01-data-exploration

September 1, 2025

[ ]:
```
# 1-Data-Exploration.ipynb
# Cell #1

import boto3
import s3fs
import pandas as pd
from datetime import datetime, timezone
import json

# --- Configuration ---
BUCKET_NAME = "iot-security-logs-ln-2025"

# --- Dynamic Path Generation ---
now_utc = datetime.now(timezone.utc)
# Using a hardcoded path to ensure we're looking at the data we know exists.
# You can change this or switch back to the dynamic path later.
# s3_path_prefix = now_utc.strftime('%Y/%m/%d/%H')
s3_path_prefix = now_utc.strftime('2025/08/30/21')
print(f"Using S3 Path Prefix: {s3_path_prefix}\n")

# --- Initialize S3 Connection ---
s3_fs = s3fs.S3FileSystem()

# --- Load, Parse, and Combine ALL Zeek Data ---
print("--- Loading and Combining All Zeek Logs from the Hour ---")
all_data_rows = []
zeek_column_names = []

try:
    zeek_log_path = f"s3://{BUCKET_NAME}/zeek-logs/{s3_path_prefix}/"
    all_zeek_files = s3_fs.ls(zeek_log_path, detail=False)
    print(f"Found {len(all_zeek_files)} Zeek log file(s) to process.")

    if not all_zeek_files:
        print("No files found. Halting execution.")
    else:
        # Loop through each file found in the directory
```

```
        for i, target_file in enumerate(all_zeek_files):
            print(f"Processing file {i+1}/{len(all_zeek_files)}: {target_file}")
            with s3_fs.open(f"s3://{target_file}", 'r') as f:
                log_data = f.read()

            # Your parsing logic to handle the JSON-wrapped logs
            # This check is to avoid errors on empty files
            if log_data.strip():
                processed_log_data = '[' + log_data.strip().replace('}{',␣
 ↪'},{') + ']'
                data = json.loads(processed_log_data)
                messages = [item['message'] for item in data]

                # Get column headers from the first file that has them
                if not zeek_column_names:
                    header_lines = [m for m in messages if m.
 ↪startswith('#fields')]
                    if header_lines:
                        zeek_column_names = header_lines[0].
 ↪replace('#fields\t', '').split('\t')

                # Extract data rows and add them to our master list
                for message in messages:
                    if not message.startswith('#'):
                        all_data_rows.append(message.split('\t'))

        # Create one large DataFrame from all the collected rows
        # This check prevents an error if no data rows were found
        if all_data_rows:
            df_zeek_raw = pd.DataFrame(all_data_rows, columns=zeek_column_names)
            print(f"\nSuccessfully combined all logs into a single DataFrame␣
 ↪with {len(df_zeek_raw)} rows!")

            # Display the first 5 rows of the combined, raw data
            display(df_zeek_raw.head(20))
        else:
            print("\nCould not find any data rows after processing all files.")

except FileNotFoundError:
    print(f"No files found in {zeek_log_path}. Ensure the sensor is running and␣
 ↪sending data.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
Using S3 Path Prefix: 2025/08/30/21

--- Loading and Combining All Zeek Logs from the Hour ---
```

```
Found 7 Zeek log file(s) to process.
Processing file 1/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-12-40-11316d0c-0606-4502-a436-bc84c9b42d88
Processing file 2/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-20-31-661ca0bc-70a3-4684-9a1f-84cbb3f42825
Processing file 3/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-25-37-4fc99942-1b2d-4a32-9ac2-3a545219ce5e
Processing file 4/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-30-47-7f3e0629-dd33-4e38-811e-52262e4edc40
Processing file 5/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-35-59-c90011e7-bc0d-40dd-9a33-3081f3c44ada
Processing file 6/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-41-03-b43ffd51-eb28-41ce-a944-a8b8f84f53eb
Processing file 7/7: iot-security-logs-ln-2025/zeek-logs/2025/08/30/21/zeek-log-
stream-1-2025-08-30-21-46-02-091bff1b-659f-4e00-ae1b-d645044f8b2d

Successfully combined all logs into a single DataFrame with 644 rows!

                   ts                 uid                id.orig_h id.orig_p  \
0   1756587331.896107  CXXxDm1QNKQoPv14u5            172.31.10.164     45826
1   1756587331.239842   CR2nvC7nIx6n5cib5            172.31.10.164     45820
2   1756587335.866458  CWfbfQ1irefwNXQTf4            172.31.10.164     54480
3   1756587328.738867  Cx3syJ3acD2q6qI9Of            172.31.10.164     32880
4   1756587328.719268   CAagnm1TsxoCOCcAh            172.31.10.164     32876
5   1756587347.981807  CloNIa4beWryYXcf8h            172.31.10.164     37890
6   1756587332.504424   CaTcHT13e9KcOyU4B            172.31.10.164     45842
7   1756587350.178798  CeyQYZ3A8iZBs9IVtf            172.31.10.164     37900
8   1756587360.983570   CVXOBczm0DrqAJEL6            172.31.10.164     60530
9   1756587356.765501   CoKNAba7ahwGApTGl            172.31.10.164     60510
10  1756587350.266680   CmYUynGGwDIXLv9uj            172.31.10.164     37914
11  1756587350.312793  CTo8du1i3sRyqpNPal            172.31.10.164     37918
12  1756587361.108320  C1MKBb4vyntxKa1NTa            172.31.10.164     60562
13  1756587350.241406  CdUnG81BCTBcRUNuBl            172.31.10.164     37906
14  1756588107.005113  C3hzXH2exgAZlnOyx4  fe80::86c7:6d32:e44b:ad4     59596
15  1756587691.086573  CcG5Cm3S3Ww2vq9OG1  fe80::86c7:6d32:e44b:ad4      5353
16  1756587811.884916  CQheqh4xG3Fo9lQ2pf  fe80::86c7:6d32:e44b:ad4      5353
17  1756587932.685735  CEcdoI1u56blDX8Lxh  fe80::86c7:6d32:e44b:ad4      5353
18  1756588053.495625  Cmqwue3Xlhk4G6Isuc  fe80::86c7:6d32:e44b:ad4      5353
19  1756588174.273862  CdFqrc3llDpkuArPyh  fe80::86c7:6d32:e44b:ad4      5353

      id.resp_h id.resp_p proto service   duration orig_bytes  … local_orig  \
0   3.15.36.113       443   tcp       -  43.955859  146959339  …          T
1   3.15.36.113       443   tcp       -  44.050181  150495541  …          T
2   3.15.36.113       443   tcp       -  40.755585  111141276  …          T
3   3.15.36.115       443   tcp       -  47.634821   97455101  …          T
4   3.15.36.115       443   tcp       -  46.802936   87495569  …          T
5    3.15.36.76       443   tcp       -  24.349912   73977711  …          T
6   3.15.36.113       443   tcp       -  42.747998   87381117  …          T
```

