Práctica 1 Sistemas de Información

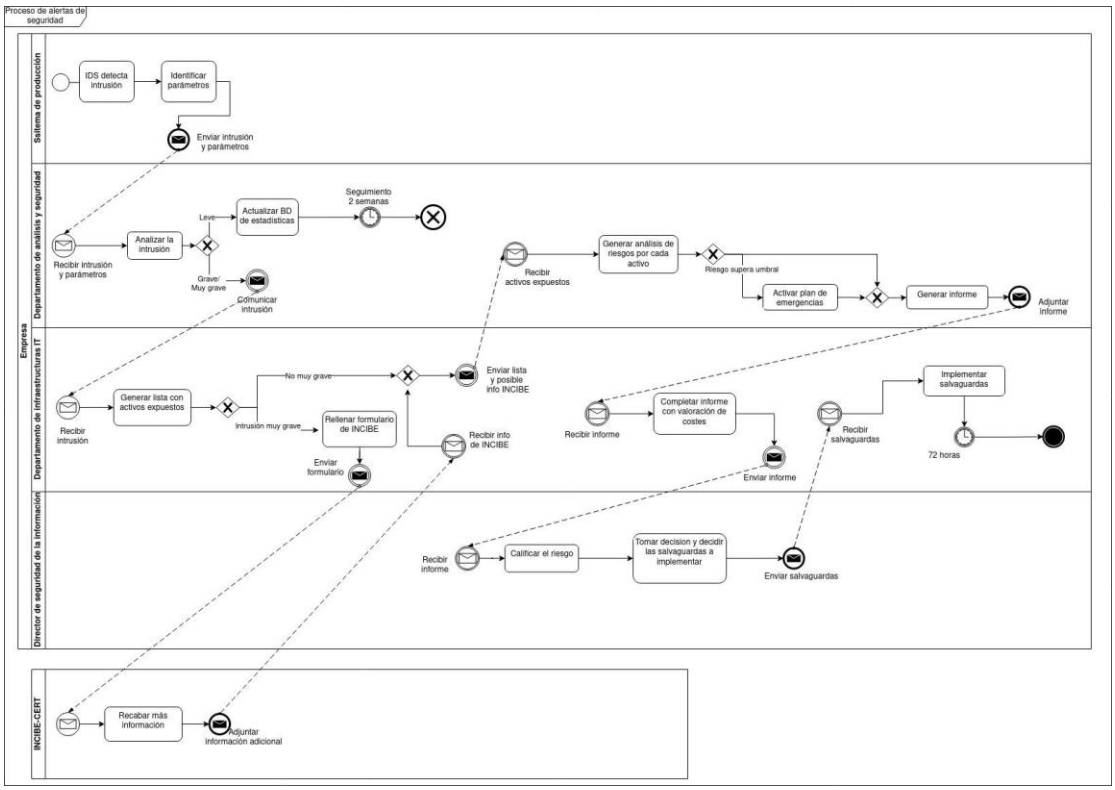
RECOPILACIÓN, ESTRUCTURACIÓN Y ANÁLISIS DE DATOS

