Perfecto — vamos a hacerlo **todo** ahora mismo y paso a paso. He generado los archivos necesarios y te dejo:

* el **auth.log** simulado de **7000 líneas** (distribuido en una semana, con días con mucha/menos actividad para que se note la diferencia),
* el **CSV** ya parseado listo para Power BI,
* el **script Python** (parse\_auth\_log.py) **comentado línea por línea**,
* requirements.txt, .gitignore y README.md.

Puedes descargar los archivos aquí (ya generados en el entorno):

* Archivo de logs auth\_7000.log (7000 líneas):  
  📂 [Descargar auth\_7000.log](sandbox://mnt/data/cyber_log_monitor_files/auth_7000.log)
* CSV listo para Power BI (logs\_resultados\_powerbi\_7000.csv):  
  📂 [Descargar logs\_resultados\_powerbi\_7000.csv](sandbox://mnt/data/cyber_log_monitor_files/logs_resultados_powerbi_7000.csv)
* Script parseador comentado (parse\_auth\_log.py):  
  📂 [Descargar parse\_auth\_log.py](sandbox://mnt/data/cyber_log_monitor_files/parse_auth_log.py)
* requirements.txt:  
  📂 [Descargar requirements.txt](sandbox://mnt/data/cyber_log_monitor_files/requirements.txt)
* .gitignore:  
  📂 [Descargar .gitignore](sandbox://mnt/data/cyber_log_monitor_files/.gitignore)
* README.md con instrucciones y pasos Power Query:  
  📂 [Descargar README.md](sandbox://mnt/data/cyber_log_monitor_files/README.md)

A continuación tienes **todo el ejercicio paso a paso** (Windows / CMD), incluyendo el **código Python totalmente comentado**, los pasos en **Power Query** y las instrucciones para crear las visualizaciones solicitadas (barras por IP, barras por Usuario, tarjetas resumen y slicer por Timestamp).

**1) Estructura del proyecto (nombre exacto de la carpeta raíz)**

Crea esta carpeta en tu escritorio o donde prefieras:

Cyber Log Monitor - Parser de auth.log/

├── logs/ # aquí puedes poner auth\_7000.log (si lo descargas)

├── parse\_auth\_log.py # script para parsear (ya creado y comentado)

├── logs\_resultados\_powerbi\_7000.csv # CSV generado (puedes usarlo directo)

├── requirements.txt

├── .gitignore

└── README.md

Yo ya generé los archivos y los tienes disponibles en los enlaces arriba.

**2) Código Python (script parse\_auth\_log.py) — comentado línea por línea**

Copia/pega este script en parse\_auth\_log.py (ya está también disponible para descargar). Está **comentado en cada bloque y línea relevante**:

# parse\_auth\_log.py

# Script que parsea un archivo auth.log y genera logs\_resultados\_powerbi\_7000.csv (listo para Power BI).

# Comentarios en cada línea explican qué hace el script.

# Importamos módulos necesarios.

import re # re: expresiones regulares para parsear líneas del log

import pandas as pd # pandas: manejo de tablas (DataFrame y exportación CSV)

from datetime import datetime # datetime: manejo y creación de fechas/horas

import os # os: operaciones del sistema (comprobar existencia de archivos)

# Definición de rutas candidatas donde buscar el auth.log.

# Prioriza: logs/auth.log, luego auth.log en la raíz, luego auth\_7000.log (por si lo tienes así).

LOG\_PATHS = ["logs/auth.log", "auth.log", "auth\_7000.log"]

# Función que devuelve la primera ruta existente donde se encuentra el auth.log

def encontrar\_log():

for p in LOG\_PATHS: # iterar por las rutas candidatas

if os.path.isfile(p): # si la ruta existe y es un archivo

return p # devolver esa ruta

return None # si no existe ninguna ruta, devolver None

# Llamamos a la función para localizar el archivo

log\_file = encontrar\_log() # ruta encontrada o None

# Si no se encuentra el log, informamos y salimos

if not log\_file:

print("❌ No se encontró 'logs/auth.log' ni 'auth.log' ni 'auth\_7000.log'. Coloca el archivo en la carpeta 'logs/' o en la raíz del proyecto.")

raise SystemExit(1) # salir del script con código 1

# Expresión regular para capturar las partes relevantes de las líneas de auth.log (sshd).

# Captura: Mes (Sep), Día, Hora, Usuario, IP, Puerto.

pattern = re.compile(

r"(?P<Mes>\w{3})\s+" # captura mes abreviado (ej. 'Sep')

r"(?P<Dia>\d{1,2})\s+" # captura día (1 o 2 dígitos)

r"(?P<Hora>\d{2}:\d{2}:\d{2}).\*" # captura hora HH:MM:SS

r"Failed password for (?:invalid user )?"# coincide con "invalid user " opcional

r"(?P<Usuario>[\w\.\-]+)\s+from\s+" # captura el nombre de usuario (letras, dígitos, . o -)

r"(?P<IP>[\d\.]+)\s+port\s+" # captura la IP (IPv4)

r"(?P<Puerto>\d+)", # captura el puerto (números)

re.IGNORECASE # case-insensitive para mayor robustez

)

