Cyber Security Attack Analysis

Created By:
Nikhil Mohan Tatfale

February 2026

1. Project Overview

This project analyzes cybersecurity attack patterns using network traffic and intrusion detection data from 40,000 recorded cyber events across multiple attack types and network segments. The primary goal is to discover meaningful patterns in attack behavior, identify high-risk time periods and geographic regions, understand malware and alert trends, and analyze severity distributions to support data-driven cybersecurity strategies.

The analysis workflow includes data collection from Kaggle, exploratory data analysis (EDA) using Python, data cleaning and null value treatment, feature extraction from raw fields, and visualization using a Power BI dashboard.

2. Dataset Summary

Rows: 40,000

Columns: 25

Key Features:

– Network Identifiers: Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol

– Traffic Details: Packet Length, Packet Type, Traffic Type, Payload Data

– Security Events: Malware Indicators, Anomaly Scores, Alerts/Warnings, Attack Type, Attack Signature, Action Taken, Severity Level

– Device & Location: User Information, Device Information, Network Segment, Geo-location Data

– Logging: Proxy Information, Firewall Logs, IDS/IPS Alerts, Log Source

Missing Data:

– Malware Indicators: 20,000 null values

– Alerts/Warnings: 20,067 null values

– Proxy Information: 19,851 null values

– Firewall Logs: 19,961 null values

– IDS/IPS Alerts: 20,050 null values

3. Exploratory Data Analysis using Python

The data preparation and cleaning process was conducted in Google Colab using Python and the Pandas library. Below are the key steps performed:

3.1 Data Loading and Initial Exploration

We started by importing the dataset using pandas and performing initial checks to understand the data structure.

```
import pandas as pd

# Load the dataset
df = pd.read_csv('/content/cybersecurity_attacks.csv')
df.head()
```

	Timestamp	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Payload Data	...	Action Taken	Severity Level	User Information	Device Information	Network Segment	Geo-location Data	Proxy Information	Firewall Logs	IDS/IPS Alerts	Log Source
0	30-05-2023 06:33	103.216.15.12	84.9.164.252	31225	17616	ICMP	503	Data	HTTP	Qui natus odio asperiores nam. Optio nobis ius...	...	Logged	Low	Reyansh Dugal	Mozilla/5.0 (compatible; MSIE 8.0; Windows NT ...	Segment A	Jamshedpur, Sikkim	150.9.97.135	Log Data	NaN	Server
1	26-08-2020 07:08	78.199.217.198	66.191.137.154	17245	48166	ICMP	1174	Data	HTTP	Aperiam quos modi officiis veritatis rem. Omni...	...	Blocked	Low	Sumer Rana	Mozilla/5.0 (compatible; MSIE 8.0; Windows NT ...	Segment B	Bilaspur, Nagaland	NaN	Log Data	NaN	Firewall
2	13-11-2022 08:23	63.79.210.48	198.219.82.17	16811	53600	UDP	306	Control	HTTP	Perferendis sapiente vitae soluta. Hic delectu...	...	Ignored	Low	Himmat Karpe	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Segment C	Bokaro, Rajasthan	114.133.48.179	Log Data	Alert Data	Firewall
3	02-07-2023 10:38	163.42.196.10	101.228.192.255	20018	32534	UDP	385	Data	HTTP	Totam maxime beatae expedita explicabo porro f...	...	Blocked	Medium	Fateh Kibe	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_11_5; ...	Segment B	Jaunpur, Rajasthan	NaN	NaN	Alert Data	Firewall
4	16-07-2023 13:11	71.166.185.76	189.243.174.238	6131	26646	TCP	1462	Data	DNS	Odit nesciunt dolorem nisi iste iusto. Animi v...	...	Blocked	Low	Dhanush Chad	Mozilla/5.0 (compatible; MSIE 5.0; Windows NT ...	Segment C	Anantapur, Tripura	149.6.110.119	NaN	Alert Data	Firewall

5 rows × 25 columns

Figure 1: Sample view of the dataset

```
# Check data structure
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Timestamp               40000 non-null  object
 1   Source IP Address       40000 non-null  object
 2   Destination IP Address  40000 non-null  object
 3   Source Port             40000 non-null  int64
 4   Destination Port        40000 non-null  int64
 5   Protocol                40000 non-null  object
 6   Packet Length           40000 non-null  int64
 7   Packet Type             40000 non-null  object
 8   Traffic Type            40000 non-null  object
 9   Payload Data            40000 non-null  object
 10  Malware Indicators      20000 non-null  object
 11  Anomaly Scores          40000 non-null  float64
 12  Alerts/Warnings         19933 non-null  object
 13  Attack Type             40000 non-null  object
 14  Attack Signature        40000 non-null  object
 15  Action Taken            40000 non-null  object
 16  Severity Level          40000 non-null  object
 17  User Information        40000 non-null  object
 18  Device Information      40000 non-null  object
 19  Network Segment         40000 non-null  object
 20  Geo-location Data       40000 non-null  object
 21  Proxy Information       20149 non-null  object
 22  Firewall Logs           20039 non-null  object
 23  IDS/IPS Alerts          19950 non-null  object
 24  Log Source              40000 non-null  object
dtypes: float64(1), int64(3), object(21)
memory usage: 7.6+ MB
```