David Billhardt Desdentado | Diego Palacios Ortega | Javier Rufo Cotobal

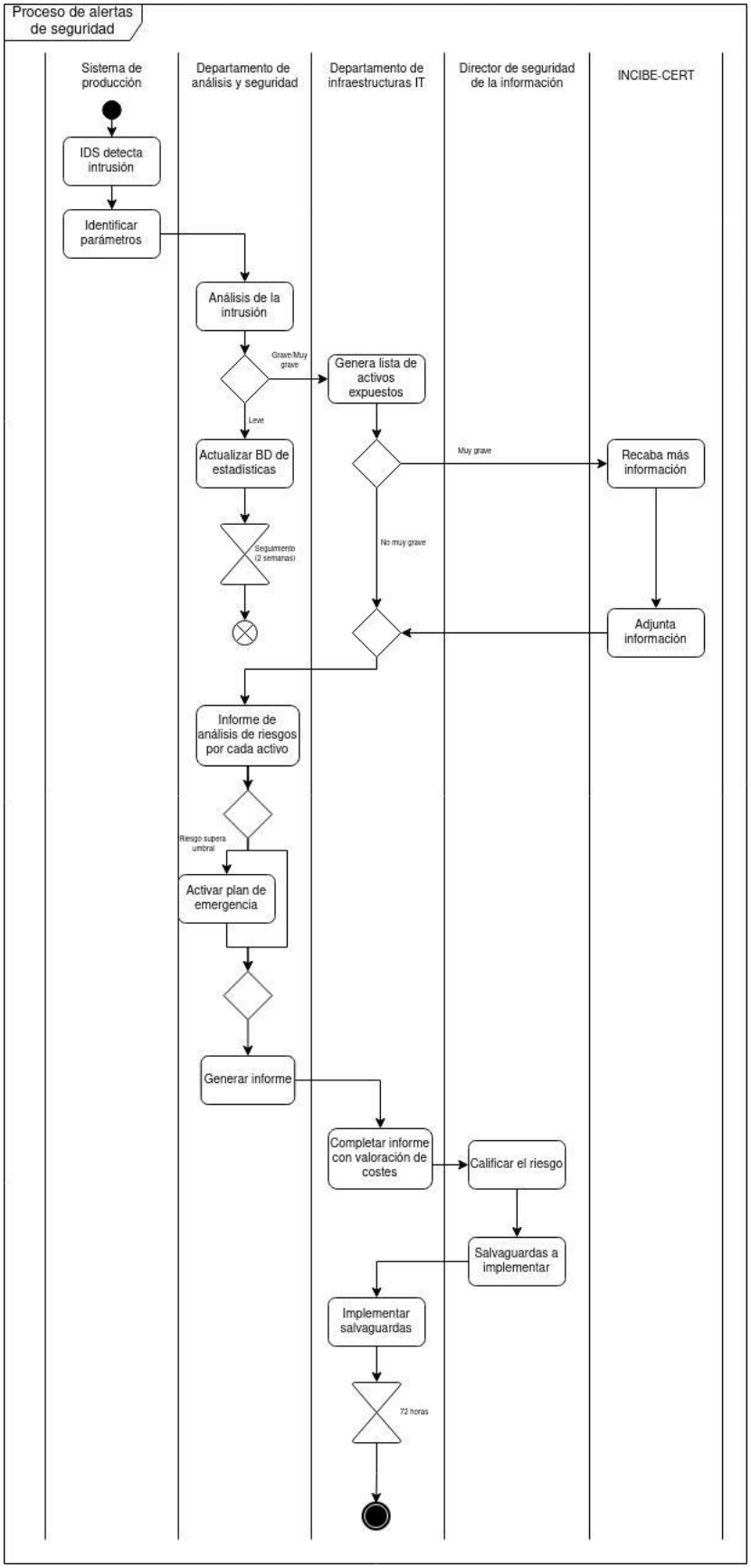
Ejercicio 1: modelos de proceso

Esquemas realizados gracias a https://app.diagrams.net/

ESQUEMA BPMN



ESQUEMA UML


Proceso de alertas de seguridad
Sistema de producción
Departamento de análisis y seguridad
Departamento de infraestructuras IT
Director de seguridad de la información
INCIBE-CERT
IDS detecta intrusión
Identificar parámetros
Análisis de la intrusión
Grave/Muy grave
Genera lista de activos expuestos
Leve
Actualizar BD de estadísticas
Muy grave
Recaba más información
Seguimiento (2 semanas)
No muy grave
Adjunta información
Informe de análisis de riesgos por cada activo
Riesgo supera umbral
Activar plan de emergencia
Generar informe
Completar informe con valoración de costes
Calificar el riesgo
Salvaguardas a implementar
Implementar salvaguardas
72 horas