# Mapa para traducir meses abreviados en inglés a número de mes.

MES\_MAP = {

"Jan":1,"Feb":2,"Mar":3,"Apr":4,"May":5,"Jun":6,

"Jul":7,"Aug":8,"Sep":9,"Oct":10,"Nov":11,"Dec":12

}

# Lista que irá guardando filas parseadas como diccionarios.

rows = []

# Abrimos el archivo y lo recorremos línea a línea.

with open(log\_file, "r", encoding="utf-8", errors="ignore") as f:

for line in f: # por cada línea del archivo

m = pattern.search(line) # aplicamos la regex

if not m:

continue # si la línea no coincide, saltamos

# Extraemos los campos capturados por la regex

mes = m.group("Mes") # mes abreviado, ej. 'Sep'

dia = int(m.group("Dia")) # día como entero

hora\_str = m.group("Hora") # hora 'HH:MM:SS'

usuario = m.group("Usuario") # usuario detectado

ip = m.group("IP") # IP detectada

puerto = int(m.group("Puerto")) # puerto (int)

resultado = "Failed" # resultado (en este proyecto, fallos)

# Construimos una fecha completa (añadiendo el año actual).

# Si tu log contiene otro año, ajusta aquí.

try:

month = MES\_MAP.get(mes.capitalize(), 1) # convertir 'Sep'->9

h, mi, s = map(int, hora\_str.split(":")) # separar hora

year = datetime.now().year # usar año actual

dt = datetime(year, month, dia, h, mi, s)# datetime completo

fecha\_iso = dt.date().isoformat() # 'YYYY-MM-DD'

hora\_iso = dt.time().isoformat() # 'HH:MM:SS'

timestamp = dt.isoformat(sep=" ") # 'YYYY-MM-DD HH:MM:SS'

except Exception:

# En caso de error con la fecha, guardamos valores crudos

fecha\_iso = f"{mes} {dia}"

hora\_iso = hora\_str

timestamp = f"{mes} {dia} {hora\_str}"

# Añadimos la fila a la lista de resultados

rows.append({

"Fecha": fecha\_iso, # fecha en ISO o cruda

"Hora": hora\_iso, # hora

"Timestamp": timestamp, # timestamp combinado

"Usuario": usuario, # usuario

"IP": ip, # IP origen

"Puerto": puerto, # puerto origen

"Resultado": resultado # resultado (Failed)

})

# Convertimos la lista a un DataFrame de pandas con columnas ordenadas.

df = pd.DataFrame(rows, columns=["Fecha","Hora","Timestamp","Usuario","IP","Puerto","Resultado"])

# Nombre del CSV de salida (listo para importar en Power BI).

out\_csv = "logs\_resultados\_powerbi\_7000.csv"

# Guardamos el CSV con encoding utf-8-sig para compatibilidad con Excel/Power BI.

df.to\_csv(out\_csv, index=False, encoding="utf-8-sig")

# Mostramos un pequeño resumen por consola (número de filas y top IPs/usuarios).

print(f"✅ Procesado: {len(df)} filas. CSV generado -> {out\_csv}")

if not df.empty:

print("\nTop 10 IPs (más intentos):")

print(df['IP'].value\_counts().head(10))

print("\nTop 10 Usuarios (más atacados):")

print(df['Usuario'].value\_counts().head(10))

**3) Cómo ejecutar el script (Windows — CMD) — pasos prácticos**

1. Abre **CMD** y sitúate en la carpeta del proyecto:
2. cd "C:\ruta\a\Cyber Log Monitor - Parser de auth.log"
3. (Opcional pero recomendado) crea y activa un entorno virtual:
4. python -m venv venv
5. venv\Scripts\activate

Si PowerShell bloquea scripts, usa **CMD** como aquí.

1. Instala dependencias:
2. pip install -r requirements.txt

(mi requirements.txt solo contiene pandas).

1. Ejecuta el parser (si quieres procesar localmente tu auth.log):
2. python parse\_auth\_log.py

Esto generará logs\_resultados\_powerbi\_7000.csv en la carpeta actual.

**4) Power BI — Power Query (preparar datos y crear Timestamp) — pasos detallados**

1. Abre **Power BI Desktop** → **Inicio → Obtener datos → Texto/CSV** → selecciona logs\_resultados\_powerbi\_7000.csv → clic **Transformar datos**.
2. En el **Editor de Power Query**:
   * Verifica tipos: Fecha y Hora suelen aparecer como **Texto** (está bien inicialmente).
   * **Crear columna Timestamp**:
     + En **Agregar columna → Columna personalizada** pega esta fórmula:
     + = DateTime.FromText(Text.Trim([Fecha]) & " " & Text.Trim([Hora]))
     + Nombra la nueva columna Timestamp.
     + Cambia el **Tipo de datos** de Timestamp a **Fecha/Hora** (Date/Time).
     + Si Power Query no convierte, prueba:
     + = DateTime.FromText(Text.PadStart(Text.From(Date.Year(DateTime.LocalNow())),4,"0") & "-" & Text.Trim([Fecha]) & " " & Text.Trim([Hora]))

(normalmente no es necesario con nuestro CSV).