Figure 2: df.info() showing column types and non-null counts

```
# Generate summary statistics
df.describe(include='all')
```

```
df.describe(include='all')
```

	Timestamp	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Payload Data	...	Action Taken	Severity Level	User Information	Device Information	Network Segment	Geo-location Data	Proxy Information	Firewall Logs	IDS/IPS Alerts	Log Source
count	40000	40000	40000	40000.000000	40000.000000	40000	40000.000000	40000	40000	40000	...	40000	40000	40000	40000	40000	40000	20149	20039	19950	40000
unique	39573	40000	40000	NaN	NaN	3	NaN	2	3	40000	...	3	3	32389	32104	3	8723	20148	1	1	2
top	24-03-2021 02:51	138.156.5.40	91.54.135.213	NaN	NaN	ICMP	NaN	Control	DNS	Fugiat tenetur natus perferendis. Mollitia bla...	...	Blocked	Medium	Heer Lad	Mozilla/5.0 (compatible; MSIE 6.0; Windows NT ...	Segment C	Ghaziabad, Meghalaya	39.123.165.122	Log Data	Alert Data	Firewall
freq	3	1	1	NaN	NaN	13429	NaN	20237	13376	1	...	13529	13435	6	35	13408	16	2	20039	19950	20116
mean	NaN	NaN	NaN	32970.356450	33150.868650	NaN	781.452725	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	18560.425604	18574.668842	NaN	416.044192	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	1027.000000	1024.000000	NaN	64.000000	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	16850.750000	17094.750000	NaN	420.000000	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	32856.000000	33004.500000	NaN	782.000000	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	48928.250000	49287.000000	NaN	1143.000000	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	65530.000000	65535.000000	NaN	1500.000000	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

11 rows × 25 columns

Figure 3: Statistical summary showing distribution of all features

Key observations from the initial exploration:

- Source and Destination Ports range from 1,024 to 65,535
- Packet Length ranges from 64 to 1,500 bytes with a mean of approximately 781
- Protocol types include ICMP, UDP, and TCP
- Traffic Types include DNS, HTTP, and FTP with near-equal distribution
- Dataset contains dtypes: float64(1), int64(3), object(21)

3.2 Missing Data Handling

We identified and addressed missing values across five columns in the dataset. Rather than dropping rows or using statistical imputation, descriptive placeholder values were used to preserve record completeness while clearly flagging the absence of security signals.

Figure 4: Missing value detection using df.isnull().sum()

Each column was filled with a contextually meaningful default string:

```
# Replace NULL values in Malware Indicators
df['Malware Indicators'].fillna('No Malware Detected', inplace=True)
```

```
#lets replace all NULL values from column Malware Indicators to No Malware Detected
df['Malware Indicators'].fillna('No Malware Detected', inplace=True)
```

```
/tmp/ipython-input-3902906304.py:2: FutureWarning: A value is trying to be set on a
The behavior will change in pandas 3.0. This inplace method will never work because

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method(

  df['Malware Indicators'].fillna('No Malware Detected', inplace=True)
```

Figure 5: Replacing nulls in Malware Indicators with 'No Malware Detected'

```
# Replace NULL values in Alerts/Warnings
df['Alerts/Warnings'].fillna('Not Triggered', inplace=True)
```

```
#lets replace all NULL values from column Alerts/Warnings to Not Triggered
df['Alerts/Warnings'].fillna('Not Triggered', inplace=True)
```

```
/tmp/ipython-input-2519170008.py:2: FutureWarning: A value is trying to be s
The behavior will change in pandas 3.0. This inplace method will never work

For example, when doing 'df[col].method(value, inplace=True)', try using 'df

  df['Alerts/Warnings'].fillna('Not Triggered', inplace=True)
```

Figure 6: Replacing nulls in Alerts/Warnings with 'Not Triggered'

```
# Replace NULL values in Proxy Information
df['Proxy Information'].fillna('Unknown', inplace=True)
```

```
#lets replace all NULL values from Proxy Information to Unknown
df['Proxy Information'].fillna('Unknown', inplace=True)
```

```
/tmp/ipython-input-3460102313.py:2: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.

  df['Proxy Information'].fillna('Unknown', inplace=True)
```