Ejercicio 2: sistema ETL

Dado que todo se encuentra en un único script, es en esta parte donde se crean también las tablas y la conexión con la base de datos:

```
conex = sqlite3.connect("Base-Datos.db")
cursor = conex.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS alerts(id INTEGER PRIMARY KEY,
timestamp TEXT, sid INTEGER, msg TEXT, clasificacion TEXT, prioridad
INTEGER, protocolo TEXT, origen TEXT, destino TEXT, puerto INTEGER)")
cursor.execute("CREATE TABLE IF NOT EXISTS responsable (nombre TEXT PRIMARY
KEY, telefono INTEGER, rol TEXT)")
cursor.execute("CREATE TABLE IF NOT EXISTS analisis (id TEXT PRIMARY KEY,
puertos_abiertos TEXT, n_puertos INTEGER, servicios INTEGER,
servicios_inseguros INTEGER, vulnerabilidades_detectadas INTEGER)")
cursor.execute("CREATE TABLE IF NOT EXISTS devices (id TEXT PRIMARY KEY, ip
TEXT, localizacion TEXT, responsable_id TEXT, analisis_id TEXT, FOREIGN KEY
(responsable_id) REFERENCES responsable(nombre), FOREIGN KEY (analisis_id)
REFERENCES analisis(id))")

with open ("datos_22_23/devices.json") as devicesJ:
    devices_json = json.load(devicesJ)
```

Se puede observar que hemos creado una tabla para alertas, otra para responsable, otra para análisis, y por último la de dispositivos, donde hemos cargado el JSON. Es importante mencionar que en la tabla análisis hemos añadido el campo "n_puertos" para poder saber de una manera más directa este valor. Además, al dividir los datos que aparecen en devices.json en tres tablas distintas, hemos definido una serie de claves foráneas que nos van a ayudar a poder relacionar los datos de estas tres tablas. En particular, en la tabla devices tenemos "responsable_id" como clave foránea que proviene del campo nombre de la tabla responsable. Lo mismo ocurre con el campo "analisis_id" que es una clave foránea proveniente del campo id de la tabla análisis. A su vez, en la tabla análisis hemos agregado otro campo llamado "id" que va a ser el que utilicemos como clave primaria. A la hora de ir insertando valores, vamos a ir incrementando un contador y ese valor será el que tomará el id.

```
contador = 0
for it in devices_json:
    puertos_abiertos_raw = it['analisis']['puertos_abiertos']
    n_puertos = len(puertos_abiertos_raw)
    puertos_abiertos = ""
    if puertos_abiertos_raw == "None":
        puertos_abiertos = puertos_abiertos_raw
        n_puertos = 0
    else:
        for i in range(n_puertos):
            puertos_abiertos += str(puertos_abiertos_raw[i])
            if i != n_puertos - 1:
                puertos_abiertos += '|'
    servicios = it['analisis']['servicios']
    servicios_inseguros = it['analisis']['servicios_inseguros']
    vulnerabilidades_detectadas = it['analisis']['vulnerabilidades_detectadas']
    cursor.execute("INSERT OR IGNORE INTO analisis (id, puertos_abiertos,
n_puertos, servicios, servicios_inseguros, vulnerabilidades_detectadas) VALUES (?,
?, ?, ?, ?, ?)", (contador, puertos_abiertos, n_puertos, servicios,
servicios_inseguros, vulnerabilidades_detectadas))
```

```
    nombre_responsable = it['responsable']['nombre']
    telefono_responsable = it['responsable']['telefono']
    rol_responsable = it['responsable']['rol']
    cursor.execute("INSERT OR IGNORE INTO responsable (nombre, telefono, rol)
VALUES (?, ?, ?)", (nombre_responsable, telefono_responsable, rol_responsable))

    device_id = it['id']
    device_ip = it['ip']
    device_localizacion = it['localizacion']

    cursor.execute("INSERT OR IGNORE INTO devices (id, ip, localizacion,
responsable_id, analisis_id) VALUES (?, ?, ?, ?, ?)", (device_id, device_ip,
device_localizacion, nombre_responsable, contador))

    contador += 1
```

