HACKEPS 2024

REPTE 1

j9G2FtnXLti6vA6KTwsHL3ttzrFju6NYx8:KQPSdwJeGgrjltHsnGZJFaeEzMrBNC

Aquesta es la contrasenya que haviem de colocar a la graella de password oculta. La part esquerra del : es la que hem trobat als metadatos

```
Nombre Archivo                  : kennedy.webp
Tamaño Archivo                  : 96 kB
Tipo Archivo                    : Extended WEBP
File Type Extension             : webp
MIME Type                       : image/webp
WebP Flags                      : EXIF
Ancho Imagen                    : 1280
Alto Imagen                     : 720
VP8 Version                     : 0 (bicubic reconstruction, normal loop)
Horizontal Scale                : 0
Vertical Scale                  : 0
Exif Byte Order                 : Big-endian (Motorola, MM)
Resolución Imagen Horizontal    : 72
Resolución Imagen Vertical      : 72
Unidad de Resolución de X e Y   : Pulgada
Posicionamiento Y y C           : Centrado
Versión Exif                    : 0232
Configuración de Componentes    : Y, Cb, Cr, -
Comentarios Usuario             : password=j9G2FtnXLti6vA6KTwsHL3ttzrFju6NYx8:code_delivered
Versión Flashpix Soportado      : 0100
Espacio Color                   : Sin calibrar
Tamaño de la Imagen             : 1280x720
Megapixels                      : 0.922
```

la part dreta és la contrasenya que se'ns havia donat al nostre grup, el 7.

Un cop escrit tot allò, mitjançant la inspecció i la network, hem trobat tot això a la informació del get de la network:

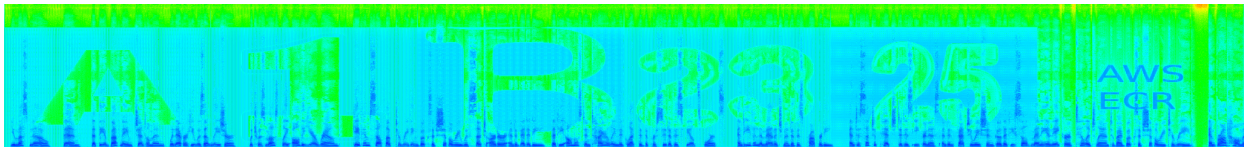

```
{
   "access_key": "AKIAYS2NWBZUZJ6CTN7S",
   "error": 0,
   "msg": "ok",
   "path": "b613e323-540b-4eb8-a79d-c381e37bd7b6",
   "secret_key": "iHvn0UlkRqTkZQs1Qrx9SJX1+9n67K4eTphq4XIK"
}
```


integrant el path a la url aconseguim passar de repte!

https://hackaton2024.useitapps.com/b613e323-540b-4eb8-a79d-c381e37bd7b6

REPTE 2

Per començar, al inspecionar la pantalla, en l'apartat de local Storage, hi trobem la url d'un audio, en copiar-la, ens envia a un audio, que era un fragment de don quijote, al descargarnos el àudio i convertirlo amb un espectrograma, hi trobarem aquestes lletres.



En aquesta imatge podem observar que tindrà alguna relació amb un contenidor de amazon(aws ecr).
Per autenticar-se a un repositori privat de **Amazon Elastic Container Registry (ECR)** i poder interactuar amb les imatges de contenidors, utilitzant Docker, es pot fer servir la següent comanda combinada de **AWS CLI** i **Docker**: aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 590184058473.dkr.ecr.us-west-2.amazonaws.com.
Seguidament busquem en la comanda **aws configure, que** és una part fonamental de la configuració inicial de la **AWS Command Line Interface (CLI)** i serveix per configurar les credencials i la configuració bàsica necessària per interactuar amb els serveis d'AWS, com el **Elastic Container Registry (ECR)**. Això és necessari perquè la CLI pugui autenticar les sol·licituds a AWS i permetre que les operacions es realitzin de manera segura.
Al configurar les credencials, ens demanarà una key, una secret-key, una regió, i un format.
La key i la secret key ens la donaven en el pas 1: "access_key": "AKIAYS2NWBZUZJ6CTN7S",
"secret_key": "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", la regió ens l'han donada com a pista (region: us-west-2), i amb format json que és una opció predeterminada i àmpliament utilitzada perquè permet una sortida estructurada i fàcil de llegir, que pot ser processada fàcilment per altres eines o scripts.

"aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 590184058473.dkr.ecr.us-west-2.amazonaws.com"
La comanda següent s'utilitza per autenticar-se a un registre de contenidors **Amazon Elastic Container Registry (ECR)** mitjançant Docker. Aquesta operació és necessària per interactuar amb repositoris privats d'imatges de contenidors allotjats en AWS.
Seguidament posem una nova comanda per a descarregar una imatge de Docker des d'un registre privat allotjat en **Amazon Elastic Container Registry (ECR)**: docker pull 590184058473.dkr.ecr.us-west-2.amazonaws.com/hackeps2024:latest
posteriorment per iniciarlo utilitzem: docker run -it 590184058473.dkr.ecr.us-west-2.amazonaws.com/hackeps2024:latest

Posteriorment posem: docker ps, per a trobar l'identificador de l'estat, i l'utilitzarem per executar-lo amb la comanda: docker exec -it dad79a17ece7 sudo /bin/bash
A l'executar-lo obtenim aquest arxiu "secret_uuid.gpg", al posar la contrasenya s'executave i ens donava el path per arribar al nivell 3.

REPTE 3

Mitjançant clicar al text se'ns va descarregar el .zip dels sonetos, amb l'ajuda de les pistes dels treballadors vam començar a contar les ciutats i les seves inicials. El sumatori de les ciutats concatenades anava a l'URL.

- Glasgow - G
- Quebec - Q
- Nairobi - N
- Florence - F
- Oslo - O
- Warsaw - W
- Tokyo - T
- New York - N
- Frankfurt - F
- Zurich - Z
- Kiev - K
- Cairo - C
- Quito - Q
- Ulaanbaatar - U
- Guangzhou - G
- Havana - H
- Fukuoka - F
- Quetta - Q
- Amsterdam - A
- Prague - P
- Munich - M
- Fes - F
- Lisbon - L
- Edinburgh - E
- Sydney - S
- Berlin - B
- Yokohama - Y
- Jakarta - J
- Rome - R
- Ottawa - O

- Vienna - V
- Vancouver - V
- Johannesburg - J
- Istanbul - I
- Xi'an - X
- Xiamen - X
- Helsinki - H
- Krakow - K
- Washington, D.C. - W
- Copenhagen - C
- Hyderabad - H
- Utrecht - U
- Qingdao - Q
- Rotterdam - R
- Edinburgh - E
- Dublin - D
- Madrid - M
- Athens - A
- Valencia - V
- Singapore - S

https://hackaton2024.useitapps.com/GQNFOWTNFZKCQUGHFQAPMFLESBYJROVVJIXXHKWCHUQREDMAVS

REPTE 4

Hem trobat que a les paraules taronges ens donen aws-kinesis, a través de la comanda

aws kinesis get-shard-iterator --stream-name "info-stream" --shard-id "shardId-000000000000" --shard-iterator-type TRIM_HORIZON
Aquesta comanda realitza una sol·licitud a AWS Kinesis per obtenir un **shard iterator** (iterador de fragment) d'un flux de Kinesis especificat. Aquest iterador és un "punter" que et permet llegir els registres dins d'un shard d'un flux de Kinesis.