7	3.15.36.76	443	tcp	-	16.918701	34302667	…	T
8	3.15.36.76	443	tcp	-	14.193711	46825187	…	T
9	3.15.36.76	443	tcp	-	18.768149	17126961	…	T
10	3.15.36.76	443	tcp	-	25.926494	29443599	…	T
11	3.15.36.76	443	tcp	-	24.887395	8222311	…	T
12	3.15.36.76	443	tcp	-	13.891650	4678387	…	T
13	3.15.36.76	443	tcp	-	25.132099	3227673	…	T
14	ff02::c	3702	udp	-	39.368820	9184	…	T
15	ff02::fb	5353	udp	dns	0.795268	896	…	T
16	ff02::fb	5353	udp	dns	0.793498	896	…	T
17	ff02::fb	5353	udp	dns	0.792776	896	…	T
18	ff02::fb	5353	udp	dns	0.771136	896	…	T
19	ff02::fb	5353	udp	dns	0.755019	896	…	T

	local_resp	missed_bytes	history	orig_pkts	orig_ip_bytes	\
0	F	146954995	^hCadCCCCDGGGGGCf	3	4500	
1	F	150492645	^hCadCCCDGGGGCGCf	2	3000	
2	F	111139828	^hCadCCCCDGGGGGf	1	1500	
3	F	97453653	^hCadCCCCDGGGGCGf	1	1500	
4	F	87492673	^hCadCCCDGGGGtCGf	2	3000	
5	F	73974815	^hCadCCCDGGCGGGf	2	3000	
6	F	87375325	^hCadCCCDGGGGCGf	4	6000	
7	F	34301219	^hCadCCCCDGGGGf	1	1500	
8	F	46823739	^hCadCCDCGGGGCf	1	1500	
9	F	17125513	^hCadCCCCDGGGGf	1	1500	
10	F	29442373	^hCadCCCCDGGGGf	1	1278	
11	F	8220863	^hCadCCCCDGGGGf	1	1500	
12	F	4677161	^hCadCCCDGGGCGf	1	1278	
13	F	0	^hCadCCCCDf	2	1626	
14	F	0	D	14	9856	
15	F	0	D	6	1184	
16	F	0	D	6	1184	
17	F	0	D	6	1184	
18	F	0	D	6	1184	
19	F	0	D	6	1184	

	resp_pkts	resp_ip_bytes	tunnel_parents	ip_proto
0	39564	6782666	-	6
1	37119	6000424	-	6
2	28121	4871943	-	6
3	52943	8840530	-	6
4	26621	3895332	-	6
5	17196	2661407	-	6
6	29247	4080379	-	6
7	14493	2175285	-	6
8	10322	1609803	-	6
9	11258	1742158	-	6
10	20703	2402675	-	6

```
11     19292       2656592              -        6
12      5140        504761              -        6
13     14848       2317790              -        6
14         0             0              -       17
15         0             0              -       17
16         0             0              -       17
17         0             0              -       17
18         0             0              -       17
19         0             0              -       17

[20 rows x 22 columns]
```

