vulnerabilidades conocidas representan el vector de ataque más común debido a la disponibilidad de exploits automatizados y la facilidad de explotación que ofrecen a atacantes con conocimientos técnicos limitados.

```
C:\Users\gabau>nmap --script vuln -p 8080 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 23:57 Hora de verano romance
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.10s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web server open and hold
|       them open as long as possible.  It accomplishes this by opening connections to
|       the target web server and sending a partial request. By doing so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       http://ha.ckers.org/slowloris/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750

Nmap done: 1 IP address (1 host up) scanned in 658.35 seconds
```

Ilustración 35. Nmap-Detección de vulnerabilidades conocidas

Luego de esperar todo el proceso de ejecución, nos da los resultados del análisis exhaustivo de las vulnerables de la aplicación web, esta ha identificado una vulnerabilidad crítica de denegación de servicio que compromete la disponibilidad del sistema ERP.

4.4.1.3.1. CVE-2007-6750: Slowloris Denial of Service Attack

El escaneo ha confirmado que la aplicación es LIKELY VULNERABLE al ataque Slowloris, una técnica de denegación de servicio de bajo consumo de recursos, pero alta efectividad. Esta vulnerabilidad permite a un atacante agotar las conexiones disponibles del servidor web manteniendo múltiples conexiones HTTP parcialmente abiertas de forma indefinida.

4.4.1.3.2. Mecanismo de explotación

El atacante establece conexiones HTTP legítimas con el servidor objetivo, pero envía headers de forma deliberadamente lenta e incompleta, manteniendo cada conexión activa sin completar la petición. Dado que Werkzeug tiene un pool limitado de conexiones concurrentes, un número relativamente pequeño de conexiones maliciosas (típicamente 100-200) puede saturar completamente la capacidad del servidor, impidiendo que usuarios legítimos accedan al sistema ERP.

4.4.1.3.3. Impacto empresarial

Esta vulnerabilidad representa un riesgo crítico para la continuidad operacional, ya que un atacante con recursos mínimos puede interrumpir completamente el acceso al sistema de gestión empresarial, afectando procesos críticos de negocio, transacciones financieras y operaciones diarias de la organización.

4.4.1.4. Análisis de autenticación SSH

El protocolo SSH (Secure Shell) constituye uno de los vectores de acceso más críticos en la infraestructura de servidores, proporcionando acceso administrativo completo al sistema operativo. Un análisis exhaustivo de la configuración y seguridad del servicio SSH es fundamental para evaluar la superficie de ataque de la instancia, ya que las vulnerabilidades en este protocolo pueden resultar en compromiso total del servidor.

Este análisis incluye la enumeración de algoritmos de intercambio de claves (KEX), verificación de algoritmos de host key, evaluación de métodos de cifrado simétrico implementados, análisis de algoritmos de autenticación de mensajes (MAC) e identificación de las claves públicas del servidor. La importancia de esta evaluación radica en que algoritmos criptográficos débiles u obsoletos pueden permitir ataques de interceptación, manipulación de tráfico o compromiso de credenciales.

```
nmap --script ssh-auth-methods,ssh2-enum-algos,ssh-hostkey -p 22
34.239.0.29
```

Este comando ejecuta tres scripts especializados para realizar una auditoría integral del servicio SSH en el puerto 22:

- **ssh-auth-methods**: Identifica los métodos de autenticación habilitados (password, publickey, keyboard-interactive).
- **ssh2-enum-algos**: Enumera todos los algoritmos criptográficos soportados (cifrado, intercambio de claves, MAC).

- **ssh-hostkey**: Extrae las claves públicas del servidor y sus huellas digitales.

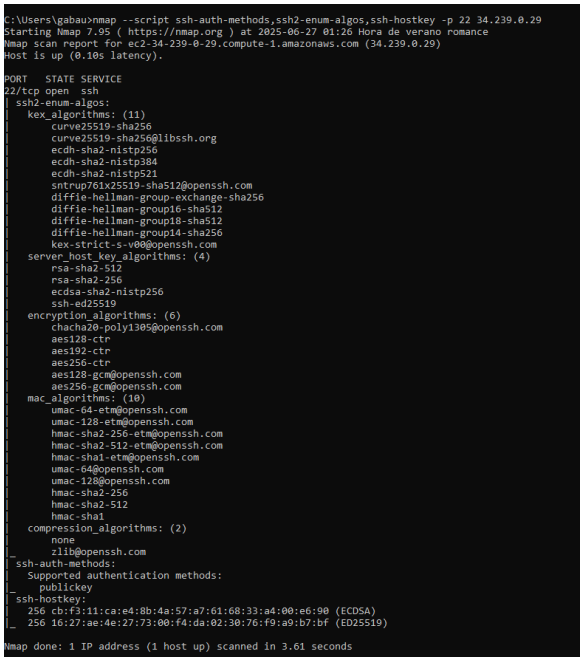


Ilustración 36. Nmap-Análisis de autenticación SSH

Con el anterior análisis se ha demostrado una configuración SSH bastante robusta.

El servidor únicamente acepta autenticación por clave pública (publickey), habiendo deshabilitado métodos menos seguros como autenticación por contraseña (password) y autenticación interactiva (keyboard-interactive). Esta configuración elimina completamente los ataques de fuerza bruta contra credenciales y representa una práctica de seguridad óptima.

La restricción exclusiva a claves públicas indica una política de acceso estricta que requiere que los administradores generen y gestionen pares de claves criptográficas, eliminando el riesgo de credenciales débiles o reutilizadas. Esta configuración impide ataques automatizados de credential stuffing y reduce significativamente la superficie de ataque.

La combinación de algoritmos criptográficos modernos con autenticación exclusivamente basada en claves públicas demuestra una configuración SSH altamente segura. La ausencia de métodos de autenticación débiles, junto con la implementación de host keys ED25519 y ECDSA, proporciona un nivel de seguridad empresarial apropiado para infraestructura crítica.

4.4.1.5. Enumeración de métodos HTTP y rutas accesibles

La enumeración de métodos HTTP y rutas accesibles es una fase fundamental del reconocimiento web que permite identificar las funcionalidades disponibles en un servidor web y posibles vectores de ataque. Esta técnica revela información crítica sobre la superficie de ataque de la aplicación web, incluyendo métodos HTTP permitidos, directorios accesibles y archivos sensibles que podrían estar expuestos.

Para la realización de este análisis hemos hecho uso de dos comandos:

```
nmap --script http-methods -p 8080 34.239.0.29
```

Con este comando nmap utiliza el http-methods para determinar qué métodos HTTP están habilitados en el servidor web que escucha en el puerto 8080. Los métodos HTTP definen las acciones que se pueden realizar sobre los recursos del servidor.

```
C:\Users\gabau>nmap --script http-methods -p 8080 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-27 01:46 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.10s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
| http-methods:
|_  Supported Methods: HEAD OPTIONS GET

Nmap done: 1 IP address (1 host up) scanned in 1.50 seconds
```

Ilustración 37.Nmap-Enumeración de métodos HTTP y rutas accesibles

Con el resultado del análisis, podemos ver que el servidor permite únicamente métodos HTTP básicos y seguros:

- **GET**: Recuperación de recursos (lectura).
- **HEAD**: Obtención de metadatos sin el cuerpo de la respuesta.
- **OPTIONS**: Consulta de métodos permitidos.

La ausencia de métodos potencialmente peligrosos como POST, PUT, DELETE, PATCH o TRACE indica una configuración relativamente segura desde el punto de vista de métodos HTTP.

Ahora vamos a pasar a enumerar los directorios y archivos:

```
nmap --script http-enum -p 8080 34.239.0.29
```

```
C:\Users\gabau>nmap --script http-enum -p 8080 34.239.0.29
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-27 01:46 Hora de verano romance
Nmap scan report for ec2-34-239-0-29.compute-1.amazonaws.com (34.239.0.29)
Host is up (0.10s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 523.79 seconds
```

Ilustración 38.Nmap-Enumeración de métodos HTTP y rutas accesibles

Con este comando se emplea el script http-enum para realizar un escaneo exhaustivo de directorios y archivos comunes en el servidor web, buscando rutas accesibles que podrían contener información sensible o puntos de entrada adicionales.

Tras realizar el escaneo no se identificó rutas accesibles utilizando los diccionarios predeterminados de Nmap. Esto puede indicar:

- Configuración restrictiva del servidor web
- Estructura de directorios no convencional
- Posible uso de rutas personalizadas o API endpoints específicos
- Medidas de seguridad que impiden la enumeración automática

Los resultados muestran un servidor con una configuración básica que limita los métodos HTTP a operaciones de lectura, lo cual es una buena práctica de seguridad. Sin embargo, la ausencia de rutas detectables no garantiza la seguridad, ya que podrían existir endpoints específicos de la aplicación que requieren técnicas de enumeración más avanzadas o conocimiento previo de la estructura de la aplicación.

4.4.2. Análisis de tráfico con Wireshark

Wireshark es una herramienta fundamental de análisis de protocolos de red que permite la captura, inspección y análisis detallado del tráfico que circula por la infraestructura de red. En el contexto de este proyecto de ciberseguridad en entornos cloud, Wireshark se utilizará para realizar un análisis exhaustivo de las comunicaciones entre los servicios AWS desplegados y los clientes que interactúan con ellos.

Con Esta herramienta se proporcionará visibilidad completa sobre los protocolos utilizados, identificará transferencia de datos no cifradas que puedan exponer información sensible, detectará credenciales transmitidas en texto plano, y revelará patrones de comunicación anómalos que podrían indicar configuraciones inseguras en los servicios RDS, DynamoDB, S3 y EC2. Además, permitirá verificar la implementación correcta de protocolos de seguridad como TLS/SSL y analizar posibles ataques man-in-the-middle o interceptación de sesiones, complementando así la información de reconocimiento obtenida mediante el escaneo de puertos y la enumeración de servicios web.

4.4.2.1. Captura de credenciales en aplicaciones HTTP

4.4.2.1.1. Man-in-the-Middle (MITM)

Los ataques Man-in-the-Middle (MITM) representan una de las amenazas más críticas en la seguridad de las comunicaciones digitales. Este tipo de ataque se produce cuando un atacante se posiciona estratégicamente entre dos partes que se comunican, interceptando, modificando o inyectando datos en el flujo de comunicación sin que ninguna de las partes legítimas sea consciente de la intrusión.

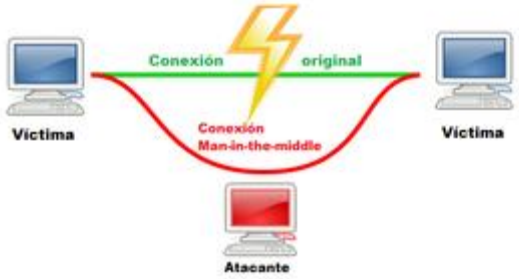


Ilustración 39 Man-in-the-Middle (MITM)

En el contexto de aplicaciones web que utilizan el protocolo HTTP sin cifrado, los ataques MITM se vuelven especialmente peligrosos debido a la naturaleza de texto plano de las transmisiones. Cuando los usuarios

introducen credenciales de acceso, información personal o datos sensibles en formularios web que se transmiten a través de HTTP, toda esta información viaja por la red sin ningún tipo de protección criptográfica, siendo completamente visible para cualquier atacante que tenga acceso al tráfico de red.

Esta vulnerabilidad fundamental del protocolo HTTP sin cifrado convierte a Wireshark en una herramienta especialmente efectiva para demostrar la facilidad con la que se pueden comprometer las credenciales de usuario en aplicaciones web mal configuradas, evidenciando la necesidad crítica de implementar HTTPS en cualquier aplicación que maneje información sensible.

Al lanzar la aplicación web, el usuario accede a la página de login de la aplicación web vulnerable desplegada en AWS EC2 (IP: 54.197.196.244:8080) e introduce sus credenciales de acceso en el formulario de autenticación. Al enviar los datos, estos se transmiten sin cifrado a través del protocolo HTTP, siendo completamente visibles para cualquier atacante que esté monitorizando el tráfico de red. Mediante Wireshark, se puede interceptar y capturar en tiempo real toda la información enviada, incluyendo nombres de usuario, contraseñas y otros datos sensibles, los cuales aparecen en texto plano dentro de los paquetes HTTP POST dirigidos al endpoint /login del servidor.

