source: https://archive.ics.uci.edu/dataset/942/rt-iot2022

Name: Dylan James N. Dejoras
Section: CPE22S3
Date Performed: 04/03/2024
Instructor: Engr. Roman Richard

The RT-IoT2022, a proprietary dataset derived from a real-time IoT infrastructure, is introduced as a comprehensive resource integrating a diverse range of IoT devices and sophisticated network attack methodologies. This dataset encompasses both normal and adversarial network behaviours, providing a general representation of real-world scenarios. Incorporating data from IoT devices such as ThingSpeak-LED, Wipro-Bulb, and MQTT-Temp, as well as simulated attack scenarios involving Brute-Force SSH attacks, DDoS attacks using Hping and Slowloris, and Nmap patterns, RT-IoT2022 offers a detailed perspective on the complex nature of network traffic. The bidirectional attributes of network traffic are meticulously captured using the Zeek network monitoring tool and the Flowmeter plugin. Researchers can leverage the RT-IoT2022 dataset to advance the capabilities of Intrusion Detection Systems (IDS), fostering the development of robust and adaptive security solutions for real-time IoT networks.

Setup

```
pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages (0.0.6)
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
rt_iot2022 = fetch_ucirepo(id=942)

# data (as pandas dataframes)
X = rt_iot2022.data.features
y = rt_iot2022.data.targets

# metadata
print(rt_iot2022.metadata)

# variable information
print(rt_iot2022.variables)
```

```
{'uci_id': 942, 'name': 'RT-IoT2022 ', 'repository_url': 'https://archive.ics.uci.edu/dataset/942/rt-iot2022', 'data_url': 'https://arch
                     name     role         type demographic description units  \
0              id.orig_p  Feature      Integer        None        None  None
1              id.resp_p  Feature      Integer        None        None  None
2                  proto  Feature  Categorical        None        None  None
3                service  Feature   Continuous        None        None  None
4          flow_duration  Feature   Continuous        None        None  None
..                   ...      ...          ...         ...         ...   ...
80  fwd_init_window_size  Feature      Integer        None        None  None
81  bwd_init_window_size  Feature      Integer        None        None  None
82  fwd_last_window_size  Feature      Integer        None        None  None
83           Attack_type   Target  Categorical        None        None  None
84                    id       ID      Integer        None        None  None

   missing_values
0              no
1              no
2              no
3              no
4              no
..            ...
80             no
81             no
82             no
83             no
84             no

[85 rows x 7 columns]
```

view X dataframe

X

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot
0	38667	1883	tcp	mqtt	32.011598	9	5
1	51143	1883	tcp	mqtt	31.883584	9	5
2	44761	1883	tcp	mqtt	32.124053	9	5
3	60893	1883	tcp	mqtt	31.961063	9	5
4	51087	1883	tcp	mqtt	31.902362	9	5
...	...	...	...	...	...	...	...
123112	59247	63331	tcp	-	0.000006	1	1
123113	59247	64623	tcp	-	0.000007	1	1
123114	59247	64680	tcp	-	0.000006	1	1
123115	59247	65000	tcp	-	0.000006	1	1
123116	59247	65129	tcp	-	0.000006	1	1

123117 rows × 83 columns

view y dataframe

y

	Attack_type
0	MQTT_Publish
1	MQTT_Publish
2	MQTT_Publish
3	MQTT_Publish
4	MQTT_Publish
...	...
123112	NMAP_XMAS_TREE_SCAN
123113	NMAP_XMAS_TREE_SCAN
123114	NMAP_XMAS_TREE_SCAN
123115	NMAP_XMAS_TREE_SCAN
123116	NMAP_XMAS_TREE_SCAN

123117 rows × 1 columns

concatenate them

```
dataframes = [X,y]
df = pd.concat(dataframes, axis = 1)
df
```

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot
0	38667	1883	tcp	mqtt	32.011598	9	5
1	51143	1883	tcp	mqtt	31.883584	9	5
2	44761	1883	tcp	mqtt	32.124053	9	5
3	60893	1883	tcp	mqtt	31.961063	9	5
4	51087	1883	tcp	mqtt	31.902362	9	5
...	...	...	...	...	...	...	...
123112	59247	63331	tcp	-	0.000006	1	1
123113	59247	64623	tcp	-	0.000007	1	1
123114	59247	64680	tcp	-	0.000006	1	1
123115	59247	65000	tcp	-	0.000006	1	1
123116	59247	65129	tcp	-	0.000006	1	1