[ ]:
```python
# 1-Data-Exploration.ipynb - Step 4: Suricata Logs
# Cell #2

import boto3
import s3fs
import pandas as pd
from datetime import datetime, timezone
import json

# --- Configuration ---
BUCKET_NAME = "iot-security-logs-ln-2025"

# --- Dynamic Path Generation ---
now_utc = datetime.now(timezone.utc)
# Using the same hardcoded path as before to ensure we're looking at the same␣
 ↪time window.
# s3_path_prefix = now_utc.strftime('%Y/%m/%d/%H')
s3_path_prefix = now_utc.strftime('2025/08/30/21')
print(f"Using S3 Path Prefix: {s3_path_prefix}\n")

# --- Initialize S3 Connection ---
s3_fs = s3fs.S3FileSystem()

# --- Load, Parse, and Combine ALL Suricata Data ---
print("--- Loading and Combining All Suricata Logs from the Hour ---")
all_suricata_events = []

try:
    # Point to the suricata-logs directory
    suricata_log_path = f"s3://{BUCKET_NAME}/suricata-logs/{s3_path_prefix}/"
    all_suricata_files = s3_fs.ls(suricata_log_path, detail=False)
    print(f"Found {len(all_suricata_files)} Suricata log file(s) to process.")

    if not all_suricata_files:
        print("No files found. Halting execution.")
```

```
    else:
        # Loop through each file found in the directory
        for i, target_file in enumerate(all_suricata_files):

            if i >= 10:
                print("\nProcessing limit of 10 files reached. Halting loop.")
                break

            print(f"Processing file {i+1}/{len(all_suricata_files)}:␣
↪{target_file}")
            with s3_fs.open(f"s3://{target_file}", 'r') as f:
                log_data = f.read()

            # This check is to avoid errors on empty files
            if log_data.strip():
                # Same logic as before to handle concatenated JSON
                processed_log_data = '[' + log_data.strip().replace('}{',␣
↪'},{') + ']'
                data = json.loads(processed_log_data)

                # --- Suricata-Specific Parsing ---
                # Loop through each event in the file
                for item in data:
                    try:
                        # The 'message' field is a string containing another␣
↪JSON object.
                        # We need to parse this inner JSON to get the actual␣
↪event data.
                        message_str = item['message']
                        suricata_event = json.loads(message_str)
                        all_suricata_events.append(suricata_event)
                    except (json.JSONDecodeError, KeyError):
                        # Skip if the message isn't valid JSON or is missing
                        continue

        # Create one large DataFrame from the list of event dictionaries
        if all_suricata_events:
            df_suricata_raw = pd.DataFrame(all_suricata_events)
            print(f"\nSuccessfully combined all logs into a single DataFrame␣
↪with {len(df_suricata_raw)} rows!")

            # Display the first 20 rows of the combined, raw data
            print("\nDisplaying the first 20 rows of Suricata events:")
            display(df_suricata_raw.head(10))
        else:
            print("\nCould not find any data rows after processing all files.")
```

```
except FileNotFoundError:
    print(f"No files found in {suricata_log_path}. This is normal if no alerts␣
 ↪have been triggered recently.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
Using S3 Path Prefix: 2025/08/30/21

--- Loading and Combining All Suricata Logs from the Hour ---

Found 452 Suricata log file(s) to process.
Processing file 1/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-10-13-c9b1f5d0-5be9-4d76-82fd-bfd72ae6a6ea
Processing file 2/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-41-a45b4992-d3d1-4d1e-954f-0934fc8c5d4d