Aquesta es la resposta, que és tot el flux de kinesis.
"ShardIterator":"AAAAAAAAAAGFM5P8Qgky141xOE7ivxxofZx6TF/mjRjsmv5kPLgphs5DGY9ux0S1czW2sAI9scHSL/lJTJpI3AK7dytm6e9L/eU6VrTpxXmtHsOjcNNMyKGWXNhEr7mgsZDEHNzcSlGsm8+W+WMDznk8dfV7+0z6lXmXR2ceQIjgVexWwR5zl/aC1l4Fsvs8cl0wcCPRf5ctQ8Wi9+4+a9zBI2NUMacPhP+gWIoc5TGvPxjhmWLjcQ=="

Seguidament posem aquesta comanda:

aws kinesis get-records --shard-iterator "AAAAAAAAAAGVt3tUXPs03nRerrWjsiX2RM0NhlTp2hlW4VsRzBQr0wp/2i1DvqPX9bFnKTbe+FP688qCsnqNvDmOjG9DHIFKyn7/Kwe6yexT/IljweSh/vi9zhhaa3weB2g/c0vLToA+I/nYgVjXjOYdOBXmCCDu0TcURIvhd+NMpWU+vUGlqHh/uUbsvxMa+fRUGYm9twB3Cq/f/1xybW7HS/ssK+Ddezv5OB5wX5yN+Bfni3UaEQ==" | Select-String -Pattern '"Data": "(.*?)"' | ForEach-Object { $_.Matches.Groups[1].Value } | Out-File "output.txt" Get-ChildItem -Recurse -Filter "output.txt"

Per a obtenir tot el fluxe pas per pas filtrat pel camp data a l'arxiu output.txt, en format base64, en descodificar-ho obtenim moltes localitzacions.

Necessitem cerca els punts al mapa mundi, o fem mitjançant un codi d'HTML. L'input és un arxiu .csv amb les dades lat lon de cada punt. Al mapa es veu la informació del punt, ja que ens donen de pista que el busquem està en una figura geomètrica, hem de cercar el punt del mig d'alguns punts del mapa que estiguin a prop uns dels altres. En aquest punt mitjà ens fiquem al Google Street View i veiem el que estàvem buscant, un la marca i el model d'un cotxe, Honda-Civic.
Però tenim una altra pista que és el xifrat cèsar, per tant, amb la web https://www.dcode.fr/cifrado-cesar i ens resulta en Ipoeb-Djwjd

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mapa de Coordenadas</title>
    <!-- Leaflet CSS -->
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
    <style>
        #map {
            height: 100vh; /* Altura del mapa para que ocupe toda la ventana */
        }
    </style>
</head>
<body>
    <h2>Sube tu archivo CSV</h2>
    <input type="file" id="fileInput" />
    <div id="map"></div>
```

```
    <!-- Leaflet JS -->
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.js"></script>
<!-- Librería para parsear CSV -->
    <script>
        // Crear el mapa centrado en el mundo
        const map = L.map('map').setView([0, 0], 2);

        // Añadir un mapa base de OpenStreetMap
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        }).addTo(map);

        // Función para manejar la carga del archivo CSV
        document.getElementById('fileInput').addEventListener('change', handleFileSelect,
false);

        function handleFileSelect(event) {
            const file = event.target.files[0];
            if (file) {
                // Usamos PapaParse para leer el archivo CSV
                Papa.parse(file, {
                    complete: function(results) {
                        const coordinates = parseCSVData(results.data);
                        addMarkersToMap(coordinates);
                        checkNearbyPoints(coordinates, 0.5); // Buscar puntos cercanos dentro
de 0.5 km
                    },
                    header: false // Asumimos que no hay cabecera en el CSV
                });
            }
        }

        // Función para parsear los datos del CSV
        function parseCSVData(data) {
            const coordinates = [];
            data.forEach(row => {
```