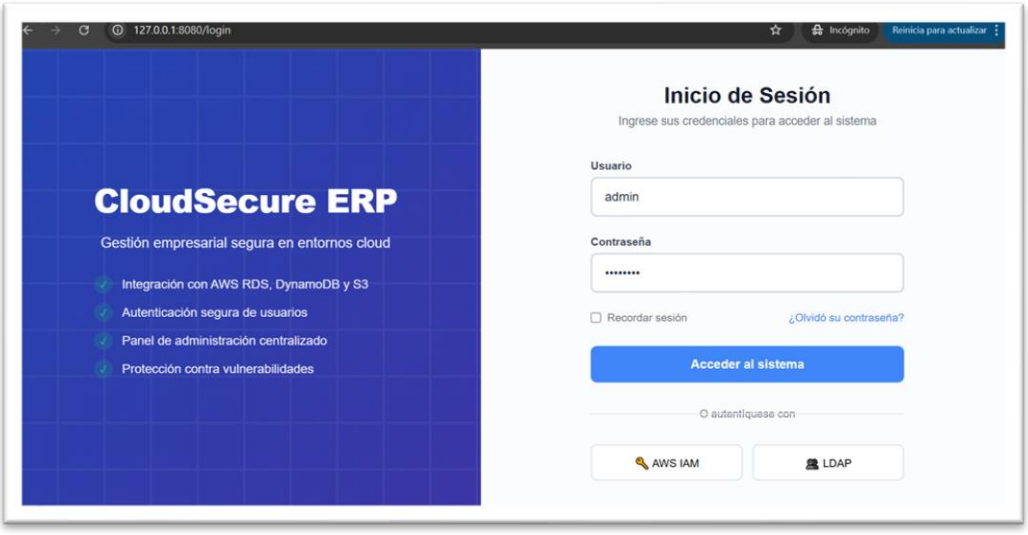


Ilustración 40. Login usuario

Si un administrador del sistema iniciara sesión en la aplicación web, sus credenciales privilegiadas quedarían completamente expuestas durante la transmisión, permitiendo a un atacante comprometer no solo

la cuenta administrativa, sino obtener acceso completo al panel de control de la aplicación. Esta escalada de privilegios representaría un riesgo crítico para la seguridad de toda la infraestructura, ya que los atacantes podrían acceder a funcionalidades administrativas sensibles, modificar configuraciones del sistema, comprometer datos de otros usuarios, y potencialmente utilizar los privilegios administrativos como punto de pivote para atacar otros componentes de la infraestructura AWS subyacente.

Para verificar esta vulnerabilidad, se configura Wireshark para capturar el tráfico de red en la interfaz Wi-Fi, aprovechando que tanto el atacante como la víctima se encuentran en la misma red local. Esta configuración permite interceptar todas las comunicaciones HTTP que circulan por el segmento de red, incluidas las credenciales transmitidas hacia la aplicación web vulnerable.

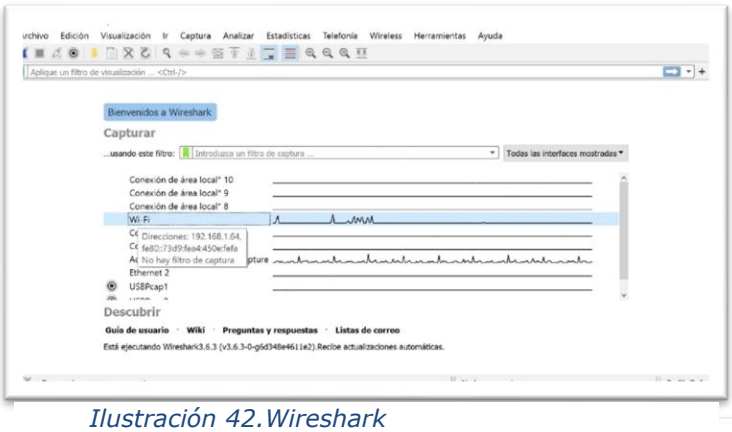


Ilustración 42.Wireshark

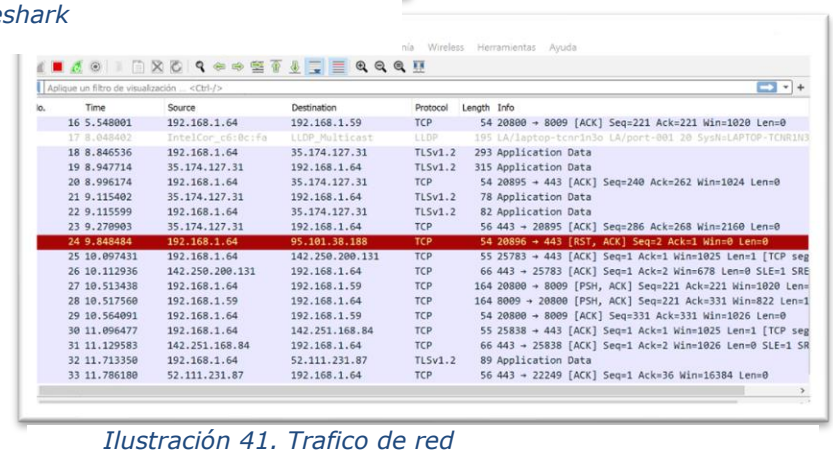


Ilustración 41. Trafico de red

Esto lo que hará es que Wireshark empiece a capturar todo el tráfico de red que pasa por la Wi-Fi, incluyendo los datos cifrados como los no cifrados.

Una vez iniciada la captura de tráfico en Wireshark, el administrador procede a autenticarse en la aplicación web introduciendo sus credenciales en el formulario de login. Durante este proceso, todas las comunicaciones HTTP son interceptadas y registradas por la máquina atacante que monitoriza el tráfico de red, permitiendo la visualización en tiempo real de los datos de autenticación transmitidos sin cifrado entre el cliente y el servidor.

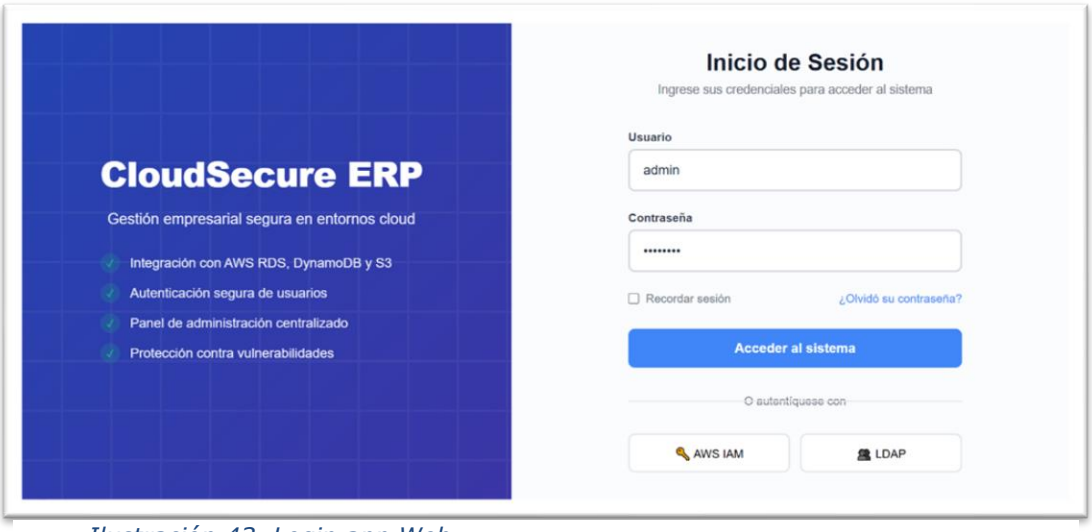


Ilustración 43. Login app Web

Cuando se haya capturado ya todos los paquetes que indicaran que se está realizando un inicio de sesión, la maquina atacante procede a detener la captura de paquetes registrados.

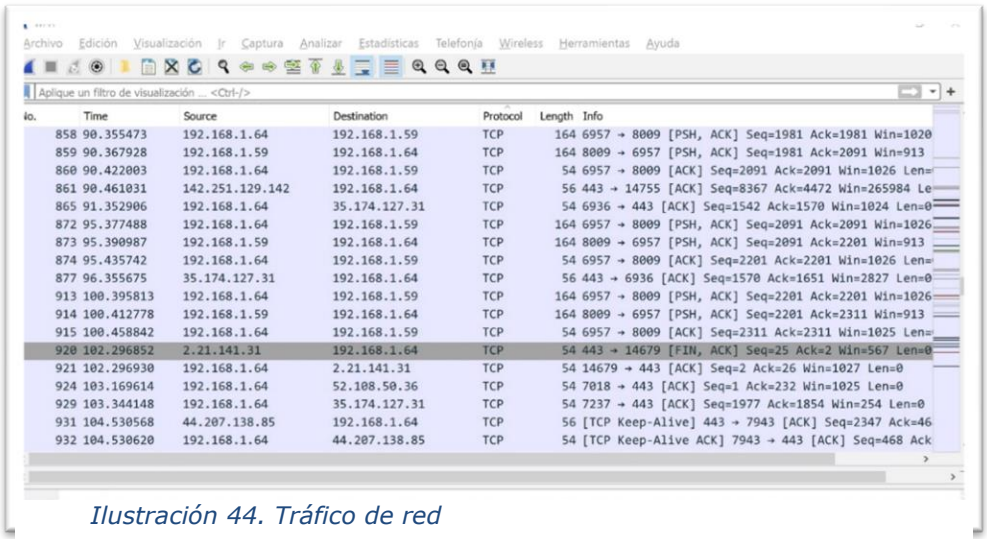


Ilustración 44. Tráfico de red

Al saber cual es la ip pública del sitio web, la maquina atacante filtra todos los paquetes a aquellas que tengan en los registros del paquete la dirección 54.197.196.244, este filtro se puede aplicar en el buscador ubicado en la parte superior con la orden ip.addr == 54.197.196.244:

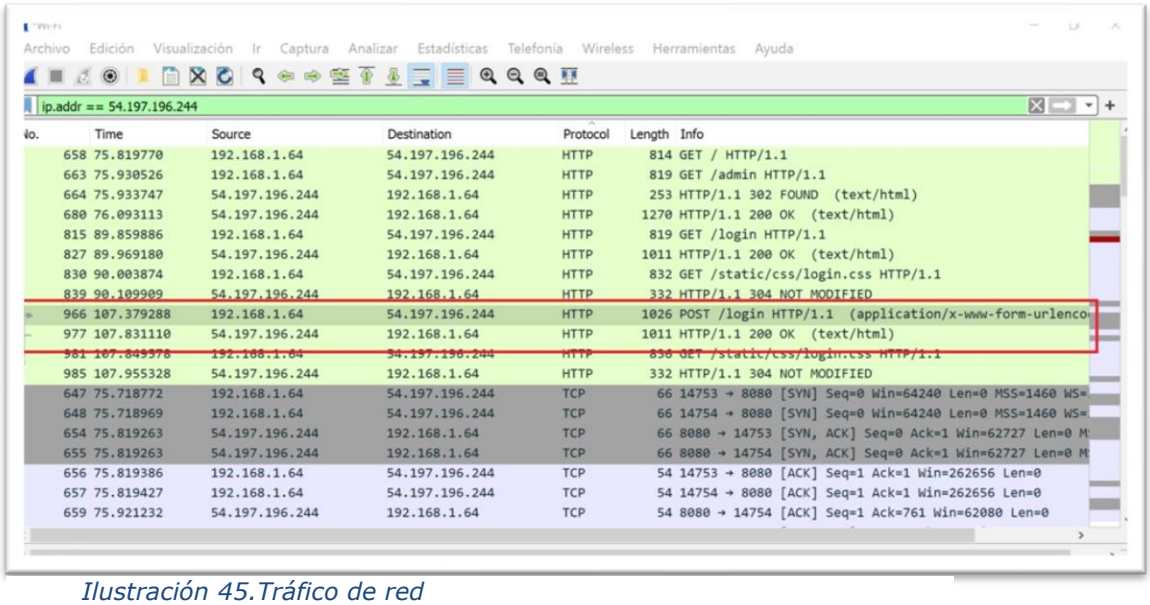


Ilustración 45.Tráfico de red

Una vez completada la captura, el atacante filtra los paquetes correspondientes a la IP objetivo buscando específicamente peticiones HTTP POST, ya que estos métodos son utilizados por la aplicación web para enviar las credenciales de autenticación al servidor y verificar su validez contra la base de datos de usuarios.