Processing file 3/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-42-4ea07fd5-03a3-4ce5-a815-3c5b3c77eaa1
Processing file 4/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-43-6f1d512b-39ea-4f89-a322-5e9b7a13aeb7
Processing file 5/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-43-c5f73ca5-b248-44d8-bacc-8dbdfba9dc39
Processing file 6/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-44-26c6e008-cf24-427e-84b5-2e6239c8e38d
Processing file 7/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-44-adf0c8f8-e72b-4a76-b454-8ac37362d5d4
Processing file 8/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-45-020bf654-3733-4f49-b49c-aca3b8257020
Processing file 9/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-45-95c6adf2-e1fc-4949-9db1-5d558b605726
Processing file 10/452: iot-security-logs-ln-2025/suricata-
logs/2025/08/30/21/suricata-log-
stream-1-2025-08-30-21-12-46-cd854044-ea24-4e60-aac2-75f09eb96ae9

Processing limit of 10 files reached. Halting loop.

Successfully combined all logs into a single DataFrame with 10508 rows!

Displaying the first 20 rows of Suricata events:
```

```
                      timestamp       flow_id in_iface event_type  \
0  2025-08-30T16:10:05.914791-0500  1.666669e+15     eth0      alert
1  2025-08-30T16:10:06.670643-0500  1.800191e+15     eth0      alert
2  2025-08-30T16:10:07.440361-0500  2.121479e+15     eth0      alert
3  2025-08-30T16:10:51.451333-0500  1.101974e+15     eth0      alert
4  2025-08-30T16:10:49.488866-0500  4.319253e+14     eth0      alert
5  2025-08-30T16:10:50.081229-0500  3.287941e+14     eth0      alert
6  2025-08-30T16:10:50.538220-0500  6.577751e+14     eth0      alert
7  2025-08-30T16:10:51.040037-0500  7.497040e+14     eth0      alert
8  2025-08-30T16:10:52.661847-0500  1.180232e+15     eth0      alert
9  2025-08-30T16:10:51.849904-0500  8.538107e+14     eth0      alert

          src_ip  src_port        dest_ip  dest_port proto  ip_v  …  \
0  217.160.0.187      80.0  172.31.10.164    53870.0   TCP   4.0  …
1  217.160.0.187      80.0  172.31.10.164    53886.0   TCP   4.0  …
2  217.160.0.187      80.0  172.31.10.164    53902.0   TCP   4.0  …
3  217.160.0.187      80.0  172.31.10.164    57908.0   TCP   4.0  …
4  217.160.0.187      80.0  172.31.10.164    57862.0   TCP   4.0  …
5  217.160.0.187      80.0  172.31.10.164    57874.0   TCP   4.0  …
6  217.160.0.187      80.0  172.31.10.164    57880.0   TCP   4.0  …
7  217.160.0.187      80.0  172.31.10.164    57892.0   TCP   4.0  …
8  217.160.0.187      80.0  172.31.10.164    57926.0   TCP   4.0  …
9  217.160.0.187      80.0  172.31.10.164    57910.0   TCP   4.0  …

                                                  flow stats  tls mdns  tcp  \
0  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
1  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
2  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
3  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
4  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
5  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
6  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
7  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
8  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN
9  {'pkts_toserver': 5, 'pkts_toclient': 4, 'byte…   NaN  NaN  NaN  NaN

  icmp_type icmp_code tx_id http  fileinfo
0       NaN       NaN   NaN  NaN       NaN
1       NaN       NaN   NaN  NaN       NaN
2       NaN       NaN   NaN  NaN       NaN
3       NaN       NaN   NaN  NaN       NaN
4       NaN       NaN   NaN  NaN       NaN
5       NaN       NaN   NaN  NaN       NaN
6       NaN       NaN   NaN  NaN       NaN
7       NaN       NaN   NaN  NaN       NaN
8       NaN       NaN   NaN  NaN       NaN
9       NaN       NaN   NaN  NaN       NaN
```

```
[10 rows x 24 columns]
```

