Malicious or Benign Websites EDA - Research Report Part 1 of 2

Dataset: https://www.kaggle.com/datasets/xwolf12/malicious-and-benign-websites/

Chris Heimbuch: https://github.com/chrisheimbuch

header1.jfif

Overview

This dataset's data was obtained by using different verified sources of benign and malicious URL's, in a low interactive client honeypot to isolate network traffic. The team crafting the dataset used additional tools to get other information, such as, server country with Whois. This project consisted to evaluate different classification models to predict malicious and benign websites, based on application layer and network characteristics.

The columns that this dataset includes are:

- **url** - It is a uniquely encoded website mask to cover the website name. Every website is unique.
- **url_length** - How many characters the length of the url.
- **number_special_characters** - The number of special characters in a URL, such as '#', '/', '%', '&'.
- **charset** - Categorical value and it is the character encoding standard.
- **server** - The operative system of the server in which was sent from the packet response.
- **content_length** - Represents the content size of the HTTP header.
- **whois_country** - The values are the countries that the server response came from.
- **whois_statepro** - The values are the state that the server response came from.
- **whois_regdate** - Whois provides the server registration date, with format DD/MM/YYY HH:MM
- **whois_updated_date** - Through the Whois, it is the laste update date from the server analyzed.
- **tcp_conversation_exchange** - This variable is the number of TCP packets exchanged between the server and the honeypot client.
- **dist_remote_tcp_port** - The number of ports detected and different to TCP.

- **remote_ips** - The total number of IPs connected to the honeypot.
- **app_bytes** - The total number of bytes transferred.
- **source_app_packets** - Packets sent from the honeypot to the server.
- **remote_app_packets** - Packets received from the server.
- **app_packets** - The total number of IP packets generated during the communication between the honeypot and the server.
- **dns_query_times** - The number of DNS packets generated during the communication between the honeypot and the server.
- **type** - This represents whether or not a website is benign or malicious. 1 represents malicious and 0 represents benign.

For my analysis, Section 1 comprised of getting familiar with my data using many data familiarity techniques, such as inspecting the shape, getting the head of my dataframe by invoking the .head() method, checking for null values, using the describe technique and the .info() technique to understand data types. Next, I did extensive data cleaning. I used KNN imputation to fill in a column with the nearest neighbors and let the algorithm handle it and added some new columns on standardized / clean data that I processed extensively. Many of the 'whois_country' and 'whois_state' data points were repeated and in different font such as uppercase and lower, and I did extensive cleaning to that. Section 2 comprised of Descriptive Questions and answers via beautified graphical representations. Section 3 comprised of inferential analysis and hypothesis testing. Finally, Section 4 is an analysis and conclusion of findings. The next notebook in this research report will cover the machine learning technical aspect of the notebook, to create models to predict if websites are malicious or benign. I designed a powerpoint presentation catered to all individuals and companies as protecting yourself or your organization against malicious people is extremely important in today's environment. This was a fun project and hope you enjoy!

Section 1: Data Cleaning

In [ ]:

```
# Standard DS imports
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import matplotlib.patheffects as path_effects

# Inferential Analysis Tests
from scipy import stats
from scipy.stats import mannwhitneyu
from scipy.stats import kruskal
from scipy.stats import ttest_ind
from scipy.stats import f_oneway

#For null value imputation
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
# Call in dataset and inspect the head.
df = pd.read_csv("dataset.csv")
df.head()
```

Out[ ]:

	URL	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CHARSET	SERVER	CONTI
0	M0_109	16	7	iso-8859-1	nginx	
1	B0_2314	16	6	UTF-8	Apache/2.4.10	
2	B0_911	16	6	us-ascii	Microsoft-HTTPAPI/2.0	
3	B0_113	17	6	ISO-8859-1	nginx	
4	B0_403	17	6	UTF-8	NaN	

5 rows × 21 columns

In [ ]:

```
#Inspect dimensionality of dataset.
print(df.shape)
```

```
(1781, 21)
```

In [ ]:

```
#Inspect the column names and see what we are working with.
df.columns
```