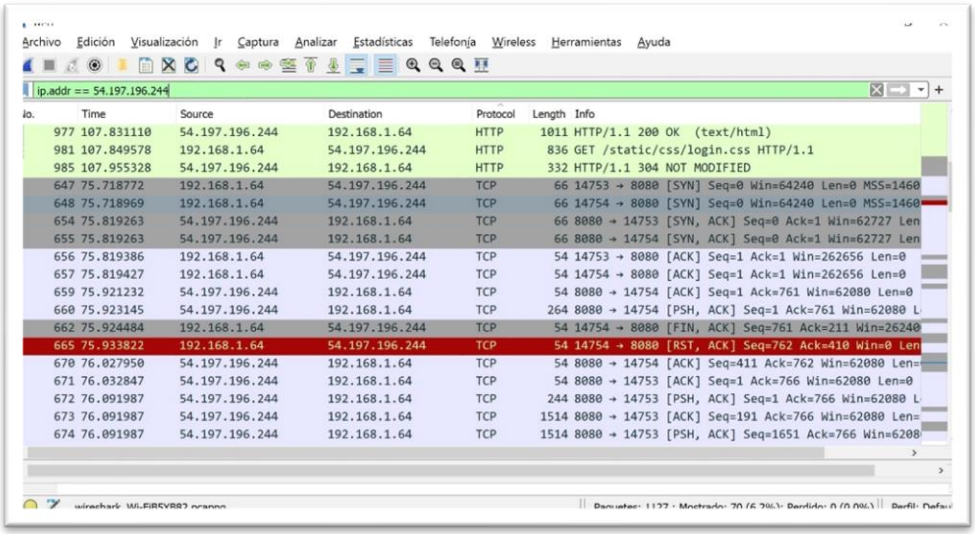


Ilustración 46. Tráfico de red

En la captura se puede observar claramente el proceso de autenticación vulnerable: el paquete 966 muestra la petición POST /login HTTP/1.1 con el tipo de contenido application/x-www-form-urlencoded, que contiene las credenciales enviadas por el administrador. Inmediatamente después, el paquete 977 presenta la respuesta del servidor con el código de estado HTTP/1.1 200 OK, confirmando que las credenciales fueron validadas correctamente y que el proceso de autenticación fue exitoso. Esta secuencia demuestra cómo las credenciales administrativas viajan en texto plano a través de la red y son aceptadas por el servidor, evidenciando la vulnerabilidad crítica en la transmisión de datos sensibles sin cifrado.

Al interceptar el paquete con la petición de autenticación, el atacante procede a analizar el tráfico capturado utilizando la función "Follow TCP Stream" de Wireshark, lo que le permite reconstruir toda la comunicación entre el cliente (192.168.1.64) y el servidor (54.197.196.244:8080). De esta manera, no solo captura las credenciales transmitidas en texto plano (username=admin&password=admin123) en la petición POST al endpoint /login, sino que también obtiene acceso a toda la sesión HTTP completa, incluyendo las cookies de sesión, headers de autenticación, respuestas del servidor y cualquier otro dato sensible que se intercambie durante la comunicación, aprovechando la vulnerabilidad crítica de que el sitio web utiliza HTTP en lugar de HTTPS, dejando toda la información expuesta sin cifrado alguno.

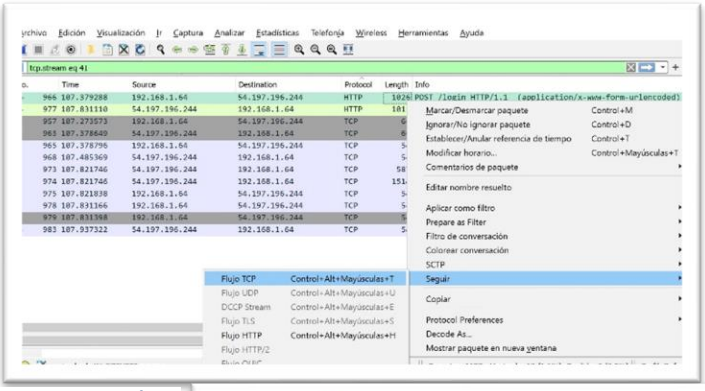


Ilustración 47.Tráfico de red

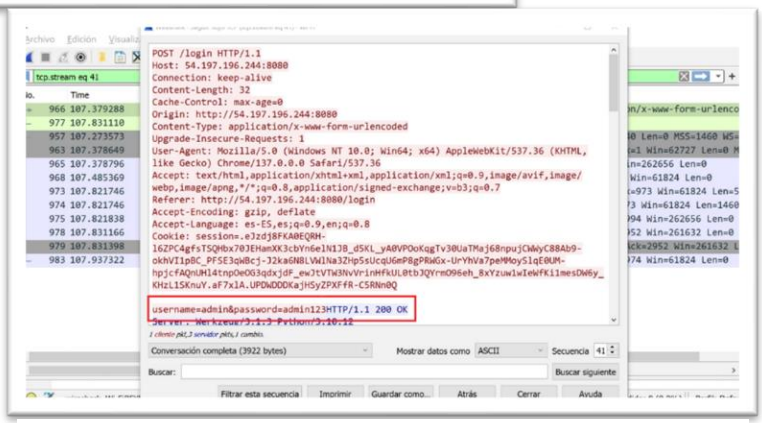


Ilustración 48. Tráfico de red

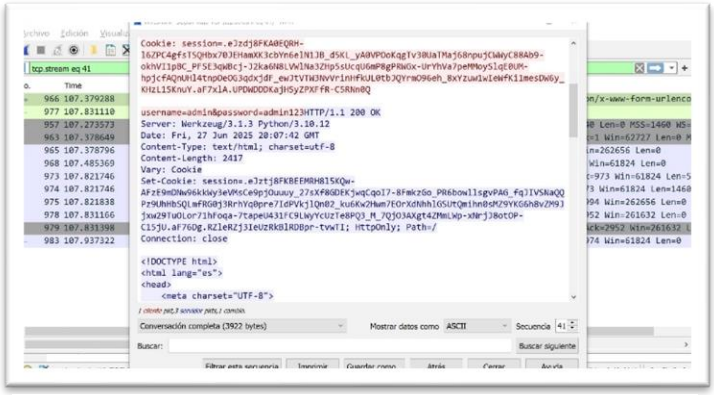


Ilustración 49. Tráfico de red

Esto demuestra lo inseguro que es esta configuración, ya que ahora el atacante puede autenticarse como administrador utilizando las credenciales capturadas, obteniendo acceso completo al sistema y comprometiendo la integridad, confidencialidad y disponibilidad de toda la aplicación web y sus datos.

4.4.2.2. Intercepción de ficheros transmitidos sin cifrado

El texto plano no constituye el único riesgo en este tipo de ataques de interceptación. La ausencia de cifrado permite también la captura completa de ficheros transmitidos durante las sesiones HTTP, incluyendo documentos confidenciales, imágenes, archivos de configuración y cualquier otro tipo de contenido que se intercambie entre cliente y servidor. Mediante el análisis del flujo TCP, el atacante puede extraer y reconstruir estos archivos en su totalidad, accediendo así a información sensible que va más allá de las simples credenciales de acceso. Esta capacidad de interceptar transferencias de ficheros amplifica significativamente el impacto del ataque, ya que compromete no solo el acceso al sistema, sino también la confidencialidad de todos los datos que circulan por la red sin protección criptográfica.

Al acceder el usuario al panel "mis-documentos", la aplicación web genera una petición al servidor para recuperar los archivos personales almacenados en el bucket S3 correspondiente. Durante este proceso, el atacante mantiene activa la captura de tráfico en Wireshark, monitorizando continuamente la red hasta confirmar que se han completado las transferencias de los documentos solicitados. Esta estrategia permite interceptar no solo las peticiones de descarga, sino también el contenido completo de los archivos transmitidos sin cifrado,

comprometiendo así la confidencialidad de la información personal del usuario.

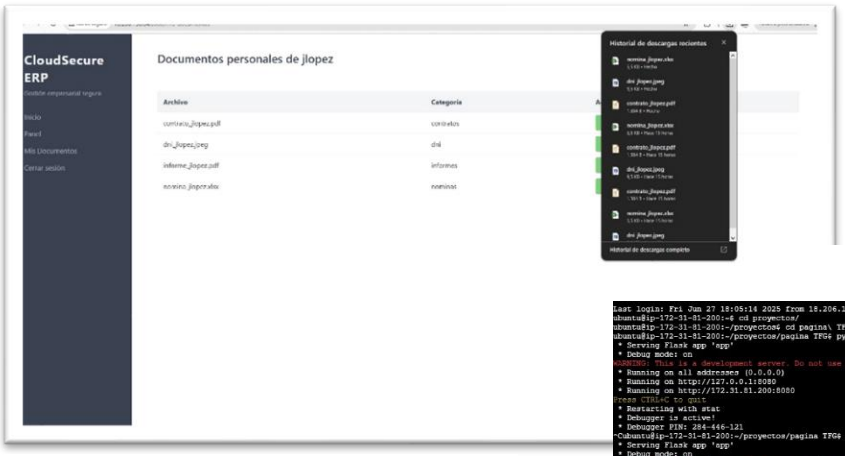


Ilustración 50. Tráfico de red

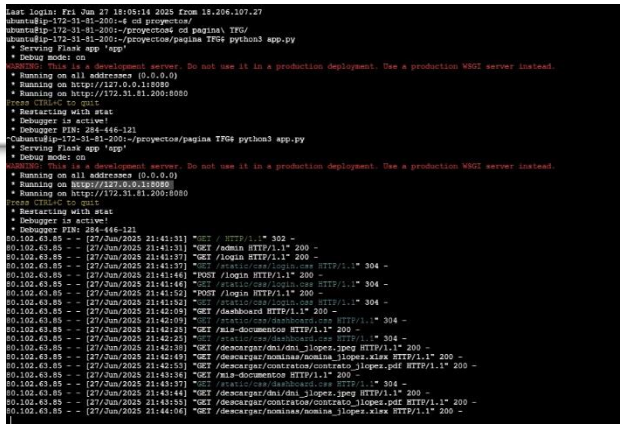


Ilustración 51. Tráfico de red

Como vemos en la imagen, el usuario ha descargado ficheros con información personal en formato PDF, XLSX y JPEG. El atacante, mediante el análisis del flujo de datos capturado, puede extraer y reconstruir estos archivos completos, obteniendo acceso directo a documentos confidenciales, hojas de cálculo con datos sensibles e imágenes privadas que contengan información personal identificable, comprometiendo así la privacidad y seguridad de los datos del usuario de manera integral.

El atacante filtra el tráfico en Wireshark utilizando el comando ip.addr == 54.197.196.244 para aislar las comunicaciones con el servidor objetivo. Posteriormente, identifica los paquetes con cabeceras HTTP que indican transferencias de documentos y analiza aquellos con tamaños de payload considerables, lo cual revela las descargas de archivos. Esta técnica permite localizar específicamente las transmisiones de ficheros entre el extenso volumen de tráfico capturado, facilitando la extracción selectiva de los documentos interceptados.