123117 rows × 84 columns

check columns

```
df.columns
```

```
Index(['id.orig_p', 'id.resp_p', 'proto', 'service', 'flow_duration',
       'fwd_pkts_tot', 'bwd_pkts_tot', 'fwd_data_pkts_tot',
       'bwd_data_pkts_tot', 'fwd_pkts_per_sec', 'bwd_pkts_per_sec',
       'flow_pkts_per_sec', 'down_up_ratio', 'fwd_header_size_tot',
       'fwd_header_size_min', 'fwd_header_size_max', 'bwd_header_size_tot',
       'bwd_header_size_min', 'bwd_header_size_max', 'flow_FIN_flag_count',
       'flow_SYN_flag_count', 'flow_RST_flag_count', 'fwd_PSH_flag_count',
       'bwd_PSH_flag_count', 'flow_ACK_flag_count', 'fwd_URG_flag_count',
       'bwd_URG_flag_count', 'flow_CWR_flag_count', 'flow_ECE_flag_count',
       'fwd_pkts_payload.min', 'fwd_pkts_payload.max', 'fwd_pkts_payload.tot',
       'fwd_pkts_payload.avg', 'fwd_pkts_payload.std', 'bwd_pkts_payload.min',
       'bwd_pkts_payload.max', 'bwd_pkts_payload.tot', 'bwd_pkts_payload.avg',
       'bwd_pkts_payload.std', 'flow_pkts_payload.min',
       'flow_pkts_payload.max', 'flow_pkts_payload.tot',
       'flow_pkts_payload.avg', 'flow_pkts_payload.std', 'fwd_iat.min',
       'fwd_iat.max', 'fwd_iat.tot', 'fwd_iat.avg', 'fwd_iat.std',
       'bwd_iat.min', 'bwd_iat.max', 'bwd_iat.tot', 'bwd_iat.avg',
       'bwd_iat.std', 'flow_iat.min', 'flow_iat.max', 'flow_iat.tot',
       'flow_iat.avg', 'flow_iat.std', 'payload_bytes_per_second',
       'fwd_subflow_pkts', 'bwd_subflow_pkts', 'fwd_subflow_bytes',
       'bwd_subflow_bytes', 'fwd_bulk_bytes', 'bwd_bulk_bytes',
       'fwd_bulk_packets', 'bwd_bulk_packets', 'fwd_bulk_rate',
       'bwd_bulk_rate', 'active.min', 'active.max', 'active.tot', 'active.avg',
       'active.std', 'idle.min', 'idle.max', 'idle.tot', 'idle.avg',
       'idle.std', 'fwd_init_window_size', 'bwd_init_window_size',
       'fwd_last_window_size', 'Attack_type'],
      dtype='object')
```

Some information of the dataframe

```
df.info()
```

```
 46  fwd_iat.tot               123117 non-null  float64
 47  fwd_iat.avg               123117 non-null  float64
 48  fwd_iat.std               123117 non-null  float64
 49  bwd_iat.min               123117 non-null  float64
 50  bwd_iat.max               123117 non-null  float64
 51  bwd_iat.tot               123117 non-null  float64
 52  bwd_iat.avg               123117 non-null  float64
 53  bwd_iat.std               123117 non-null  float64
 54  flow_iat.min              123117 non-null  float64
 55  flow_iat.max              123117 non-null  float64
 56  flow_iat.tot              123117 non-null  float64
 57  flow_iat.avg              123117 non-null  float64
 58  flow_iat.std              123117 non-null  float64
 59  payload_bytes_per_second  123117 non-null  float64
 60  fwd_subflow_pkts          123117 non-null  float64
 61  bwd_subflow_pkts          123117 non-null  float64
 62  fwd_subflow_bytes         123117 non-null  float64
 63  bwd_subflow_bytes         123117 non-null  float64
 64  fwd_bulk_bytes            123117 non-null  float64
 65  bwd_bulk_bytes            123117 non-null  float64
 66  fwd_bulk_packets          123117 non-null  float64
 67  bwd_bulk_packets          123117 non-null  float64
 68  fwd_bulk_rate             123117 non-null  float64
 69  bwd_bulk_rate             123117 non-null  float64
 70  active.min                123117 non-null  float64
 71  active.max                123117 non-null  float64
 72  active.tot                123117 non-null  float64
 73  active.avg                123117 non-null  float64
 74  active.std                123117 non-null  float64
 75  idle.min                  123117 non-null  float64
 76  idle.max                  123117 non-null  float64
 77  idle.tot                  123117 non-null  float64
 78  idle.avg                  123117 non-null  float64
 79  idle.std                  123117 non-null  float64
 80  fwd_init_window_size      123117 non-null  int64
 81  bwd_init_window_size      123117 non-null  int64
 82  fwd_last_window_size      123117 non-null  int64
 83  Attack_type               123117 non-null  object
dtypes: float64(47), int64(34), object(3)
memory usage: 78.9+ MB
```