```
            if (row.length >= 2) { // Asegurarnos de que haya al menos dos columnas (latitud, longitud)
                const lat = parseFloat(row[0]);
                const lon = parseFloat(row[1]);
                if (!isNaN(lat) && !isNaN(lon)) {
                    coordinates.push([lat, lon]);
                }
            }
        });
        return coordinates;
    }

    // Función para agregar los marcadores al mapa
    function addMarkersToMap(coordinates) {
        coordinates.forEach(([lat, lon]) => {
            const marker = L.marker([lat, lon]).addTo(map);
            // Mostrar las coordenadas como etiqueta encima del marcador
            marker.bindTooltip(`Lat: ${lat.toFixed(5)}, Lon: ${lon.toFixed(5)}`, { permanent: true, direction: "top" });
        });
    }

    // Función para calcular la distancia entre dos puntos (Haversine formula)
    function calculateDistance(lat1, lon1, lat2, lon2) {
        const R = 6371; // Radio de la Tierra en kilómetros
        const dLat = toRad(lat2 - lat1);
        const dLon = toRad(lon2 - lon1);
        const a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
                  Math.cos(toRad(lat1)) * Math.cos(toRad(lat2)) *
                  Math.sin(dLon / 2) * Math.sin(dLon / 2);
        const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
        const distance = R * c; // Distancia en kilómetros
        return distance;
    }

    // Convertir grados a radianes
    function toRad(degrees) {
        return degrees * Math.PI / 180;
    }
```

```
        // Función para comprobar los puntos cercanos
        function checkNearbyPoints(coordinates, threshold) {
            for (let i = 0; i < coordinates.length; i++) {
                for (let j = i + 1; j < coordinates.length; j++) {
                    const lat1 = coordinates[i][0];
                    const lon1 = coordinates[i][1];
                    const lat2 = coordinates[j][0];
                    const lon2 = coordinates[j][1];
                    const distance = calculateDistance(lat1, lon1, lat2, lon2);

                    // Si la distancia es menor que el umbral, mostramos los puntos cercanos
                    if (distance < threshold) {
                        console.log(`Puntos cercanos: (${lat1}, ${lon1}) y (${lat2}, ${lon2}) -
Distancia: ${distance.toFixed(2)} km`);
                    }
                }
            }
        }
    </script>
</body>
</html>
```

https://hackaton2024.useitapps.com/Ipoeb-Djwjd

STEP 5:
Per al 5è pas, ens donaven una imatge, aquesta tenia un qr amagat, que es podia observar una mica editant la foto, amb la web https://georgeom.net/StegOnline/image obtenim el qr amb els colors invertits, en invertir-los per a poder llegir el qr obteniam el següent QR: arn:aws:s3:::accompliancenames.

```
 aws s3 ls s3://accompliancenames
                           PRE directories/
 aws s3 ls s3://accompliancenames/directories/
                           PRE 0a466f59-8bd7-47fe-aa78-dd989aca7462/
                           PRE 1d46f4e7-e926-49a4-9c1f-d36062c98817/
                           PRE 228d7786-cec7-426d-8598-622a201361a5/
                           PRE 24140aa8-a9f1-4c41-9f56-19007d86a1d3/
                           PRE 289eb9ca-8acf-4094-8902-44e0b0393703/
                           PRE 3d557a1d-1934-4313-ae8d-7d100ed10de9/
                           PRE 7f95bdc3-9079-4405-addb-9bd0faf41e97/
                           PRE 87fc30be-46f4-48b9-be4c-20082ccbfe29/
```

PRE 8cd79908-e916-401a-b3fd-3358c0034b00/
PRE a221d1be-8665-45a1-9949-d20c6b30b1df/
PRE e485fdbb-277b-4e88-916e-61fabd696cfd/

A partir d'obtindre això, hem optat per realitzar un script en python, on accedia a cada directori, i dins de cada directori, es trobava un .zip, també hem afegit al script que ho descarregui i que ho extragues en una carpeta extracted, amb això obtenim una carpeta "names" que conté 7919 elements.