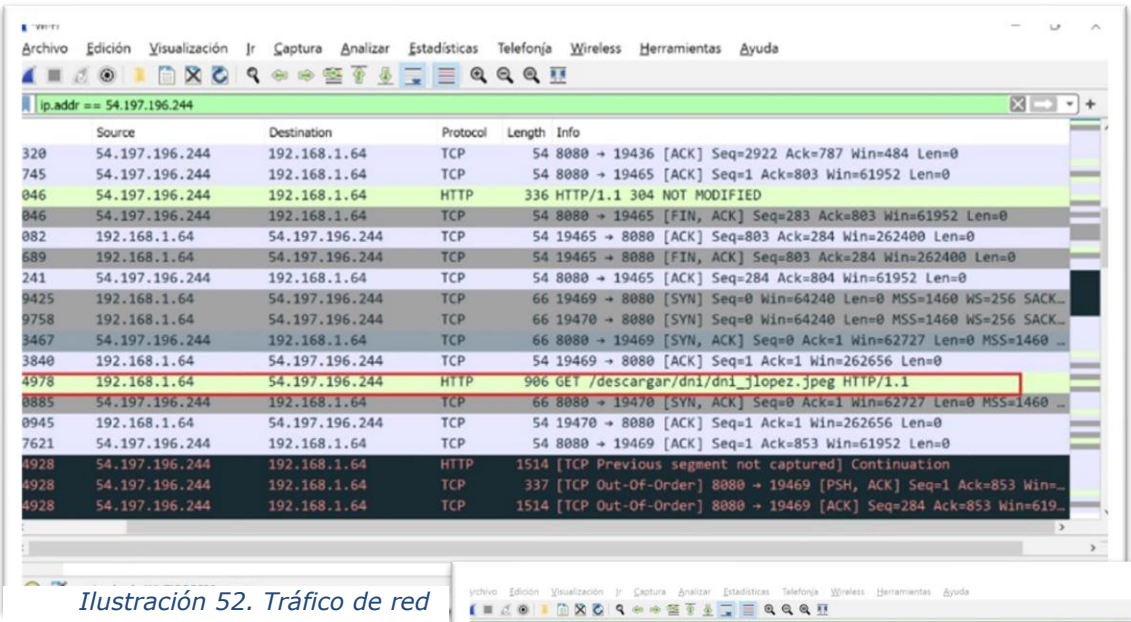


Ilustración 52. Tráfico de red

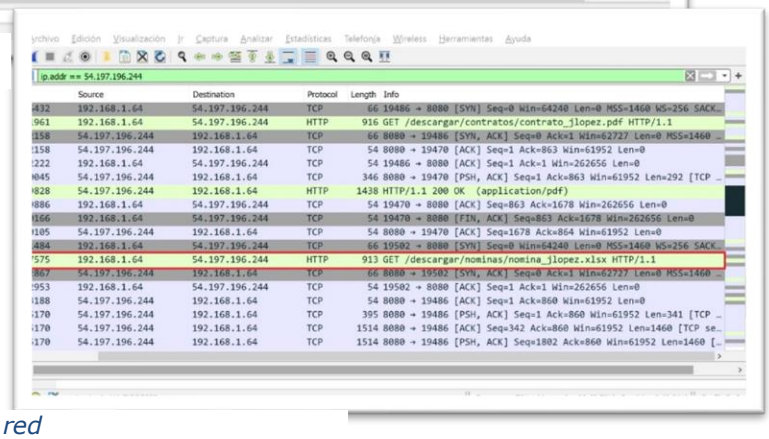


Ilustración 53. Tráfico de red

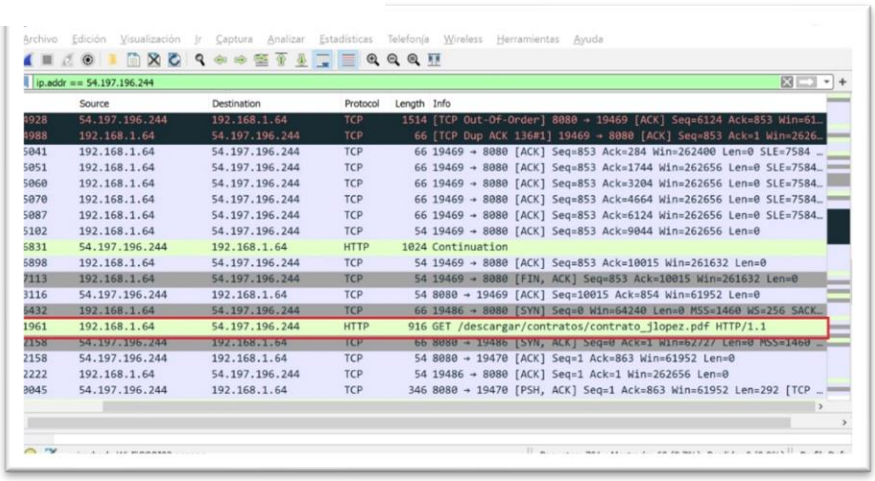


Ilustración 54. Tráfico de red

El atacante una vez que interceptación los fichero puede exportarlo yendo a Archivo>Exportar objetos>HTTP:

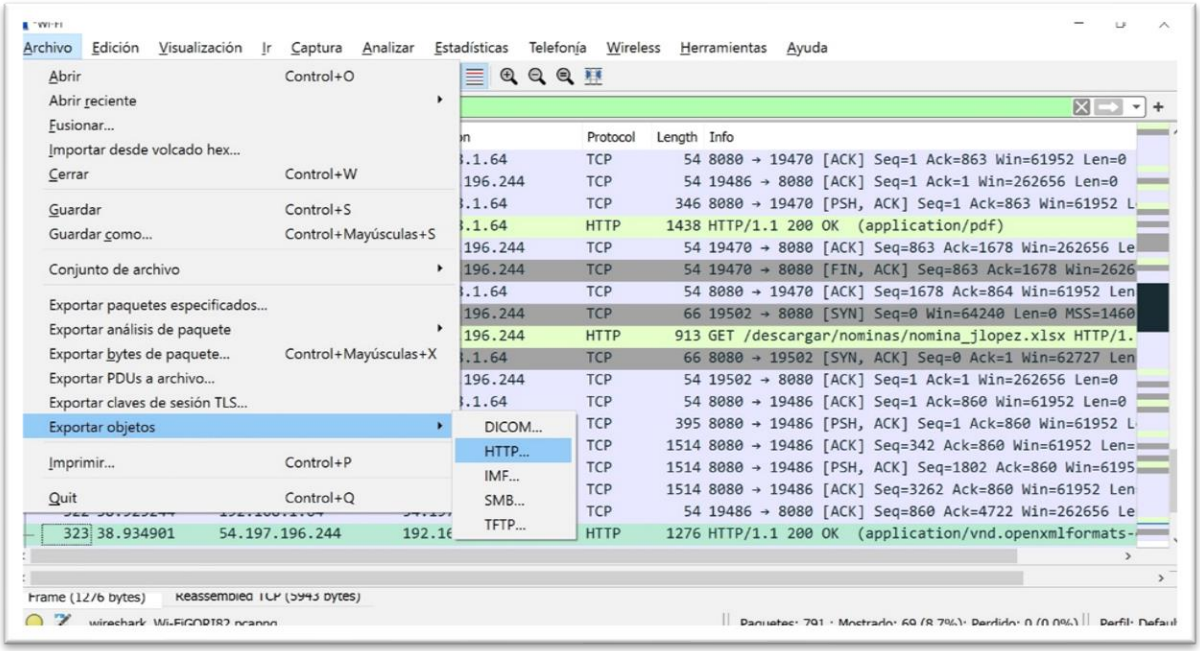


Ilustración 55. Exportación Ficheros

Una vez ahí, Wireshark lista todos los ficheros que se han captado y permite al atacante poder guardarlos en su formato original.

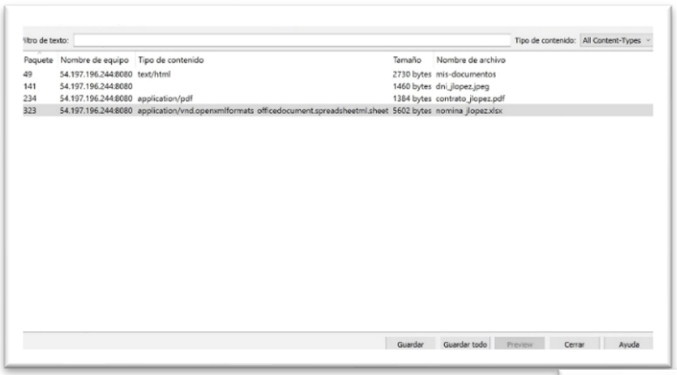
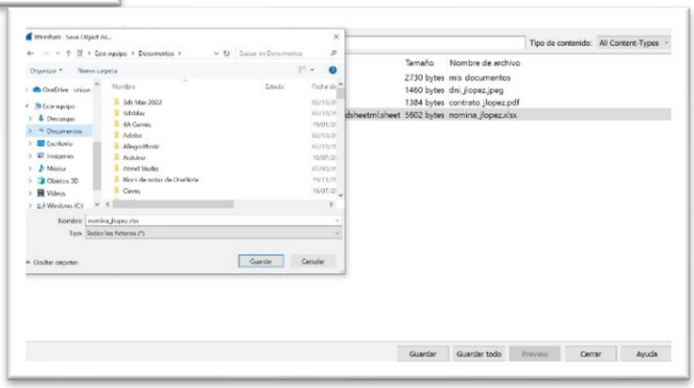


Ilustración 57. Exportación Ficheros

Ilustración 56. Exportación Ficheros



Ahora el atacante tiene acceso a todo tipo de información privada, incluyendo datos personales identificables, documentos confidenciales y credenciales de autenticación. Esta brecha compromete gravemente la privacidad del usuario y expone información crítica susceptible de ser utilizada para suplantación de identidad, fraude o chantaje. El impacto trasciende al usuario individual, generando responsabilidades legales bajo normativas como el RGPD y comprometiendo la reputación y confianza en la organización responsable de la protección de estos datos.

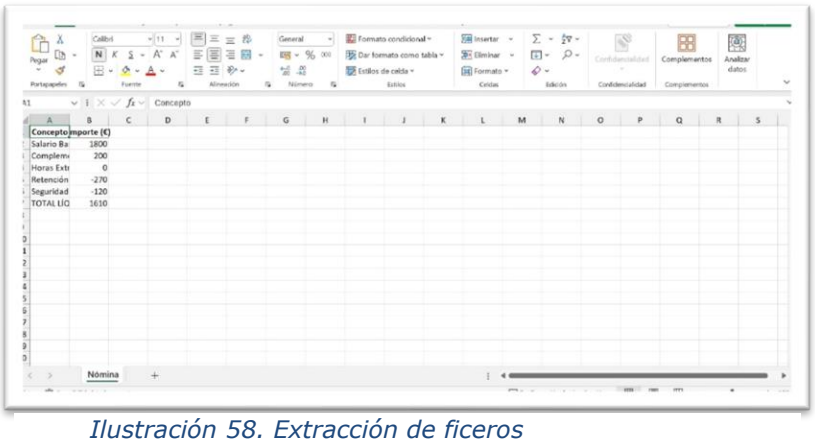


Ilustración 58. Extracción de ficeros

4.4.3. Auditoría de vulnerabilidades con SQLMap

4.4.3.1. ¿Qué son las SQL Injections?

Las SQL Injections son una de las vulnerabilidades más críticas en aplicaciones web. Ocurren cuando una aplicación permite que datos no validados del usuario se inserten directamente en consultas SQL, permitiendo a un atacante manipular la base de datos.

4.4.3.2. ¿Cómo se producen?

Las SQL Injections se dan cuando el código de la aplicación construye consultas SQL de forma insegura:

```
SELECT * FROM users WHERE username = '$username' AND password = '$password'
```

Si un atacante introduce en la web: admin' OR '1'='1' --' AND password = '$password', la consulta se convierte en:

```
SELECT * FROM users WHERE username = 'admin' OR '1'='1'
```

Lo cual siempre devolverá todos los registros de la tabla users porque '1'='1' es siempre verdades

4.4.3.3. ¿Por qué ocurren?

Las SQL Injections ocurren principalmente por la falta de validación adecuada de los datos de entrada del usuario, lo que permite que información maliciosa llegue directamente a las consultas SQL. Otro factor crítico es la concatenación directa de strings en las consultas, donde los desarrolladores construyen sentencias SQL uniendo directamente los datos del usuario sin ningún tipo de filtrado. La ausencia de consultas preparadas (prepared statements) agrava el problema, ya que estas separan la lógica SQL de los datos, impidiendo que el código malicioso se ejecute. Muchas aplicaciones confían excesivamente en las validaciones del lado cliente, que pueden ser fácilmente bypaseadas por un atacante, y finalmente, la falta de escapado de caracteres especiales permite que símbolos como comillas simples o punto y coma alteren la estructura de las consultas SQL, dando lugar a la inyección de código malicioso.

4.4.3.4. SQLMap: La herramienta definitiva

SQLMap es la herramienta más potente para detectar y explotar SQL Injections de forma automatizada. Puede:

- Extraer datos de bases de datos
- Detectar vulnerabilidades automáticamente
- Realizar bypass de autenticación
- Obtener acceso al sistema operativo

4.4.3.4.1. Detección de puntos vulnerables

En esta sección se evalúa la seguridad de la aplicación web mediante el uso de SQLMap. El comando utilizado permite identificar bases de datos accesibles a través de formularios web vulnerables:

```
py sqlmap.py –url http://34.239.0.29:8080/login --dbs --batch --forms
```

Este comando analiza el formulario de login de la aplicación, busca parámetros susceptibles a inyección SQL y enumera las bases de datos disponibles en caso de encontrar vulnerabilidades. La opción --batch automatiza el proceso sin requerir intervención manual, mientras que --forms se enfoca específicamente en los campos de formulario presentes en la página.

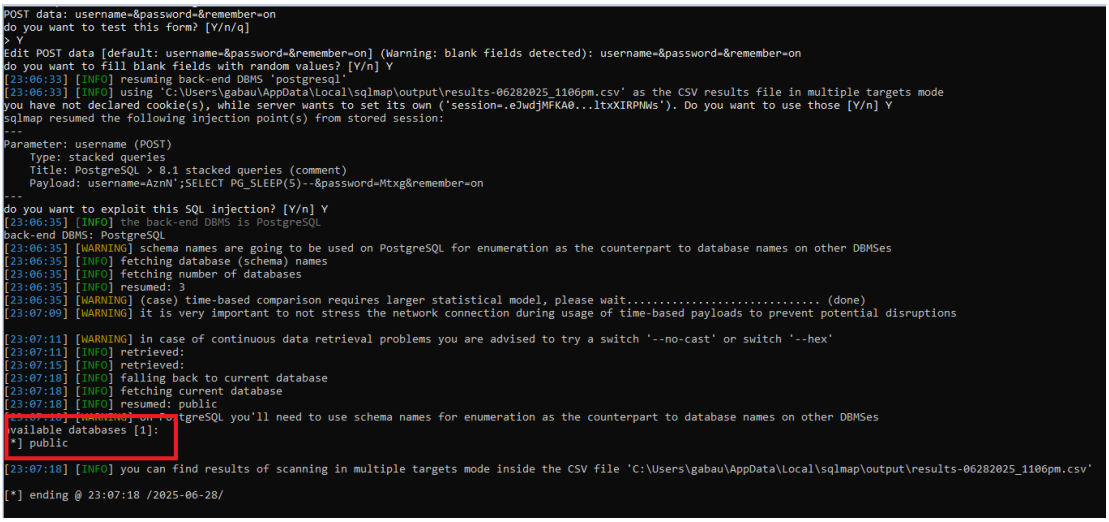


Ilustración 59- SQLMap-Detección de puntos vulnerables

La captura muestra una auditoría exitosa de SQL injection realizada con SQLMap contra una aplicación web en el puerto 8080. La herramienta detectó una vulnerabilidad de tipo "stacked queries" en el formulario de login, específicamente en los parámetros username, password y remember del método POST. El sistema objetivo utiliza PostgreSQL versión 8.1 como gestor de base de datos.