Figure 7: Replacing nulls in Proxy Information with 'Unknown'

```
# Replace NULL values in Firewall Logs
df['Firewall Logs'].fillna('Unknown', inplace=True)
```

```
#lets replace all NULL values from Firewall Logs to Unknown
df['Firewall Logs'].fillna('Unknown', inplace=True)
```

```
/tmp/ipython-input-3500741801.py:2: FutureWarning: A value is trying to b
The behavior will change in pandas 3.0. This inplace method will never wo

For example, when doing 'df[col].method(value, inplace=True)', try using

  df['Firewall Logs'].fillna('Unknown', inplace=True)
```

Figure 8: Replacing nulls in Firewall Logs with 'Unknown'

```
# Replace NULL values in IDS/IPS Alerts
df['IDS/IPS Alerts'].fillna('Unknown', inplace=True)
```

```
#lest replace all Null values from IDS/IPS Alerts to Unknown
df['IDS/IPS Alerts'].fillna('Unknown', inplace=True)
```

```
/tmp/ipython-input-1423474136.py:2: FutureWarning: A value is trying
The behavior will change in pandas 3.0. This inplace method will neve

For example, when doing 'df[col].method(value, inplace=True)', try us

  df['IDS/IPS Alerts'].fillna('Unknown', inplace=True)
```

Figure 9: Replacing nulls in IDS/IPS Alerts with 'Unknown'

3.3 Column Standardization

All column names were converted to snake_case format for consistency and easier downstream processing. Special characters such as '/' were also handled by renaming specific columns manually.

```
# Standardize all column names to snake case
df.columns = df.columns.str.lower().str.replace(' ', '_')

# Rename columns with special characters
df.rename(columns={
'alerts/warnings': 'alerts or warnings',
'geo-location data': 'geological location data',
'ids/ips alerts': 'ids or ips alerts'
}, inplace=True)
df.columns
```

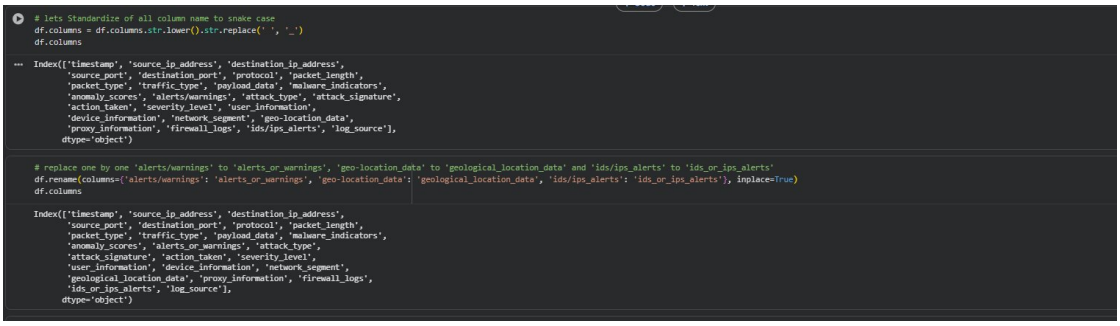

Figure 10: Standardized column names in snake_case

3.4 Feature Extraction

Three new feature sets were derived from existing raw columns to enhance analytical capabilities and enable richer dashboard filtering.

1. Date and Time from Timestamp: The timestamp column was parsed into separate date and time columns, and the original timestamp column was dropped.

```
# Parse timestamp and split into date and time
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['date'] = df['timestamp'].dt.date
df['time'] = df['timestamp'].dt.time
df.drop('timestamp', axis=1, inplace=True)
```

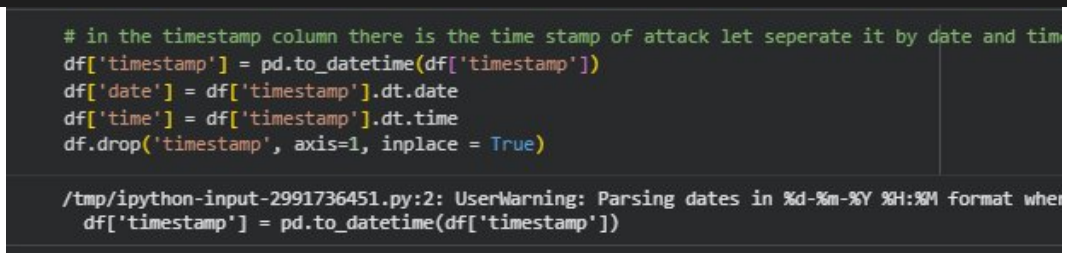

Figure 11: Extracting date and time from the timestamp column

2. City and State from Geo-location Data: The geological_location_data column containing 'City, State' strings was split into two separate columns.

```
# Split geological location data into city and state
df[['city', 'state']] = df['geological_location_data'].str.split(',',
expand=True)
df['city'] = df['city'].str.strip()
df['state'] = df['state'].str.strip()
df.drop('geological_location_data', axis=1, inplace=True)
```