[ ]:
```python
# Cell 3: Feature Engineering (Corrected and Future-Proofed)
import pandas as pd
import numpy as np

print("--- Starting Feature Engineering ---")
df_featured = df_zeek_raw.copy()
numeric_cols = ['duration', 'orig_bytes', 'resp_bytes', 'missed_bytes',␣
 ↪'orig_pkts',
                'orig_ip_bytes', 'resp_pkts', 'resp_ip_bytes']

df_featured[numeric_cols] = df_featured[numeric_cols].replace('-', np.nan)
df_featured[numeric_cols] = df_featured[numeric_cols].apply(pd.to_numeric,␣
 ↪errors='coerce')
df_featured[numeric_cols] = df_featured[numeric_cols].fillna(0)
print("Cleaned and converted numeric columns.")

# FIX: Convert the 'ts' column to a numeric type first to avoid the␣
 ↪FutureWarning
ts_numeric = pd.to_numeric(df_featured['ts'], errors='coerce')
df_featured['ts'] = pd.to_datetime(ts_numeric, unit='s', utc=True)
print("Converted timestamp column to timezone-aware UTC datetime objects.")

df_featured = df_featured.set_index('ts')

# FIX: Use 'min' for resampling instead of the deprecated 'T'
feature_df = (df_featured.groupby('id.orig_h')
              .resample('1min')
              .agg(
                  orig_bytes_sum=('orig_bytes', 'sum'),
                  resp_bytes_sum=('resp_bytes', 'sum'),
                  orig_pkts_sum=('orig_pkts', 'sum'),
                  resp_pkts_sum=('resp_pkts', 'sum'),
                  duration_mean=('duration', 'mean'),
                  unique_dest_ips=('id.resp_h', 'nunique'),
                  unique_dest_ports=('id.resp_p', 'nunique'),
                  conn_count=('uid', 'count')
              ))

feature_df = feature_df.reset_index()
feature_df = feature_df.fillna(0)

print("\nSuccessfully aggregated data into 1-minute windows for each device.")
print(f"Created feature table with {len(feature_df)} device-minute rows.")
print("\n--- Model-Ready Feature Table (First 10 Rows) ---")
display(feature_df.head(10))
```

```
--- Starting Feature Engineering ---
Cleaned and converted numeric columns.

Converted timestamp column to timezone-aware UTC datetime objects.

Successfully aggregated data into 1-minute windows for each device.
Created feature table with 149 device-minute rows.

--- Model-Ready Feature Table (First 10 Rows) ---

/tmp/ipykernel_1640/768495983.py:25: FutureWarning: DataFrameGroupBy.resample
operated on the grouping columns. This behavior is deprecated, and in a future
version of pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
  .agg(
```