SQLMap logró enumerar exitosamente las bases de datos disponibles y determinó que la base de datos actual es "public". La vulnerabilidad de stacked queries es particularmente grave porque permite ejecutar múltiples consultas SQL consecutivas, lo que significa que un atacante podría no solo extraer información sensible, sino también modificar datos o potencialmente ejecutar comandos del sistema.

4.4.3.4.2. Enumeración de tablas

Una vez confirmada la vulnerabilidad de stacked queries en PostgreSQL e identificada la base de datos "public", el siguiente paso consistió en enumerar las tablas existentes dentro de dicha base de datos. Para ello se ejecutó el comando:

```
py sqlmap.py --url http://34.239.0.29:8080/login -D public --tables --batch --forms
```

Este comando aprovecha la misma vulnerabilidad detectada anteriormente en el formulario de login pero esta vez especifica la base de datos objetivo (-D public) y solicita la enumeración de tablas (--tables). Este comando permite descubrir la estructura interna de la base de datos, revelando qué tablas contiene y proporcionando información valiosa sobre la organización de los datos almacenados, lo cual es fundamental para planificar posteriores extracciones de información específica.

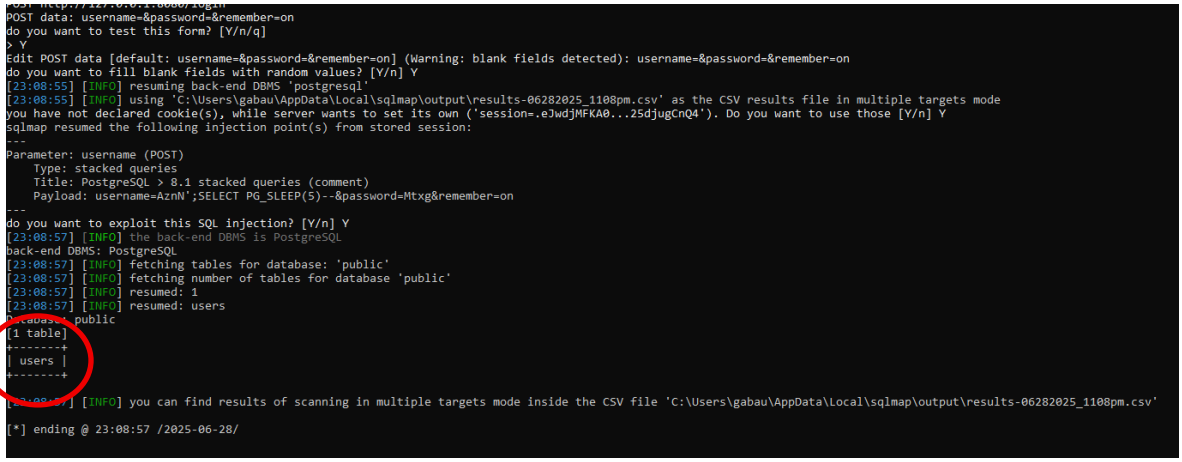


Ilustración 60. SQLMap-Enumeración de tablas

En este caso ha detectado la tabla de users que es la que conecta con el formulario de la página web para la validación de usuarios.

4.4.3.4.3. Extracción de datos sensibles

Tras identificar las tablas disponibles en la base de datos "public", se procedió a extraer información crítica de la tabla "users", que presumiblemente contiene credenciales y datos personales de los usuarios del sistema. El proceso se realizó en dos fases, primero se ejecutó:

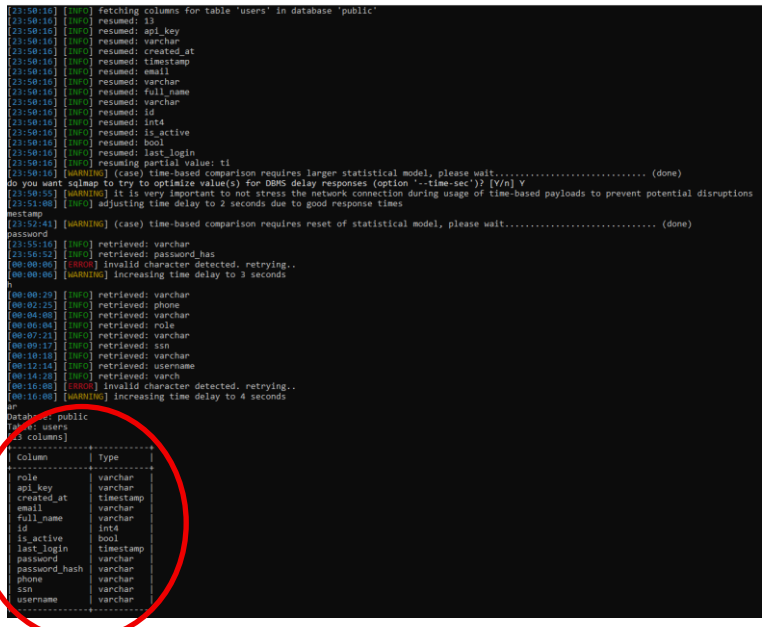
```
py sqlmap.py --url http://34.239.0.29:8080/login -D public -T users --columns --batch --forms
```

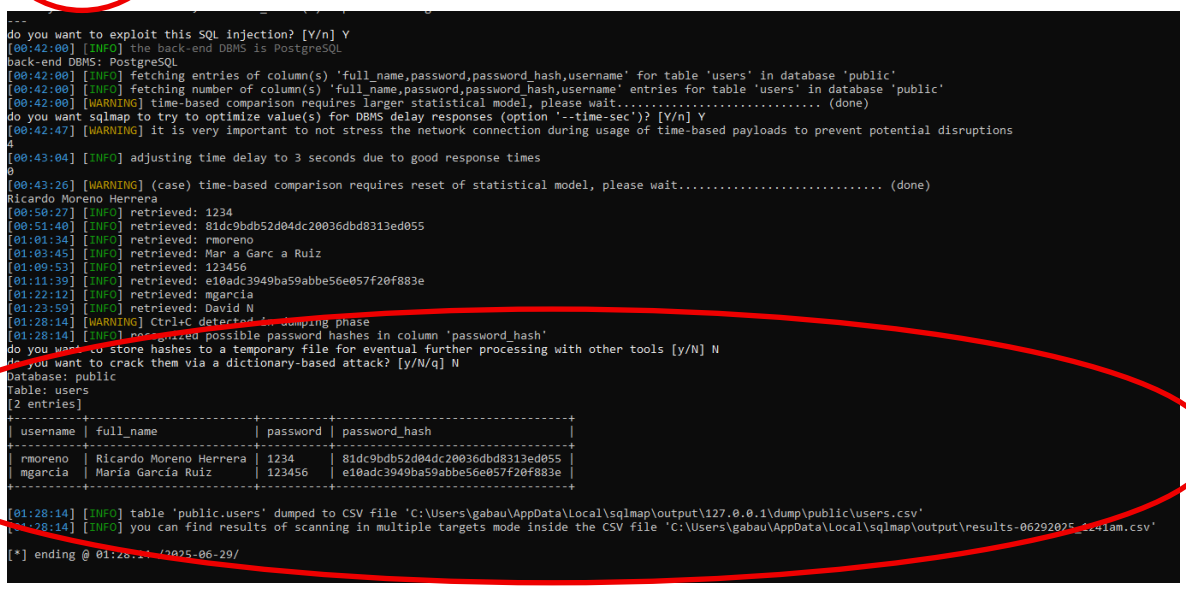
Con este comando enumeramos las columnas específicas de la tabla users, revelando la estructura de los datos almacenados. Una vez identificadas las columnas relevantes, se procedió con la extracción masiva de los datos mediante:

```
py sqlmap.py --url http://34.239.0.29:8080/ -D public -T users -C
"username,full_name,password,password_hash" --dump --batch --forms
```

Con este comando se extraen específicamente los campos más sensibles: nombres de usuario, nombres completos, contraseñas en texto plano y sus correspondientes hashes. Esta extracción representa una brecha de seguridad crítica, ya que compromete completamente la confidencialidad de las credenciales de los usuarios y permite el acceso no autorizado a sus cuentas.

Ilustración 61.SQLMap-Extracción de datos sensibles





Cuando se extraen los datos de la base de datos, estos se descargan en un fichero csv para poder tener de forma ordenada cada columna de la tabla extraida, en esta prueba solo se sacaron dos filas por temas de tiempos de ejecuación.

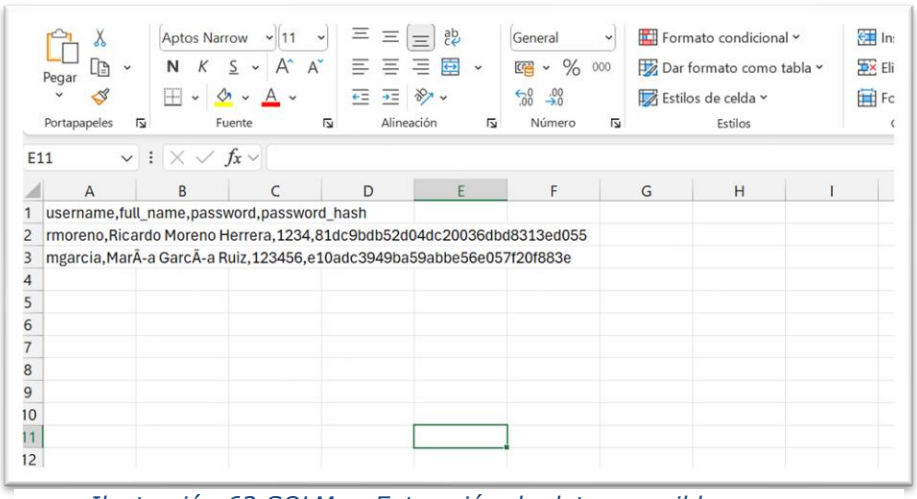


Ilustración 62.SQLMap-Extracción de datos sensibles

Como vemos lo ha nombrado exactamente igual que los registros de la tabla users.

4.4.4. Evaluación de contraseñas con Hashcat

En el ámbito de la seguridad de bases de datos, es una práctica estándar almacenar las contraseñas de los usuarios de forma cifrada en lugar de texto plano para proteger la confidencialidad en caso de una brecha de seguridad. Sin embargo, no todos los métodos de cifrado ofrecen el mismo nivel de protección, y la elección de un algoritmo de hash débil puede comprometer gravemente la seguridad del sistema. Las herramientas especializadas en crackeo de contraseñas como Hashcat pueden explotar estas debilidades para recuperar las contraseñas originales mediante ataques de fuerza bruta, diccionario o rainbow tables.

En el caso de la base de datos comprometida, se identificó que las contraseñas estaban protegidas mediante algoritmo de hash MD5. Este método funciona aplicando una función matemática unidireccional que convierte la contraseña original en una cadena alfanumérica de longitud fija de 32 caracteres hexadecimales. El proceso es irreversible teóricamente, ya que no existe una función inversa directa para recuperar el texto original. Sin embargo, MD5 presenta serias vulnerabilidades de

seguridad: es extremadamente rápido de calcular, lo que permite realizar millones de intentos por segundo, carece de salt (valor aleatorio añadido), es vulnerable a ataques de colisión, y existen extensas bases de datos con hashes MD5 precalculados que facilitan su descifrado.

4.4.4.1. Preparación de los hashes

Una vez extraídos los hashes MD5 de las contraseñas desde la base de datos comprometida, es necesario prepararlos adecuadamente para su procesamiento con Hashcat. Este paso implica crear un archivo de texto plano que contenga únicamente los valores hash, eliminando cualquier información adicional como nombres de usuario, metadatos o formato de base de datos que pudiera interferir con el proceso de crackeo. Los hashes extraídos se almacenan en un archivo denominado "hashes.txt", donde cada línea contiene un hash individual en formato hexadecimal de 32 caracteres. Esta organización permite que Hashcat pueda leer e interpretar correctamente los hashes objetivo, optimizando el proceso de comparación durante los ataques de diccionario, fuerza bruta o rainbow tables. La correcta preparación del archivo es fundamental para el éxito del proceso de crackeo, ya que cualquier carácter adicional o formato incorrecto puede causar errores en la herramienta o impedir que los hashes sean procesados adecuadamente.