En la parte de inserción hemos tenido que tratar el array que aparece en el campo "puertos abiertos" del archivo devices.json ya que, no podemos insertar un array. Por tanto, hemos contado el número de puertos que hay y hemos guardado los puertos como string, separados por el símbolo "|". De esta manera, si tenemos que tratar con ellos, solo tendremos obtener el valor de "n_puerto" e ir recorriendo la string tras hacerle un split por el símbolo indicado.

Una vez esto se ha establecido, procedemos a calcular los valores establecidos, de manera que tras la ejecución del código que se encarga de imprimir los parámetros de interés...

```
contador = 0
alerts_csv = pd.read_csv("datos_22_23/alerts.csv")
alerts_csv.to_sql('alerts', conex, if_exists='replace', index=True,
index_label="id")

dispositivos = pd.read_sql_query("SELECT * from devices", conex)
print(f"Numero de dispositivos: {len(dispositivos)}")
dispositivos = pd.read_sql_query("SELECT d.*, a.puertos_abiertos, a.n_puertos,
a.servicios, a.servicios_inseguros, a.vulnerabilidades_detectadas, r.telefono,
r.rol FROM devices AS d JOIN analisis AS a ON (d.analisis_id = a.id) JOIN
responsable as r ON (d.responsable_id = r.nombre)", conex)
count = 0
for column in dispositivos:
    for i in dispositivos[column]:
        if i == "None":
            count += 1
print(f"Número de campos None: {count}")

alertas = pd.read_sql_query("SELECT * from alerts", conex)
print(f"Número de alertas: {len(alertas)}")

analisis_data = pd.read_sql_query("SELECT * from analisis", conex)
media = analisis_data['n_puertos'].mean()
desviacion = analisis_data['n_puertos'].std()
print(f"Media del total de puertos abiertos: {media:.4f}")
print(f"Desviacion estándar del total de puertos abiertos: {desviacion:.4f}")

media = analisis_data['servicios_inseguros'].mean()
desviacion = analisis_data['servicios_inseguros'].std()
print(f"Media del número de servicios inseguros detectados: {media:.4f}")
print(f"Desviación estándar del número de servicios inseguros detectados:
{desviacion:.4f}")
```

```
media = analisis_data['vulnerabilidades_detectadas'].mean()
desviacion = analisis_data['vulnerabilidades_detectadas'].std()
print(f"Media del número de vulnerabilidades detectadas: {media:.4f}")
print(f"Desviación estándar del número de vulnerabilidades detectadas:
{desviacion:.4f}")

minimo_npuertos = analisis_data['n_puertos'].min()
print(f"Valor mínimo del total de puertos abiertos: {minimo_npuertos}")
maximo_npuertos = analisis_data['n_puertos'].max()
print(f"Valor máximo del total de puertos abiertos: {maximo_npuertos}")

minimo_vulnerabilidades_detectadas =
analisis_data['vulnerabilidades_detectadas'].min()
print(f"Valor minimo del número de vulnerabilidades detectadas:
{minimo_vulnerabilidades_detectadas}")
maximo_vulnerabilidades_detectadas =
analisis_data['vulnerabilidades_detectadas'].max()
print(f"Valor maximo del número de vulnerabilidades detectadas:
{maximo_vulnerabilidades_detectadas}")
```