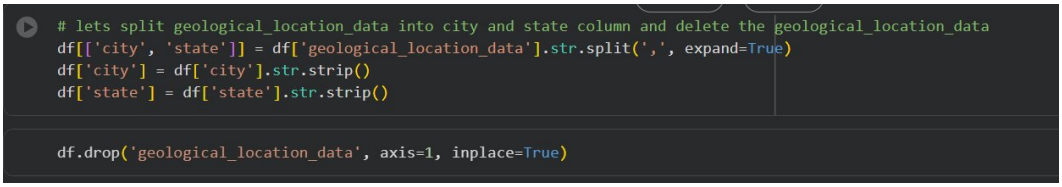

Figure 12: Extracting city and state from geo-location data

3. Operating System from Device Information: A custom function was applied to parse the device_information field and extract the operating system name.

```
def extract_os(device_info):
if pd.isna(device_info):
return 'Unknown'
os_list = ['Windows', 'Linux', 'Mac OS', 'iOS', 'Android', 'Windows CE']
for os in os_list:
if os.lower() in device_info.lower():
return os
return 'Other'
df['operating_system'] = df['device_information'].apply(extract_os)
```

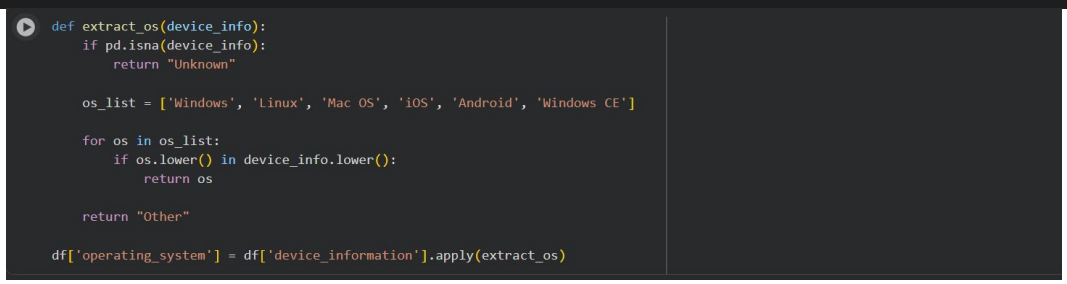

Figure 13: Extracting operating system from device information

4. Dashboard in Power BI

The cleaned and enriched dataset was loaded into Power BI to create an interactive dashboard enabling comprehensive year-over-year cyber attack analysis by attack types, network traffic, time periods, operating systems, and geographic regions.

4.1 DAX Measures Created

The following custom measures were built to power the dashboard KPIs:

– **Hour extracted** from time column for hourly attack pattern analysis

– **Month and Year extracted** from date column for trend analysis

– **Total Alerts Triggered:** Count of records where alerts_or_warnings is not 'Not Triggered'

– **Total Malware Detected:** Count of records where malware_indicators is not 'No Malware Detected'