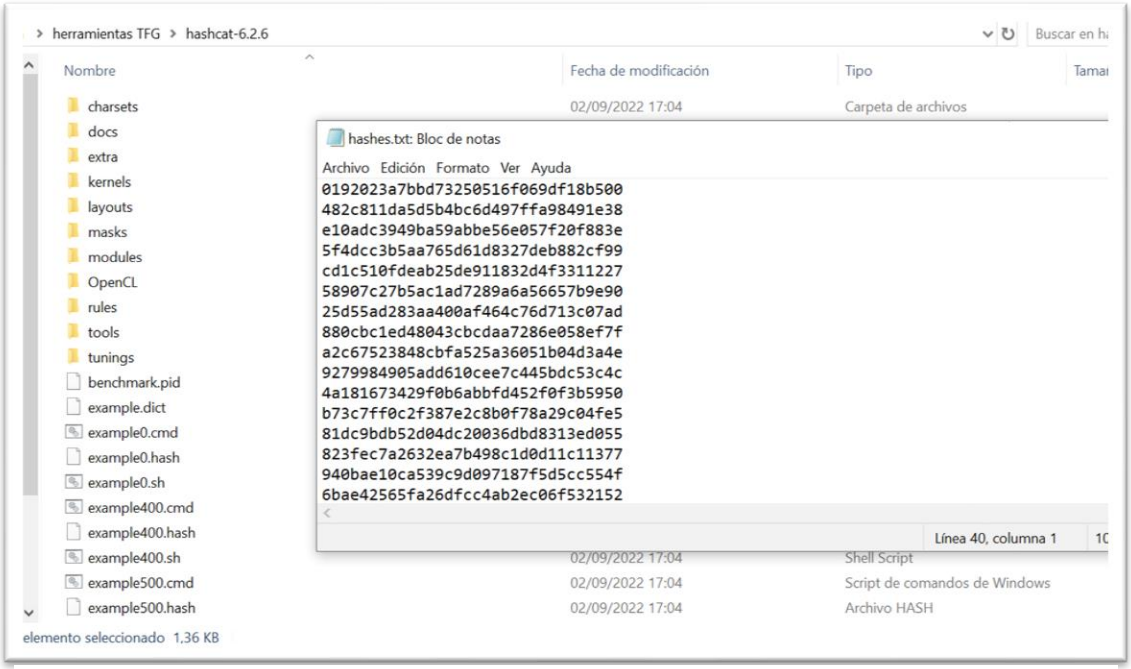


Ilustración 63.Hashes.txt

Una vez creado el fichero hashes.txt se guardará en el directorio del software de Hashcat.

4.4.4.2. Ataque de Diccionario

El ataque de diccionario constituye una de las técnicas más eficientes para el crackeo de contraseñas hash, especialmente cuando los usuarios tienden a utilizar contraseñas comunes o predecibles. Esta metodología se basa en comparar los hashes objetivo contra una lista precompilada de palabras frecuentemente utilizadas como contraseñas, incluyendo términos del diccionario, nombres propios, fechas comunes, y combinaciones típicas de caracteres. La efectividad de este enfoque radica en que la mayoría de los usuarios eligen contraseñas débiles por comodidad, utilizando palabras reconocibles en lugar de combinaciones aleatorias complejas. Al tratarse de hashes MD5, la velocidad de procesamiento es extremadamente alta, permitiendo que Hashcat evalúe millones de combinaciones por segundo hasta encontrar coincidencias.

Para este ataque se utilizará el diccionario "rockyou.txt", que es una de las listas de contraseñas más famosas y efectivas en el ámbito de la ciberseguridad. Este archivo contiene más de 14 millones de contraseñas reales obtenidas de la brecha de seguridad de la empresa RockYou en 2009, donde se comprometieron las credenciales de 32 millones de usuarios que estaban almacenadas en texto plano. La importancia de este diccionario radica en que representa patrones reales de comportamiento de usuarios, incluyendo las contraseñas más comúnmente utilizadas, variaciones típicas, y combinaciones predecibles, lo que lo convierte en una herramienta extremadamente efectiva para ataques de diccionario.

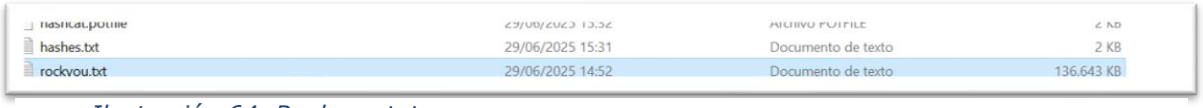


Ilustración 64. Rockyou.txt

Utilizando el siguiente comando se ejecuta el ataque de diccionario de los hashes objetivos y rockyou.txt:

```
hashcat -m 0 -a 0 hashes.txt rockyou.txt
```

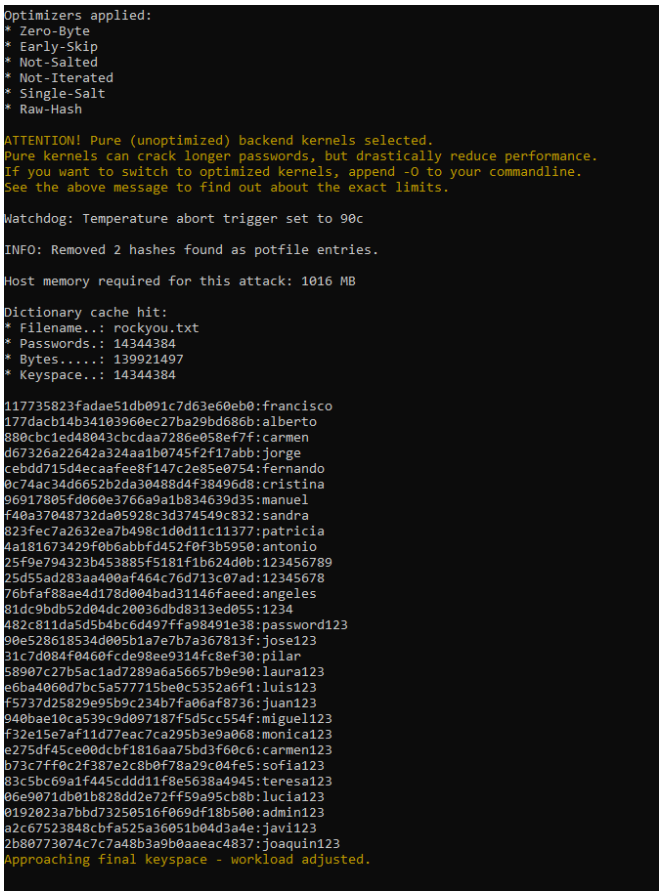


Ilustración 65. Hashcat-Ataque de diccionario

Como vemos en el resultado del comando, ha logrado desencriptar por ataque de diccionario 38 de los 40 hashes objetivos.

4.4.4.3. Ataque Hibrido

El ataque híbrido en hashcat combina un diccionario con patrones de caracteres adicionales. Hay dos modos: el modo 6 añade caracteres al final de palabras del diccionario (como "password" + "123" = "password123"), y el modo 7 los añade al principio. En este caso, hemos usado modo 6 con ?d?d?d, que toma cada palabra del rockyou.txt y prueba todas las combinaciones añadiendo tres dígitos del 000 al 999. Esto es más eficiente que la fuerza bruta porque aprovecha palabras reales que la gente usa como base, pero encuentra variaciones comunes como "admin123" o "maria456". Es perfecto para contraseñas que siguen el patrón típico de palabra común más números o símbolos simples.

Para realizar el ataque hibrido empleamos el siguiente comando en modo 6:

```
hashcat -m 0 -a 6 hashes.txt rockyou.txt ?d?d?d
```

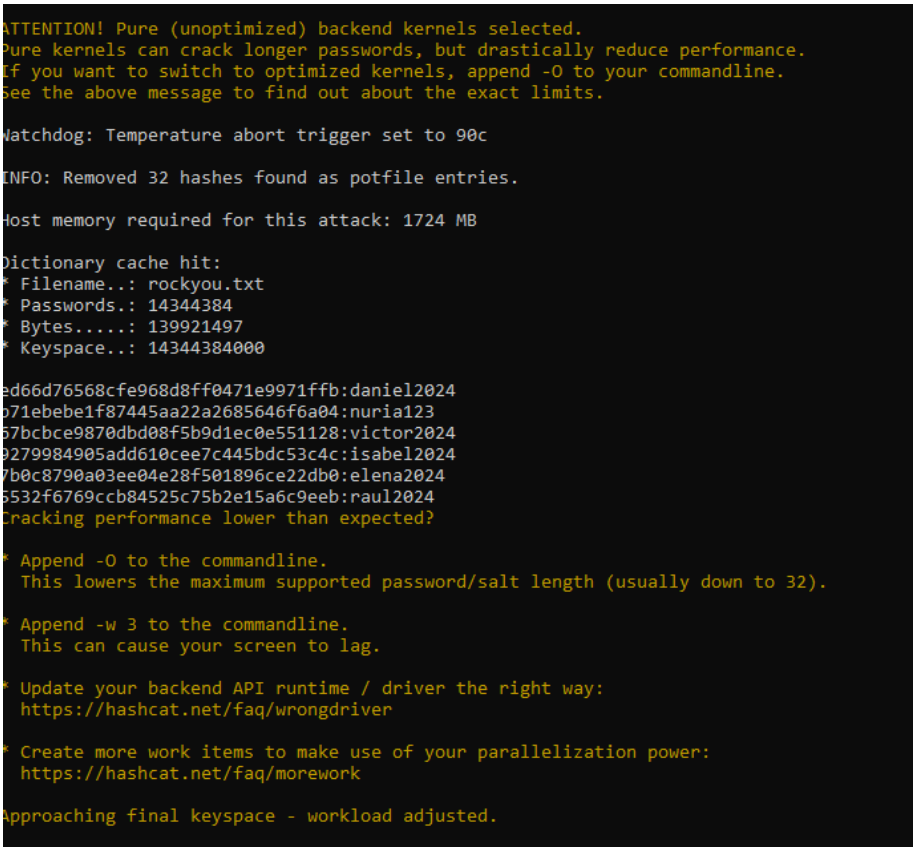


Ilustración 66. Hashcat-Ataque Híbrido

En este caso el ataque híbrido solo ha descifrado 6 claves hash ya que está diseñado para encontrar las contraseñas que son palabras del diccionario más números, como "password123" o "admin456". Las que quedan sin crackear probablemente tienen patrones más complejos que no siguen la fórmula simple de "palabra + 3 dígitos", como contraseñas completamente aleatorias o con caracteres especiales en medio.

4.4.4.4. Ataque de Fuerza Bruta

El ataque de fuerza bruta prueba todas las combinaciones posibles de caracteres de forma sistemática. Si especificas una máscara como ?l?l?l?l (4 letras minúsculas), probará desde "aaaa", "aaab", "aaac"... hasta "zzzz". No usa diccionarios ni patrones inteligentes, simplemente prueba todo.

En este trabajo de auditoría de contraseñas no se ha empleado el ataque de fuerza bruta debido a que este presenta una complejidad computacional exponencial que lo hace inviable para este estudio. Con un conjunto de caracteres estándar (letras, números y símbolos especiales), una contraseña de 8 caracteres genera aproximadamente 95^8 combinaciones posibles. Considerando que ya se logró un 95% de efectividad (38/40 hashes) mediante métodos de diccionario e híbridos, la relación coste-beneficio de implementar fuerza bruta para los 2 hashes restantes no se justifica desde una perspectiva de eficiencia computacional y consumo energético. Los recursos necesarios superarían significativamente los beneficios obtenidos en el contexto de esta investigación.

4.4.4.5. Resultados y Análisis de Rendimiento

Tabla 1. Resultados Desencriptación

Técnica	Modo Hashcat	Hashes Crackeados	Tiempo Ejecución	Eficiencia
Ataque de diccionario	-a 0	38/40	~30 segundos	Alta
Ataque Híbrido	-a 6	6/40	2-3 minutos	Media
Ataque de Fuerza Bruta	-a 3	N/A	Días/semanas	Muy Baja

4.5. IMPLEMENTACIÓN DE SOLUCIONES DE SEGURIDAD

4.5.1. Transferencia segura de datos mediante TLS/SSL

La seguridad en la transmisión de datos es esencial para proteger la información sensible que circula entre clientes y servidores, especialmente en entornos de nube donde la comunicación se realiza a través de redes públicas. Para garantizar la confidencialidad e integridad de estos datos, se utiliza el protocolo TLS (Transport Layer Security), sucesor del protocolo SSL, que establece un canal cifrado seguro entre el cliente y el servidor.TLS funciona mediante un proceso inicial llamado handshake, en el que cliente y servidor negocian los algoritmos criptográficos a utilizar, intercambian certificados digitales para autenticar la identidad del servidor (y opcionalmente del cliente), y acuerdan claves secretas para cifrar la comunicación. Durante este proceso se emplean técnicas de criptografía asimétrica (como RSA o ECC) para el intercambio seguro de claves y criptografía simétrica (como AES) para cifrar los datos transmitidos.