	id.orig_h	ts	orig_bytes_sum	resp_bytes_sum	\
0	172.31.0.1	2025-08-30 21:01:00+00:00	1792.0	0.0	
1	172.31.0.1	2025-08-30 21:02:00+00:00	0.0	0.0	
2	172.31.0.1	2025-08-30 21:03:00+00:00	1792.0	0.0	
3	172.31.0.1	2025-08-30 21:04:00+00:00	0.0	0.0	
4	172.31.0.1	2025-08-30 21:05:00+00:00	1792.0	0.0	
5	172.31.0.1	2025-08-30 21:06:00+00:00	0.0	0.0	
6	172.31.0.1	2025-08-30 21:07:00+00:00	1792.0	0.0	
7	172.31.0.1	2025-08-30 21:08:00+00:00	18368.0	0.0	
8	172.31.0.1	2025-08-30 21:09:00+00:00	1792.0	0.0	
9	172.31.0.1	2025-08-30 21:10:00+00:00	0.0	0.0	

	orig_pkts_sum	resp_pkts_sum	duration_mean	unique_dest_ips	\
0	12	0	0.795472	1	
1	0	0	0.000000	0	
2	12	0	0.793615	1	
3	0	0	0.000000	0	
4	12	0	0.792963	1	
5	0	0	0.000000	0	
6	12	0	0.771354	1	
7	28	0	40.616044	1	
8	12	0	0.755161	1	
9	0	0	0.000000	0	

	unique_dest_ports	conn_count
0	1	2
1	0	0
2	1	2
3	0	0
4	1	2
5	0	0
6	1	2

```
7                   1           2
8                   1           2
9                   0           0
```

[ ]:
```
# Cell 4: Process Suricata Alerts (Corrected and Future-Proofed)
print("\n--- Starting Suricata Alert Processing ---")

alert_data = df_suricata_raw[df_suricata_raw['event_type'] == 'alert'].copy()
alert_details = pd.json_normalize(alert_data['alert'])
alert_details.index = alert_data.index
df_suricata_alerts = alert_data[['timestamp', 'src_ip', 'dest_ip',␣
 ↪'dest_port']].join(alert_details)

df_suricata_alerts['timestamp'] = pd.
 ↪to_datetime(df_suricata_alerts['timestamp']).dt.tz_convert('UTC')
print("Converted timestamp to timezone-aware UTC.")

# --- CRITICAL FIX IS HERE ---
# Determine the time range from our valid Zeek data
min_time = feature_df['ts'].min()
max_time = feature_df['ts'].max()

# Filter Suricata alerts to only include those within the same time range
df_suricata_alerts_filtered = df_suricata_alerts[
    (df_suricata_alerts['timestamp'] >= min_time) &␣
 ↪(df_suricata_alerts['timestamp'] <= max_time)
]
print(f"Filtered down to {len(df_suricata_alerts_filtered)} Suricata alert␣
 ↪events within the correct time window.")

df_suricata_alerts_filtered = df_suricata_alerts_filtered.set_index('timestamp')

# FIX: Use 'min' for resampling instead of the deprecated 'T'
suricata_features_df = (df_suricata_alerts_filtered.groupby('src_ip')
                        .resample('1min')
                        .agg(
                            alert_count=('signature', 'count'),
                            unique_alert_signatures=('signature', 'nunique')
                        ))

suricata_features_df = suricata_features_df.reset_index()
print(f"\nSuccessfully created Suricata feature table with␣
 ↪{len(suricata_features_df)} device-minute alert rows.")
print("\n--- Suricata Alert Features (First 10 Rows) ---")
display(suricata_features_df.head(10))
```

```
--- Starting Suricata Alert Processing ---
Converted timestamp to timezone-aware UTC.
Filtered down to 16 Suricata alert events within the correct time window.


Successfully created Suricata feature table with 2 device-minute alert rows.

--- Suricata Alert Features (First 10 Rows) ---

/tmp/ipykernel_1640/581238322.py:28: FutureWarning: DataFrameGroupBy.resample
operated on the grouping columns. This behavior is deprecated, and in a future
version of pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
  .agg(

          src_ip                 timestamp  alert_count  \
0  217.160.0.187 2025-08-30 21:10:00+00:00           13
1  217.160.0.187 2025-08-30 21:11:00+00:00            3

   unique_alert_signatures
0                        1
1                        1
```