...se obtienen los siguientes resultados:

```
Número de dispositivos: 7
Número de campos None: 8
Número de alertas: 200225
Media del total de puertos abiertos: 3.0000
Desviación estándar del total de puertos abiertos: 1.6330
Media del número de servicios inseguros detectados: 0.7143
Desviación estándar del número de servicios inseguros detectados: 0.9512
Media del número de vulnerabilidades detectadas: 15.5714
Desviación estándar del número de vulnerabilidades detectadas: 17.5391
Valor mínimo del total de puertos abiertos: 0
Valor máximo del total de puertos abiertos: 5
Valor mínimo del número de vulnerabilidades detectadas: 2
Valor máximo del número de vulnerabilidades detectadas: 52
```

Ejercicio 3: agrupaciones de datos

Para este ejercicio ha sido necesario realizar diversas agrupaciones de los datos, según cifras de prioridad (entre 1 y 3), y por los meses de julio y agosto.

En cada uno de los dataframes resultantes, se han calculado aquellos parámetros de interés que se nos solicitaban.

El procedimiento ha sido el siguiente:

1. Volcar una query que consta de la unión de las 4 tablas por medio de operaciones JOIN sobre un dataframe
2. Emplear el método loc() sobre el dataframe para filtrar según los valores que se tomaban en cada columna que nos interesase (prioridad == X, siendo X el dígito que indica prioridad, etc.)
3. Enviar ese dataframe filtrado (y una cadena de caracteres con el mensaje que queremos imprimir en la cabecera) a una función que se encarga de calcular y mostrar los parámetros de interés.

De esta forma:

1:

```
vulns = pd.read_sql_query("SELECT al.prioridad, al.timestamp, d.*,
an.puertos_abiertos, an.n_puertos, an.servicios, an.servicios_inseguros,
an.vulnerabilidades_detectadas, r.telefono, r.rol FROM alerts AS al JOIN
devices AS d ON (al.origen = d.ip OR al.destino = d.ip) JOIN analisis AS an
ON (d.analisis_id = an.id) JOIN responsable as r ON (d.responsable_id =
r.nombre)", conex)
```

2:

```
calculos_ej3(vulns.loc[vulns['prioridad'] == 1], "Alertas de prioridad 1")
calculos_ej3(vulns.loc[vulns['prioridad'] == 2], "Alertas de prioridad 2")
calculos_ej3(vulns.loc[vulns['prioridad'] == 3], "Alertas de prioridad 3")

calculos_ej3(vulns.loc[pd.to_datetime(vulns['timestamp']).dt.month == 7],
"Alertas de julio")
calculos_ej3(vulns.loc[pd.to_datetime(vulns['timestamp']).dt.month == 8],
"Alertas de agosto")
```

3:

```
def calculos_ej3(df, str):
    print("--- "+str+" ---")
    v = df["vulnerabilidades_detectadas"]
    print(f"Número de observaciones: {len(v)}")
    count = 0
    for column in df:
        for i in df[column]:
            if i == "None":
                count += 1
    print(f"Número de valores ausentes: {count}")
    #Mediana
    print(f"Mediana: {v.median()}")
```

```
    # Media
    print(f"Media: {v.mean()}")
    # Varianza
    print(f"Varianza: {v.var()}")
    # Valores maximo y minimo
    print(f"Valor maximo: {v.max()}")
    print(f"Valor minimo: {v.min()}")
```

Nos quedan como resultados:

```
--- Alertas de prioridad 1 ---
Número de observaciones: 2591
Número de valores ausentes: 2591
Mediana: 15.0
Media: 15.0
Varianza: 0.0
Valor maximo: 15
Valor minimo: 15
--- Alertas de prioridad 2 ---
Número de observaciones: 6585
Número de valores ausentes: 6585
Mediana: 15.0
Media: 15.573120728929386
Varianza: 20.880170469170793
Valor maximo: 52
Valor minimo: 15
--- Alertas de prioridad 3 ---
Número de observaciones: 190603
Número de valores ausentes: 190603
Mediana: 15.0
Media: 14.902871413356557
Varianza: 1.0589860467123635
Valor maximo: 15
Valor minimo: 4
```

```
--- Alertas de julio ---
Número de observaciones: 11304
Número de valores ausentes: 11304
Mediana: 15.0
Media: 14.163924274593064
Varianza: 16.197757083887378
Valor maximo: 52
Valor minimo: 4
--- Alertas de agosto ---
Número de observaciones: 187558
Número de valores ausentes: 187558
Mediana: 15.0
Media: 14.973378901459816
Varianza: 0.7371736637661558
Valor maximo: 52
Valor minimo: 4
```