Una vez establecido el canal seguro, todos los paquetes que se envían entre cliente y servidor están cifrados, lo que impide que terceros intercepten, lean o modifiquen la información durante la transmisión. Además, TLS asegura la integridad de los datos mediante códigos de autenticación (MAC), y protege contra ataques de suplantación, garantizando que el usuario se conecta al servidor legítimo.

Esta sección presenta la implementación práctica de TLS/SSL para proteger la transferencia de datos en aplicaciones web, haciendo especial énfasis en la configuración de servidores proxy inversos, como Nginx, y el uso de certificados digitales.

4.5.1.1. Configuración TLS/SSL con Nginx

Para asegurar la transferencia de datos en una aplicación web desplegada en un entorno cloud, es común utilizar Nginx como servidor proxy inverso que gestione la terminación TLS/SSL. Esto facilita centralizar la seguridad y dejar que la aplicación backend se enfoque en la lógica, mientras Nginx se encarga del cifrado y descifrado de las comunicaciones.

El primer paso es instalar Nginx en la instancia. En Ubuntu, esto se realiza ejecutando el comando **sudo apt update** para actualizar la lista de paquetes, seguido de **sudo apt install nginx** para instalar el servidor. Una vez instalado, Nginx se ejecuta como un servicio que se puede iniciar, detener o reiniciar con comandos como **sudo systemctl start nginx** y **sudo systemctl enable nginx** para asegurarse de que arranque automáticamente al iniciar el sistema.

A continuación, es necesario generar un certificado SSL. Para pruebas, un certificado autofirmado es suficiente y puede crearse mediante OpenSSL con un comando como:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

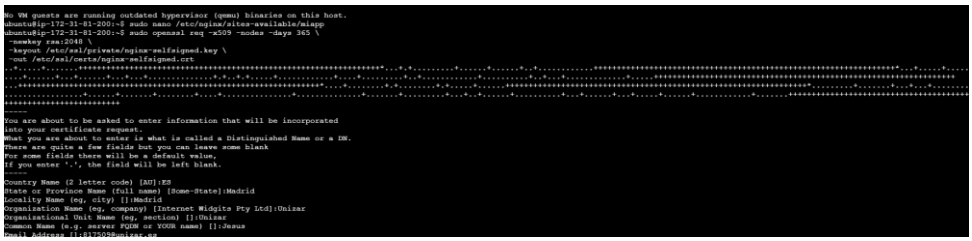


Ilustración 67. Certificados TLS

Este comando crea una clave privada y un certificado válido por un año, almacenados en las rutas típicas para certificados en Linux.