4.2 Dashboard Overview

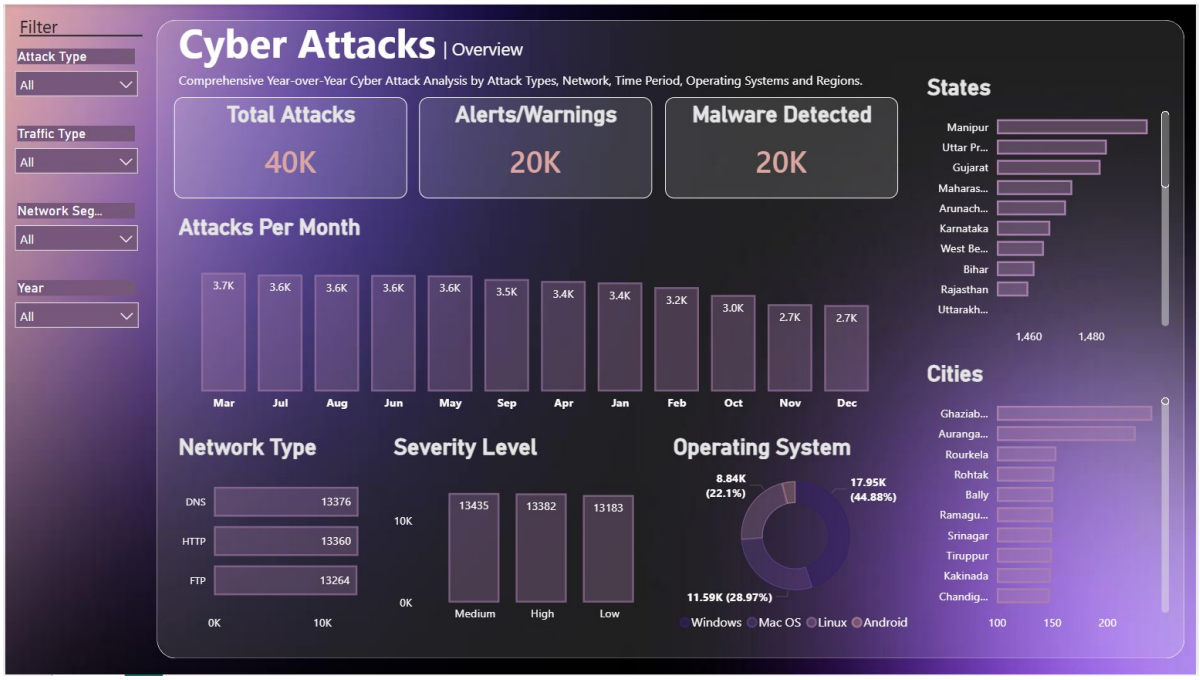

Figure 14: Power BI Dashboard — Cyber Attacks Overview

4.3 Dashboard Highlights

Key Performance Indicators:

– Total Attacks: 40K

– Total Alerts/Warnings Triggered: 20K

– Total Malware Detected: 20K

Visual Components:

– Attacks Per Month — March records the highest attack volume (3.7K), with December and November being the lowest (2.7K each)

– Network Type — DNS (13,376), HTTP (13,360), and FTP (13,264) traffic types are nearly evenly distributed

– Severity Level — Medium (13,435), High (13,382), and Low (13,183) severity attacks are similarly balanced

– Operating System — Windows dominates at 44.88%, followed by Linux (28.97%) and Mac OS (22.1%)

– States — Manipur, Uttar Pradesh, and Gujarat are the top states by attack volume

– Cities — Ghaziabad, Aurangabad, and Rourkela are the top cities by attack volume

Interactive Filters: The dashboard includes slicers for Attack Type, Traffic Type, Network Segment, and Year, enabling users to drill down into specific attack categories and analyze behavior patterns dynamically.

5. Analysis Conclusion

Based on the comprehensive exploratory data analysis and dashboard findings, the following conclusions are drawn:

1. Windows is the Most Targeted Platform

Windows is the most affected operating system, accounting for nearly 45% of all recorded attack events, followed by Linux at 28.97% and Mac OS at 22.1%. This indicates that Windows-based devices remain the primary target of cyber threats in this dataset.

2. Attack Volume Peaks in Early and Mid Year

The monthly attack distribution reveals that March records the highest number of attacks (3.7K), followed closely by July and August (3.6K each). Attack frequency gradually declines through the second half of the year, with November and December recording the lowest volumes at 2.7K each.

3. Malware and Alert Coverage is Approximately 50%

Out of 40,000 total attack records, approximately 20,000 triggered alerts and 20,000 involved detected malware. This means roughly half of all recorded events generated security signals, while the remaining half either went undetected or did not meet alerting thresholds — highlighting a significant visibility gap in the security infrastructure.

4. Geographic Concentration of Attacks

The geographic analysis reveals that Manipur, Uttar Pradesh, and Gujarat are the top states by attack volume, while Ghaziabad, Aurangabad, and Rourkela are the most frequently targeted cities. This geographic concentration suggests that certain regions face disproportionately higher exposure to cyber threats.

5. Network Traffic Types are Evenly Distributed

DNS (13,376), HTTP (13,360), and FTP (13,264) traffic types are nearly evenly distributed across all recorded attacks. This uniform spread indicates that attackers are exploiting all major network protocols with similar frequency, rather than concentrating efforts on a single vector.

6. Severity Levels Show No Clear Dominant Threat Category

Medium (13,435), High (13,382), and Low (13,183) severity attacks are distributed almost equally across all recorded events. This balance suggests that the dataset captures a broad spectrum of threat activity, from minor probing attempts to serious breaches, with no single severity tier dominating the overall attack landscape.