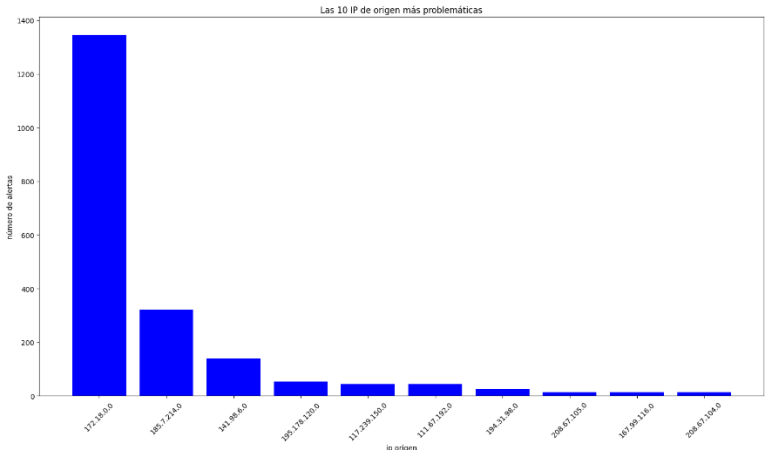
Como conclusiones de este apartado, podemos ver lo siguiente:

- Las alertas graves tienen valor 15 de forma fija.
- Hay cierta serendipia entre el número de valores ausentes con el total de veces que aparece “None” (se contabilizan tantos como campos a “None” haya, aunque eso implique sumar 2 ó 3 para una misma entrada), ya que coinciden los valores de forma exacta.
- Hay más vulnerabilidades en alertas medias, que en todos los demás niveles. Además, se alcanza tanto el máximo, como un mayor promedio aquí.
- Hubo gran densidad de observaciones durante agosto, en mucha mayor medida que en julio.

Ejercicio 4: funciones del MIS con gráficas

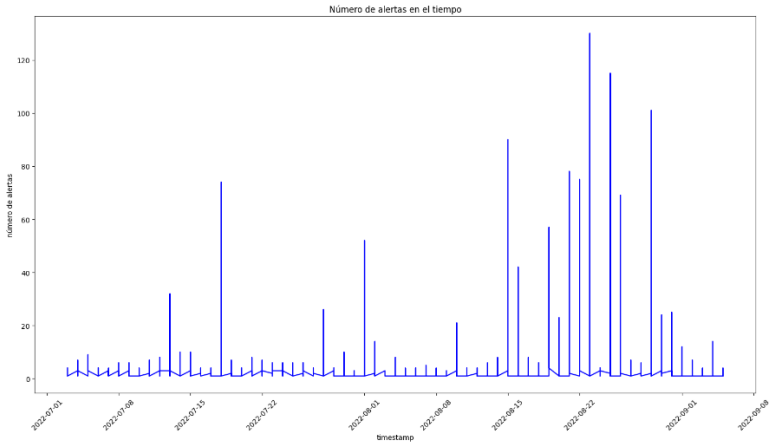
1: MOSTRAR LAS 10 IP MÁS PROBLEMÁTICAS

```
ip_problematicas = pd.read_sql_query("SELECT origen, COUNT(*) AS n FROM
alerts WHERE prioridad = 1 GROUP BY origen", conex)
ip_problematicas = ip_problematicas.sort_values('n', ascending=False)[:10]
pl.bar(ip_problematicas['origen'], ip_problematicas['n'], color="blue")
pl.xticks(rotation=45)
pl.xlabel("ip origen")
pl.ylabel("número de alertas")
pl.title("Las 10 IP de origen más problemáticas")
pl.show()
```