Check datatypes of dataframe

```
df.dtypes.unique()
```

```
array([dtype('int64'), dtype('O'), dtype('float64')], dtype=object)
```

In this case, I decided to focus on attack types.

```
attk_types = list(df['Attack_type'].unique())
attk_types
```

```
['MQTT_Publish',
 'Thing_Speak',
 'Wipro_bulb',
 'ARP_poisioning',
 'DDOS_Slowloris',
 'DOS_SYN_Hping',
 'Metasploit_Brute_Force_SSH',
 'NMAP_FIN_SCAN',
 'NMAP_OS_DETECTION',
 'NMAP_TCP_scan',
 'NMAP_UDP_SCAN',
 'NMAP_XMAS_TREE_SCAN']
```

```
proto_types = list(df['proto'].unique())
```

```
proto_types
```

```
['tcp', 'udp', 'icmp']
```

```
service_types = list(df['service'].unique())
service_types
```

```
['mqtt', '-', 'http', 'dns', 'ntp', 'ssl', 'dhcp', 'irc', 'ssh', 'radius']
```

Function for getting indices

```
def indices(lizt):
  for x in lizt: # for loop for x in attk_types
    lambda_use = lambda x: x # had to assign lambda x: x as a variable for the enumerate() function
    return [lambda_use(i) for i, _ in enumerate(lizt)] # return the indices which are aligned with the values
indices(attk_types) # call function
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
indices(proto_types)
indices(service_types)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
df_b = pd.DataFrame(attk_types)
df_b # df_b is a new dataframe
```

	0
0	MQTT_Publish
1	Thing_Speak
2	Wipro_bulb
3	ARP_poisioning
4	DDOS_Slowloris
5	DOS_SYN_Hping
6	Metasploit_Brute_Force_SSH
7	NMAP_FIN_SCAN
8	NMAP_OS_DETECTION
9	NMAP_TCP_scan
10	NMAP_UDP_SCAN
11	NMAP_XMAS_TREE_SCAN

Next steps: View recommended plots

Rename the column to attack types

```
df_b.rename(columns = {0 : 'Attack Types'})
```

	Attack Types
0	MQTT_Publish
1	Thing_Speak
2	Wipro_bulb
3	ARP_poisioning
4	DDOS_Slowloris
5	DOS_SYN_Hping
6	Metasploit_Brute_Force_SSH
7	NMAP_FIN_SCAN
8	NMAP_OS_DETECTION
9	NMAP_TCP_scan
10	NMAP_UDP_SCAN
11	NMAP_XMAS_TREE_SCAN

```
# in this case, we use the categorical columns, we apply lambda to the dataframes in which we get their indices using the x variable and .inde
df['Attack_type'] = df.apply(lambda x: attk_types.index(x['Attack_type']),axis = 1) # axis = 1 to execute
df['proto'] = df.apply(lambda x: proto_types.index(x['proto']),axis = 1)
df['service'] = df.apply(lambda x: service_types.index(x['service']),axis = 1)
```

```
df # as you can see, the proto, service, and Attack_type column are changed accordingly
```

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_data_pkts_tot	bwd_data_pkts_tot	fwd_pkts_
0	38667	1883	0	0	32.011598	9	5	3	3	0
1	51143	1883	0	0	31.883584	9	5	3	3	0
2	44761	1883	0	0	32.124053	9	5	3	3	0
3	60893	1883	0	0	31.961063	9	5	3	3	0
4	51087	1883	0	0	31.902362	9	5	3	3	0
...	...	...	...	...	...	...	...	...	...	
123112	59247	63331	0	1	0.000006	1	1	0	0	167772
123113	59247	64623	0	1	0.000007	1	1	0	0	144631
123114	59247	64680	0	1	0.000006	1	1	0	0	167772
123115	59247	65000	0	1	0.000006	1	1	0	0	167772
123116	59247	65129	0	1	0.000006	1	1	0	0	167772

123117 rows × 84 columns