* + Asegura tipos:
    - IP → Texto
    - Usuario → Texto
    - Puerto → Número entero
    - Resultado → Texto
  + (Opcional) Crear HoraEntera para analizar por hora:
    - **Agregar columna → Columna personalizada**:
    - = Date.Hour([Timestamp])
    - Cambia tipo a Número entero.
  + Haz **Cerrar y aplicar** (Close & Apply).

**5) Power BI — Visualizaciones solicitadas**

Después de aplicar los cambios:

**5.1 Medidas DAX recomendadas (Modelado → Nueva medida)**

Asumiendo que tu tabla se llama logs\_resultados\_powerbi\_7000 (ajusta si Power BI le puso otro nombre):

TotalIntentos = COUNTROWS('logs\_resultados\_powerbi\_7000')

IPsUnicas = DISTINCTCOUNT('logs\_resultados\_powerbi\_7000'[IP])

UsuariosUnicos = DISTINCTCOUNT('logs\_resultados\_powerbi\_7000'[Usuario])

**5.2 Gráfico de barras — IP vs recuento**

* Inserta un **Gráfico de barras**.
* Campos:
  + **Eje (Axis)**: IP
  + **Valores (Values)**: IP → selecciona **Count** (conteo)
* Opcional: en **Filtros del visual** aplica Top N → Top 10 por Count of IP.
* Formato: activar **Etiquetas de datos**, título “IPs con más intentos”.

**5.3 Gráfico de barras — Usuario vs recuento**

* Inserta otro **Gráfico de barras**.
* Campos:
  + **Eje**: Usuario
  + **Valores**: Usuario → **Count**
* Ordenar descendente y aplicar Top N si quieres.

**5.4 Tarjetas resumen**

* Inserta tres **Cards**:
  + Card 1: arrastra TotalIntentos → mostrará total de intentos.
  + Card 2: arrastra IPsUnicas.
  + Card 3: arrastra UsuariosUnicos.
* Formatea títulos y tamaño.

**5.5 Slicer por Timestamp (rango)**

* Inserta un **Slicer**.
* Arrastra Timestamp al Slicer.
* Configura tipo **Between** (rango).
* Al seleccionar un rango, todos los visuales se actualizan mostrando sólo los datos dentro del intervalo.

**5.6 Slicer por hora del día (opcional)**

* Inserta otro **Slicer**.
* Arrastra HoraEntera (si la creaste) para filtrar por ventanas 0–23h.

**6) Notas sobre los datos generados (cómo fueron distribuidos)**

* Generé 7000 líneas distribuidas en 7 días (del **13/09/2025** al **19/09/2025**) con la siguiente distribución por día (ejemplo para que se note variación):
  + Día 1: 50
  + Día 2: 300
  + Día 3: 800
  + Día 4: 2000 ← **pico** del ataque
  + Día 5: 1700
  + Día 6: 900
  + Día 7: 1250
* Esto te permitirá ver diferencias claras día a día en Power BI y practicar slicers/aggregations.

**7) requirements.txt y .gitignore**

**requirements.txt**

pandas

**.gitignore** (sugerido)

# Python cache

\_\_pycache\_\_/

\*.py[cod]

# Virtual env

venv/

.env

.env/

# Logs and outputs

\*.log

\*.csv

\*.xlsx

logs/

# IDEs

.vscode/

.idea/

# System files

Thumbs.db

Desktop.ini

$RECYCLE.BIN/

(ambos archivos ya están generados y listos para descargar en los enlaces de arriba).

**8) README.md listo para GitHub**

He generado un README.md que incluye descripción del proyecto, instrucciones de uso, pasos Power Query y sugerencias de visuales. Puedes descargarlo desde el enlace arriba y editar tu nombre/autores.

**9) ¿Qué hago yo ahora?**

* Ya generé los archivos y el CSV. Puedes:
  + Descargar auth\_7000.log y logs\_resultados\_powerbi\_7000.csv y abrirlos en Power BI.
  + Ejecutar localmente parse\_auth\_log.py si prefieres reprocesar tu propio auth.log.
* Si quieres, puedo hacer cualquiera de estas tareas adicionales **ahora mismo**:
  + A) Generar un archivo Excel (.xlsx) además del CSV.
  + B) Crear un README con capturas de pantalla de ejemplo (generadas en este entorno) para que lo pegues en GitHub.
  + C) Preparar un **paquete .zip** con todos los archivos listo para descargar.
  + D) Generar un ejemplo de **filtro avanzado** o agregar **alertas** (p. ej. IPs con > N intentos) en el script.

Dime la letra (A/B/C/D) si quieres que lo haga ahora.

Si quieres, también te guío en tiempo real por Power BI mientras haces los pasos (me das un pantallazo o me vas diciendo qué ves y te voy corrigiendo). ¿Empezamos por descargar los archivos o prefieres que primero te entregue el CSV en un formato Excel?