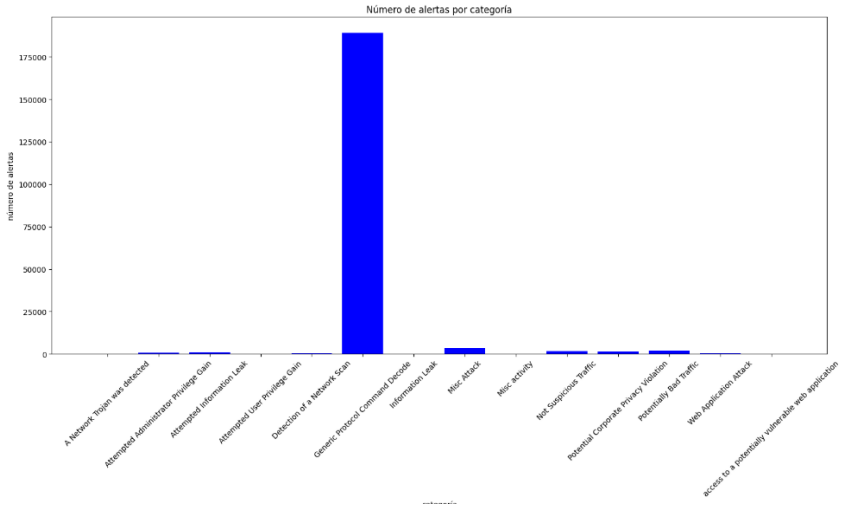
2: NÚMERO DE ALERTAS EN EL TIEMPO

```
alertas_tiempo = pd.read_sql_query("SELECT timestamp, COUNT(*) AS n FROM
alerts GROUP BY timestamp ORDER BY timestamp", conex)
alertas_tiempo['timestamp'] =
pd.to_datetime(alertas_tiempo['timestamp']).dt.date
alertas_tiempo.groupby('timestamp').sum()
pl.plot(alertas_tiempo.get('timestamp'), alertas_tiempo.get('n'),
color="blue")
pl.xticks(rotation=45)
pl.xlabel("timestamp")
pl.ylabel("número de alertas")
pl.title("Número de alertas en el tiempo")
pl.show()
```



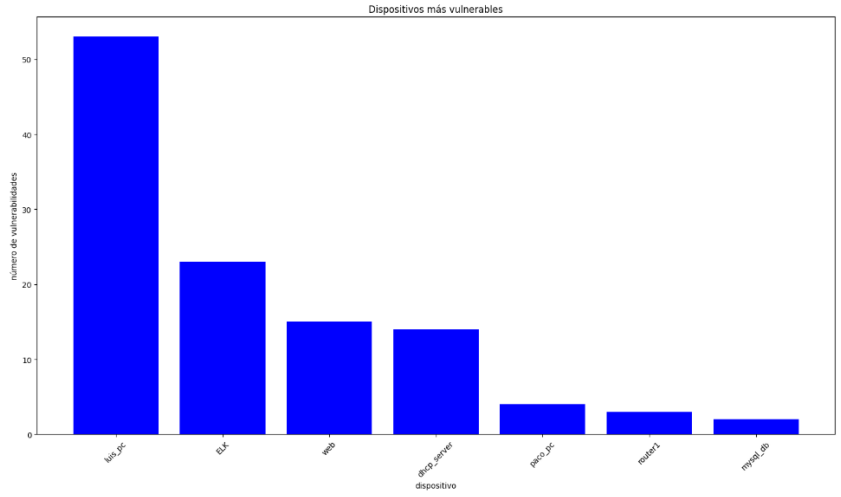
3: NÚMERO DE ALERTAS POR CATEGORÍA

```
alertas_clasificacion = pd.read_sql_query("SELECT clasificacion, COUNT(*)
AS n FROM alerts GROUP BY clasificacion", conex)
pl.bar(alertas_clasificacion['clasificacion'], alertas_clasificacion['n'],
color="blue")
pl.xticks(rotation=45)
pl.xlabel("categoría")
pl.ylabel("número de alertas")
pl.title("Número de alertas por categoría")
pl.show()
```