[ ]:
```python
# Cell 5: Combine Zeek and Suricata Features (No changes needed here)
import pandas as pd

print("--- Combining Zeek and Suricata Feature Sets ---")

zeek_renamed = feature_df.rename(columns={'id.orig_h': 'device_ip', 'ts':␣
 ↪'timestamp'})
suricata_renamed = suricata_features_df.rename(columns={'src_ip': 'device_ip'})

combined_features_df = pd.merge(
    zeek_renamed,
    suricata_renamed,
    on=['device_ip', 'timestamp'],
    how='left'
)

combined_features_df[['alert_count', 'unique_alert_signatures']] =␣
 ↪combined_features_df[['alert_count', 'unique_alert_signatures']].fillna(0)

print("Successfully merged Zeek and Suricata features!")
print(f"Final feature table has {len(combined_features_df)} rows.")

print("\n--- Final Combined Feature Table (First 10 Rows) ---")
display(combined_features_df.head(10))
```

```
--- Combining Zeek and Suricata Feature Sets ---
Successfully merged Zeek and Suricata features!
Final feature table has 149 rows.

--- Final Combined Feature Table (First 10 Rows) ---
```

	device_ip	timestamp	orig_bytes_sum	resp_bytes_sum	\
0	172.31.0.1	2025-08-30 21:01:00+00:00	1792.0	0.0	
1	172.31.0.1	2025-08-30 21:02:00+00:00	0.0	0.0	
2	172.31.0.1	2025-08-30 21:03:00+00:00	1792.0	0.0	
3	172.31.0.1	2025-08-30 21:04:00+00:00	0.0	0.0	
4	172.31.0.1	2025-08-30 21:05:00+00:00	1792.0	0.0	
5	172.31.0.1	2025-08-30 21:06:00+00:00	0.0	0.0	
6	172.31.0.1	2025-08-30 21:07:00+00:00	1792.0	0.0	
7	172.31.0.1	2025-08-30 21:08:00+00:00	18368.0	0.0	
8	172.31.0.1	2025-08-30 21:09:00+00:00	1792.0	0.0	
9	172.31.0.1	2025-08-30 21:10:00+00:00	0.0	0.0	

	orig_pkts_sum	resp_pkts_sum	duration_mean	unique_dest_ips	\
0	12	0	0.795472	1	
1	0	0	0.000000	0	
2	12	0	0.793615	1	
3	0	0	0.000000	0	
4	12	0	0.792963	1	
5	0	0	0.000000	0	
6	12	0	0.771354	1	
7	28	0	40.616044	1	
8	12	0	0.755161	1	
9	0	0	0.000000	0	

	unique_dest_ports	conn_count	alert_count	unique_alert_signatures
0	1	2	0.0	0.0
1	0	0	0.0	0.0
2	1	2	0.0	0.0
3	0	0	0.0	0.0
4	1	2	0.0	0.0
5	0	0	0.0	0.0
6	1	2	0.0	0.0
7	1	2	0.0	0.0
8	1	2	0.0	0.0
9	0	0	0.0	0.0

[ ]:
```python
# Cell 6: Save Combined Features to S3
import sagemaker

print("--- Saving final feature set to S3 ---")

sagemaker_session = sagemaker.Session()
```

```
bucket = sagemaker_session.default_bucket()
prefix = 'iot-intrusion-detection/features' # A dedicated folder for our clean␣
 ↪data

# Define the S3 path where we will save the file
output_path = f"s3://{bucket}/{prefix}/combined_features.parquet"

# Save the DataFrame to S3 in Parquet format
combined_features_df.to_parquet(output_path)

print(f"Successfully saved feature set to: {output_path}")
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/home/sagemaker-user/.config/sagemaker/config.yaml
--- Saving final feature set to S3 ---
Successfully saved feature set to: s3://sagemaker-us-east-2-696680564117/iot-
intrusion-detection/features/combined_features.parquet
```