Luego se debe configurar Nginx para usar este certificado y activar TLS. El archivo de configuración habitual se encuentra en **/etc/nginx/sites-available/default**. En él, se añade un bloque para redirigir todo el tráfico HTTP (puerto 80) a HTTPS (puerto 443), y otro bloque que escucha en 443 configurado con los certificados, protocolos TLS modernos y cifrados seguros, además de realizar proxy inverso a la aplicación Flask que corre en el puerto 8080 de localhost. Para poder abrir el archivo de configuración debemos abrirlo como administradores con el siguiente comando:

```
sudo nano /etc/nginx/sites-available/default
```

```
server {
    listen 80 default_server;
    server_name _;

    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl default_server;
    server_name _;

    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers HIGH:!aNULL:!MD5;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Ilustración 68. configuración servidor Nginx

Finalmente, tras modificar la configuración, se verifica que no haya errores con sudo nginx -t y se reinicia el servicio con sudo systemctl reload nginx para aplicar los cambios.

Con esta configuración, todas las comunicaciones entre los clientes y el servidor estarán cifradas mediante TLS, mejorando la seguridad y protegiendo la transferencia de datos sensibles frente a interceptaciones o manipulaciones.

4.5.1.2. Protección de datos mediante tokenización

En el proyecto se implementó un sistema de tokenización para proteger los datos sensibles almacenados en la base de datos de usuarios. La tokenización es un método mediante el cual los datos originales, como correos electrónicos, números de teléfono, números de seguridad social, claves API e incluso el hash de la contraseña, se sustituyen por valores tokens generados aleatoriamente que no contienen información legible ni deducible. Esto significa que, aunque un atacante acceda a la base de datos tokenizada, no podrá obtener los datos reales sin acceso al sistema que mantiene el mapeo reversible.

Para lograr esta tokenización, se creó un mecanismo que genera tokens aleatorios de 16 caracteres compuestos por letras y números. Antes de crear un nuevo token, el sistema verifica en la tabla tokens_map si ya existe un token asignado para el valor original, lo que evita duplicidades y mantiene la consistencia. En caso de no existir, genera un token único comprobando que no se repita en la base de datos para asegurar la unicidad. Este mapeo reversible entre valores originales y tokens permite mantener la privacidad en la tabla principal users_tokenized, donde se almacenan los tokens en lugar de los datos originales, preservando la integridad referencial mediante el identificador de usuario. De esta manera, la aplicación puede operar sobre datos tokenizados, mientras que el sistema es capaz de traducirlos si se requiere, por ejemplo, para procesos internos seguros o auditorías.

La tokenización se realiza automáticamente para todos los usuarios mediante un proceso que recorre cada registro, aplicando la generación o reutilización de tokens para cada campo sensible. Para asegurar la integridad del proceso, se implementaron verificaciones posteriores que comparan el número de registros originales, tokenizados y el total de tokens creados, garantizando que no existan discrepancias.

4.5.1.3. Gestión segura de claves IAM

En el desarrollo del proyecto, la gestión segura de las claves IAM (Access Key ID y Secret Access Key) ha sido un aspecto crítico para garantizar la protección del acceso a los servicios de AWS utilizados, tales como RDS, S3 y DynamoDB. Estas credenciales permiten a la aplicación interactuar con los recursos en la nube, por lo que su manejo adecuado es esencial para prevenir accesos no autorizados y posibles brechas de seguridad. Para evitar la exposición de estas claves sensibles, se descartó el almacenamiento directo en el código fuente o en archivos de configuración sin protección. En su lugar, se adoptó el uso de variables de entorno dentro de la instancia EC2 donde se ejecuta la aplicación, lo que mantiene las credenciales fuera del código y evita su inclusión accidental en repositorios o sistemas de control de versiones.

Las variables de entorno son pares clave-valor almacenados en el sistema operativo que contienen información de configuración sensible o específica del entorno, como credenciales, rutas o configuraciones. Usarlas permite mantener estos datos fuera del código fuente, mejorando la seguridad y facilitando la gestión en distintos entornos (desarrollo, pruebas, producción).

En Python, las variables de entorno se pueden acceder mediante el módulo os usando os.getenv('NOMBRE_VARIABLE'). Para facilitar su manejo, especialmente en proyectos Flask, es común usar librerías como python-dotenv, que cargan automáticamente variables definidas en un archivo .env al entorno de ejecución. Esto permite separar la configuración sensible del código, evitando exponer datos como claves API o contraseñas. Una ventaja adicional de utilizar un archivo .env es que puede incluirse dentro de la estructura del proyecto (aunque nunca en repositorios públicos), lo que facilita la portabilidad: al mover la carpeta del proyecto a otro entorno, basta con mantener el archivo .env y asegurarse de que el sistema lo cargue correctamente, sin necesidad de redefinir manualmente las variables en el sistema operativo.

4.5.1.4. Autenticación Reforzada con MFA

Para proteger el acceso a los recursos críticos del entorno cloud, como las consolas de administración y los servicios de AWS (EC2, RDS, S3), se debe implementar un sistema de autenticación multifactor (MFA) en los usuarios IAM con privilegios elevados. El sistema MFA añade una capa adicional de seguridad al requerir, además del nombre de usuario y la contraseña, un código temporal de 6 dígitos generado por una aplicación autenticadora (como Microsoft Authenticator o Authy). Este código se renueva cada 30 segundos, por lo que incluso si las credenciales se vieran comprometidas, no sería posible acceder sin el segundo factor.

Esta medida refuerza significativamente la autenticidad del usuario, reduciendo el riesgo de accesos no autorizados y cumpliendo con buenas prácticas de ciberseguridad en entornos cloud. Para este proyecto, se configuró MFA en los usuarios IAM a través de la consola de AWS, utilizando dispositivos virtuales gratuitos y aplicaciones móviles seguras.

5. RESULTADOS

5.1. PRUEBAS Y RESULTADOS DE LA PROTECCIÓN EN LA TRANSMISIÓN DE DATOS

Al volver entrar al sitio web desde la ip pública, ahora nos marca que se está usando un medio de conexión HTTPS por medio de la ruta https://3.87.34.45/login:

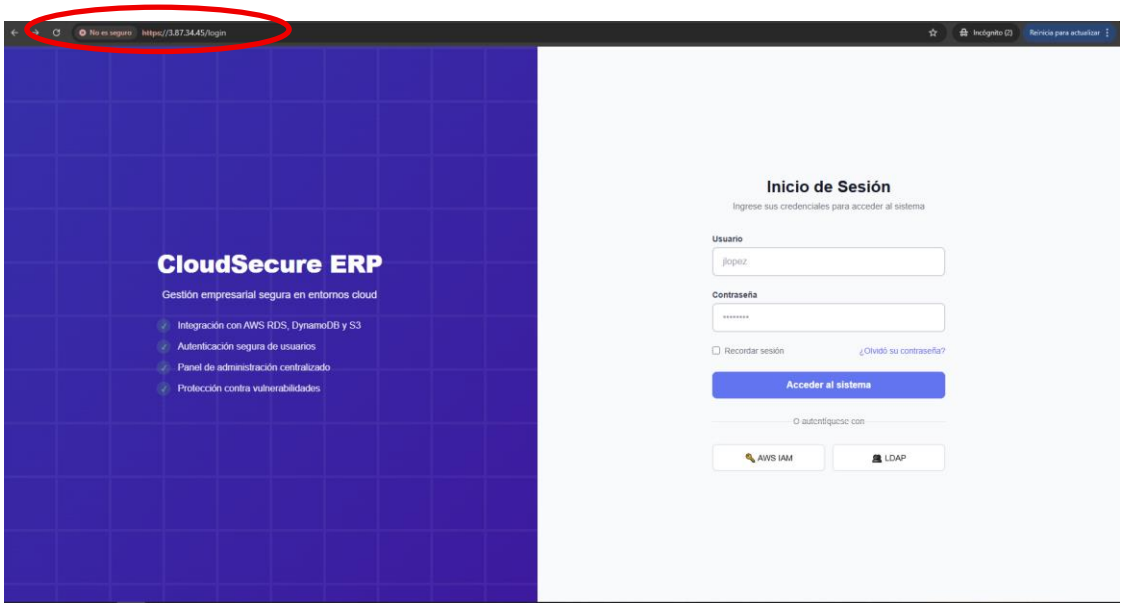


Ilustración 69. Transferencia Cifrada

Aunque el navegador pueda mostrar una advertencia indicando que el sitio no es seguro, debido a que se está utilizando un certificado SSL/TLS auto firmado y no emitido por una autoridad certificadora reconocida, la conexión sigue estando cifrada y protegida mediante el protocolo TLS.

Esto significa que, a pesar de la advertencia, la comunicación entre el cliente y el servidor se realiza a través de un canal cifrado que impide la interceptación o manipulación de los datos durante la transmisión. Por tanto, el objetivo principal de TLS —garantizar la confidencialidad e integridad de la información— se cumple plenamente.

Esto puede comprobarse utilizando Wireshark para capturar la transferencia de paquetes. Al analizar la comunicación, se observa que los datos están cifrados mediante el protocolo TLS, lo que garantiza que la información transmitida no pueda ser leída ni modificada por terceros:

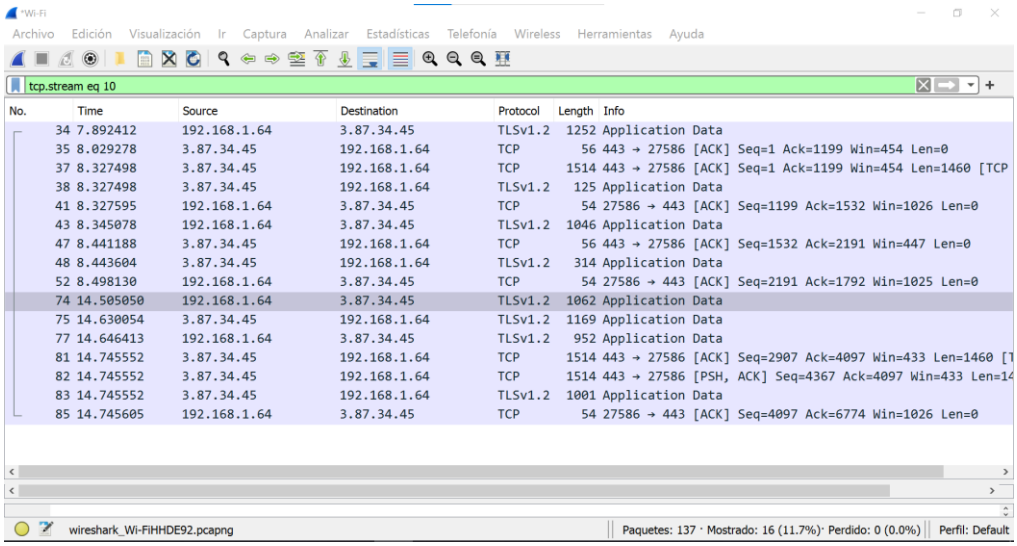


Ilustración 70. Transferencia Cifrada.

Si desplegamos el seguimiento TCP, comprobamos que toda la información está encriptada:

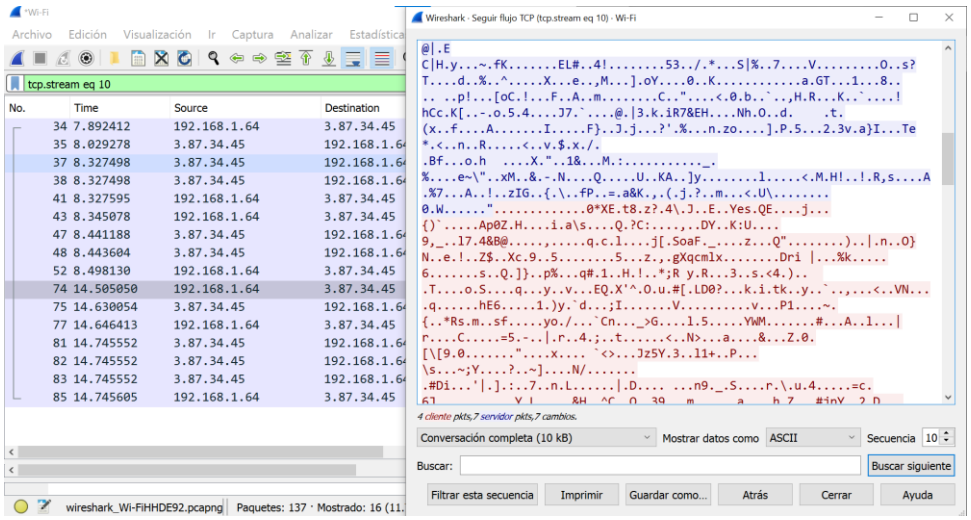


Ilustración 71.Transferencia Cifrada

En entornos de producción, para eliminar esta advertencia y ofrecer confianza a los usuarios, se recomienda utilizar certificados emitidos por autoridades certificadoras reconocidas, como Let's Encrypt, que validan la identidad del servidor y permiten a los navegadores confiar en la conexión.

5.2. VALIDACIÓN DE LA TOKENIZACIÓN EN LA BASE DE DATOS

se ha conseguido la tokenización de todos los campos de la tabla users para que de esta manera se pueda conservar la privacidad y anonimidad de los datos registrados y por medio de una herramienta automatizada y escalable:

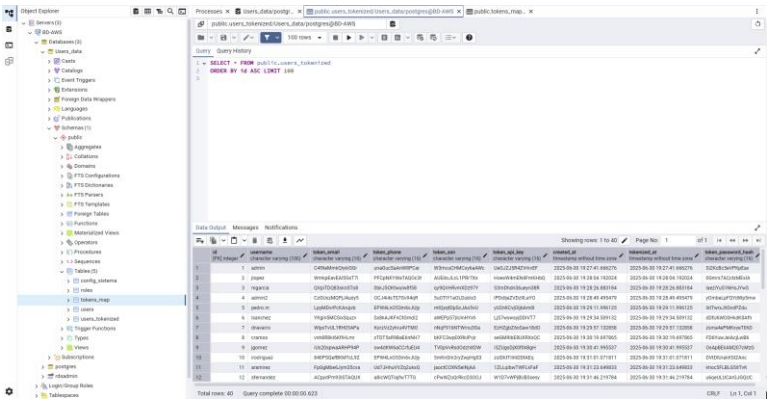


Ilustración 72.Users_tokenized

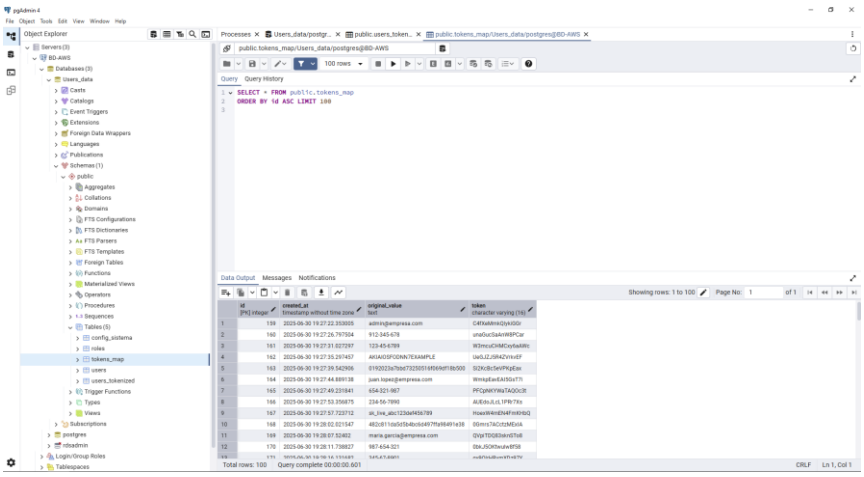


Ilustración 73.Tokens_map

De esta manera, aunque extrajeran la base de datos de los usuarios solo tendrían acceso a datos ilegibles que no podrían descifrar.

5.3. RESULTADOS DE AUTENTICACIÓN MFA

Ahora cada vez que se quiera hacer uso de un servicio de AWS, el sistema solicita una clave en constante cambio al propietario, de esta manera no se hará un mal uso de las herramientas por personal no autorizado.

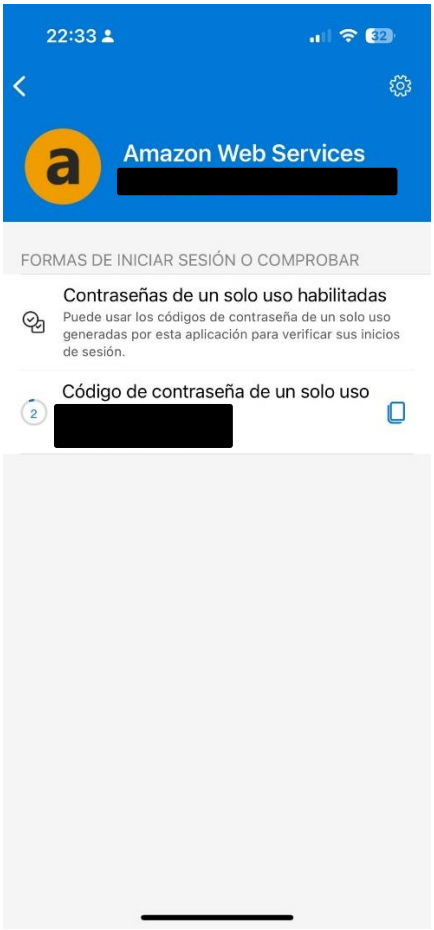


Ilustración 74. Autenticación MFA

6. CONCLUSIONES

El presente Trabajo de Fin de Grado ha abordado la problemática de la ciberseguridad de datos en entornos de computación en la nube, proporcionando un análisis exhaustivo de las vulnerabilidades, amenazas y soluciones de seguridad disponibles. A través del desarrollo de los diferentes capítulos, se ha logrado cumplir con los objetivos planteados inicialmente, obteniendo una visión integral de los desafíos que enfrentan las organizaciones al migrar sus datos a infraestructuras cloud. El objetivo principal de analizar las mejores prácticas de ciberseguridad para la protección de datos en la nube se ha alcanzado satisfactoriamente mediante el estudio detallado de herramientas de seguridad, estándares internacionales y casos de uso reales. La investigación ha demostrado que la implementación de medidas de seguridad multicapa, combinada con políticas de acceso granulares y cifrado de extremo a extremo, constituye la base fundamental para una estrategia de protección eficaz. Los objetivos específicos relacionados con la evaluación de vulnerabilidades comunes y el análisis de herramientas de monitorización han sido cumplidos a través del estudio comparativo de diferentes soluciones tecnológicas. Los resultados obtenidos evidencian que las configuraciones erróneas de servicios cloud y la gestión inadecuada de credenciales constituyen las principales causas de exposición de datos sensibles.

El desarrollo de esta investigación abre múltiples líneas de trabajo futuro que permitirían profundizar y expandir el conocimiento adquirido. Una de las áreas más prometedoras es la implementación de inteligencia artificial y machine learning para la detección proactiva de amenazas en tiempo real, desarrollando algoritmos capaces de identificar patrones anómalos de comportamiento antes de que se materialicen en incidentes de seguridad.

La evolución hacia arquitecturas zero-trust representa otra línea de investigación relevante, particularmente en el desarrollo de frameworks específicos para entornos multi-cloud que permitan mantener políticas de seguridad consistentes independientemente del proveedor de servicios. La implementación de blockchain como tecnología de soporte para la integridad y trazabilidad de datos en la nube presenta oportunidades interesantes para futuras investigaciones, particularmente en sectores regulados donde la inmutabilidad de registros es crítica. Finalmente, la evaluación del impacto de regulaciones emergentes en las estrategias de ciberseguridad cloud requiere análisis continuos que permitan a las organizaciones anticiparse a los requerimientos normativos futuros.

7. OBJETIVOS DE DESARROLLO SOSTENIBLE

Los objetivos de este Trabajo Fin de Grado están alineados con los siguientes Objetivos de Desarrollo Sostenible (ODS) y metas, de la Agenda 2030:

- Objetivo 4 - Garantizar una educación inclusiva y equitativa de calidad y promover oportunidades de aprendizaje permanente para todos
- Meta 4.4 De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento

- Objetivo 9 - Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación
- Meta 9.4 De aquí a 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales, y logrando que todos los países tomen medidas de acuerdo con sus capacidades respectivas

- Objetivo 11 - Lograr que las ciudades sean más inclusivas, seguras, resilientes y sostenibles
- Meta 11.c Proporcionar apoyo a los países menos adelantados, incluso mediante asistencia financiera y técnica, para que puedan construir edificios sostenibles y resilientes utilizando materiales locales

- Objetivo 16 - Promover sociedades justas, pacíficas e inclusivas
- Meta 16.a Fortalecer las instituciones nacionales pertinentes, incluso mediante la cooperación internacional, para crear a todos los niveles, particularmente en los países en desarrollo, la capacidad de prevenir la violencia y combatir el terrorismo y la delincuencia

8. BIBLIOGRAFÍA

Amazon Web Services. (2024). *AWS Well-Architected Framework – Security Pillar*. Obtenido de Amazon Web Services: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar

Amazon Web Services. (18 de 03 de 2024). *Protecting Data Using Encryption*. Obtenido de Amazon Web Services: https://docs.aws.amazon.com/whitepapers/latest/introduction-aws-security/protecting-data.html

Amazon Web Services. (27 de 06 de 2024). *Using SSL with a database instance*. Obtenido de Amazon Web Services: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html

Cloudflare, Inc. (24 de 03 de 2024). *What is HTTPS?* Obtenido de Cloudflare: https://www.cloudflare.com/learning/ssl/what-is-https/

National Institute of Standards and Technology. (10 de 07 de 2017). *Digital Identity Guidelines: Authentication and Lifecycle Management (SP 800-63B)*. Obtenido de NIST: https://pages.nist.gov/800-63-3/sp800-63b.html

Nmap Project. (27 de 06 de 2024). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Obtenido de Nmap: https://nmap.org/book/

OWASP Foundation. (2016 de 04 de 2023). *Cryptographic Storage Cheat Sheet*. Obtenido de OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html

OWASP Foundation. (2023). *Transport Layer Protection Cheat Sheet*. Obtenido de OWASP Cheat Sheet Series.

SQLMap Developers. (27 de 06 de 2024). *web: sqlmap – automatic SQL injection and database takeover tool*. Obtenido de SQLMap: https://sqlmap.org/

Relación de documentos

(X) Memoria 81 páginas
(X) Anexos NN páginas

La Almunia, a 30 de 06 de 2025

Firmado: Jesús Gabriel García Salomón