4: DISPOSITIVOS MÁS VULNERABLES

```
devices_vulns = pd.read_sql_query("SELECT d.id,
a.servicios_inseguros+a.vulnerabilidades_detectadas as inseguridad FROM
devices AS d JOIN analisis AS a ON (d.analisis_id = a.id)", conex)
devices_vulns = devices_vulns.sort_values('inseguridad', ascending=False)
pl.bar(devices_vulns['id'], devices_vulns['inseguridad'], color="blue")
pl.xticks(rotation=45)
pl.xlabel("dispositivo")
pl.ylabel("número de vulnerabilidades")
pl.title("Dispositivos más vulnerables")
pl.show()
```

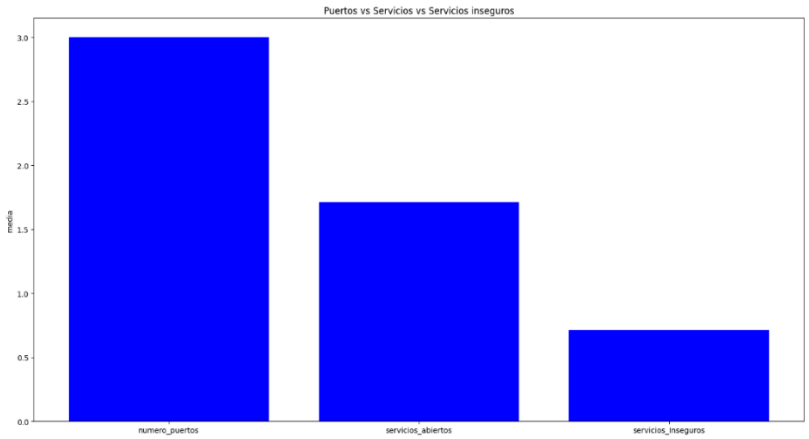


5: MEDIA DE PUERTOS ABIERTOS, SERVICIOS INSEGUROS Y SERVICIOS DETECTADOS

```
puertos_servicios = pd.read_sql_query("SELECT AVG(n_puertos) AS
numero_puertos, AVG(servicios) AS servicios_abiertos,
AVG(servicios_inseguros) AS servicios_Inseguros FROM analisis", conex)

puertos_servicios1 = []
puertos_servicios1.append(puertos_servicios['numero_puertos'][0])
puertos_servicios1.append(puertos_servicios['servicios_abiertos'][0])
puertos_servicios1.append(puertos_servicios['servicios_Inseguros'][0])

fig, ax = pl.subplots()
ax.bar(list(puertos_servicios), puertos_servicios1, color="blue")
ax.set_xlabel("")
ax.set_ylabel("media")
ax.set_title("Puertos vs Servicios vs Servicios inseguros")
pl.show()
```



Conclusiones de este apartado

- La IP 172.18.0.0 es la que más alertas está generando, necesitaría ser inspeccionada, y filtrar efectivamente el tráfico que sale de allí.
- Como ya se observó en el anterior apartado, las alertas se intensificaron durante el mes de agosto.
- La inmensa (y práctica totalidad) de las alertas son del tipo “Detection of a Network Scan”. Si está suponiendo semejante volumen de alertas, quizás debería moderarse la facilidad con la que se puede hacer un escaneo de la red.
- El dispositivo que más vulnerabilidades está sufriendo es el PC de Luis, así que sería conveniente que se revisase dicho dispositivo.

Enlace al repositorio:

https://github.com/Diego-AltF4/Practica1_Sistemas_Informacion