Out[ ]:

```
Index(['URL', 'URL_LENGTH', 'NUMBER_SPECIAL_CHARACTERS', 'CHARSET', 'SERVER',
       'CONTENT_LENGTH', 'WHOIS_COUNTRY', 'WHOIS_STATEPRO', 'WHOIS_REGDATE',
       'WHOIS_UPDATED_DATE', 'TCP_CONVERSATION_EXCHANGE',
       'DIST_REMOTE_TCP_PORT', 'REMOTE_IPS', 'APP_BYTES', 'SOURCE_APP_PACKETS',
       'REMOTE_APP_PACKETS', 'SOURCE_APP_BYTES', 'REMOTE_APP_BYTES',
       'APP_PACKETS', 'DNS_QUERY_TIMES', 'Type'],
      dtype='object')
```

In [ ]:

```
# Call info to inspect data types and get a preliminary investigtion of any null va
df.info()

print(df.isna().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1781 entries, 0 to 1780
Data columns (total 21 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   URL                        1781 non-null   object 
 1   URL_LENGTH                 1781 non-null   int64  
 2   NUMBER_SPECIAL_CHARACTERS  1781 non-null   int64  
 3   CHARSET                    1774 non-null   object 
 4   SERVER                     1605 non-null   object 
 5   CONTENT_LENGTH             969 non-null    float64
 6   WHOIS_COUNTRY              1475 non-null   object 
 7   WHOIS_STATEPRO             1419 non-null   object 
 8   WHOIS_REGDATE              1654 non-null   object 
 9   WHOIS_UPDATED_DATE         1642 non-null   object 
 10  TCP_CONVERSATION_EXCHANGE  1781 non-null   int64  
 11  DIST_REMOTE_TCP_PORT       1781 non-null   int64  
 12  REMOTE_IPS                 1781 non-null   int64  
 13  APP_BYTES                  1781 non-null   int64  
 14  SOURCE_APP_PACKETS         1781 non-null   int64  
 15  REMOTE_APP_PACKETS         1781 non-null   int64  
 16  SOURCE_APP_BYTES           1781 non-null   int64  
 17  REMOTE_APP_BYTES           1781 non-null   int64  
 18  APP_PACKETS                1781 non-null   int64  
 19  DNS_QUERY_TIMES            1780 non-null   float64
 20  Type                       1781 non-null   int64  
dtypes: float64(2), int64(12), object(7)
memory usage: 292.3+ KB
URL                            0
URL_LENGTH                     0
NUMBER_SPECIAL_CHARACTERS      0
CHARSET                        7
SERVER                       176
CONTENT_LENGTH               812
WHOIS_COUNTRY                306
WHOIS_STATEPRO               362
WHOIS_REGDATE                127
WHOIS_UPDATED_DATE           139
TCP_CONVERSATION_EXCHANGE      0
DIST_REMOTE_TCP_PORT           0
REMOTE_IPS                     0
APP_BYTES                      0
SOURCE_APP_PACKETS             0
REMOTE_APP_PACKETS             0
SOURCE_APP_BYTES               0
REMOTE_APP_BYTES               0
APP_PACKETS                    0
DNS_QUERY_TIMES                1
Type                           0
dtype: int64
```

In [ ]:

```
#Seeing how many null values I have and in what columns.
df.isna().sum()
```

Out[ ]:

```
URL                            0
URL_LENGTH                     0
NUMBER_SPECIAL_CHARACTERS      0
CHARSET                        7
SERVER                       176
CONTENT_LENGTH               812
WHOIS_COUNTRY                306
WHOIS_STATEPRO               362
WHOIS_REGDATE                127
WHOIS_UPDATED_DATE           139
TCP_CONVERSATION_EXCHANGE      0
DIST_REMOTE_TCP_PORT           0
REMOTE_IPS                     0
APP_BYTES                      0
SOURCE_APP_PACKETS             0
REMOTE_APP_PACKETS             0
SOURCE_APP_BYTES               0
REMOTE_APP_BYTES               0
APP_PACKETS                    0
DNS_QUERY_TIMES                1
Type                           0
dtype: int64
```

In [ ]:

```
# Simple Statistical summary of data set with numerical values.
df.describe()
```

Out[ ]:

	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CONTENT_LENGTH	TCP_CONVERSATI
count	1781.000000	1781.000000	969.000000	
mean	56.961258	11.111735	11726.927761	
std	27.555586	4.549896	36391.809051	
min	16.000000	5.000000	0.000000	
25%	39.000000	8.000000	324.000000	
50%	49.000000	10.000000	1853.000000	
75%	68.000000	13.000000	11323.000000	
max	249.000000	43.000000	649263.000000	

In [ ]:

```
#Seeing entire DF.
df
```

Out[ ]:

	URL	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CHARSET	SERVER	CO
0	M0_109	16	7	iso-8859-1	nginx	
1	B0_2314	16	6	UTF-8	Apache/2.4.10	
2	B0_911	16	6	us-ascii	Microsoft-HTTPAPI/2.0	
3	B0_113	17	6	ISO-8859-1	nginx	
4	B0_403	17	6	UTF-8	NaN	
...	...	...	...	...	...	
1776	M4_48	194	16	UTF-8	Apache	
1777	M4_41	198	17	UTF-8	Apache	
1778	B0_162	201	34	utf-8	Apache/2.2.16 (Debian)	
1779	B0_1152	234	34	ISO-8859-1	cloudflare-nginx	
1780	B0_676	249	40	utf-8	Microsoft-IIS/8.5	

1781 rows × 21 columns

Just a quick preliminary review of my data, it is messy. Columns are uppercase with the last column being title case, there are null values across many rows with some reaching over 800 null values, string data has dashes and certain strings that are the same name such as UTF-8 in the "CHARSET" column are upper case and lower case, the date fields are all in different order with some random characters. Therefore, I will add some consistency across my data to prepare for the exploratory data analysis.

In [ ]:

```python
#Going to create a copy of the df to work with.
df_copy = df.copy()
```

In [ ]:

```python
#Make all columns lowercase
df_copy.columns = df_copy.columns.str.lower()
```

In [ ]:

```python
#Clean up the 'charset' column: replace the dashes with spaces and make it Title ca
df_copy['charset'] = df_copy['charset'].str.replace('-', ' ')
df_copy['charset'] = df_copy['charset'].str.title()
df_copy['charset'].unique()
```

Out[ ]:
```
array(['Iso 8859 1', 'Utf 8', 'Us Ascii', nan, 'Windows 1251', 'Iso 8859',
       'Windows 1252'], dtype=object)
```

In [ ]:
```
#Going to perform null value imputation with KNN imputation method.

#Encode only the known values.
le = LabelEncoder()
df_copy['charset_encoded'] = df_copy['charset']
df_copy.loc[df_copy['charset'].notnull(), 'charset_encoded'] = le.fit_transform(df_

#Apply KNN imputation on the encoded column.
imputer = KNNImputer(n_neighbors=3)
df_copy['charset_encoded'] = imputer.fit_transform(df_copy[['charset_encoded']])

#Inverse transform the encoded data back to the original categories.
df_copy['charset_imputed'] = df_copy['charset_encoded'].round().astype(int)
df_copy['charset_imputed'] = le.inverse_transform(df_copy['charset_imputed'])

#Compare the original and imputed columns to verify our imputation has succeeded.
imputed_values = df_copy[df_copy['charset'].isnull()]

print(imputed_values[['charset', 'charset_imputed']])
```

```
     charset charset_imputed
35       NaN        Us Ascii
81       NaN        Us Ascii
125      NaN        Us Ascii
159      NaN        Us Ascii
952      NaN        Us Ascii
977      NaN        Us Ascii
1069     NaN        Us Ascii
```

In [ ]:
```
#Dropping as no longer needed.
df_copy = df_copy.drop(columns=["charset_encoded", "charset"])
```

In [ ]:
```
#Sanity check.
df_copy.isna().sum()
```

Out[ ]:

```
url                            0
url_length                     0
number_special_characters      0
server                       176
content_length               812
whois_country                306
whois_statepro               362
whois_regdate                127
whois_updated_date           139
tcp_conversation_exchange      0
dist_remote_tcp_port           0
remote_ips                     0
app_bytes                      0
source_app_packets             0
remote_app_packets             0
source_app_bytes               0
remote_app_bytes               0
app_packets                    0
dns_query_times                1
type                           0
charset_imputed                0
dtype: int64
```

In [ ]:

```
#Looking into "server" column now. There is alot of random noise here.
df_copy['server'].unique()
```

```
Out[ ]: array(['nginx', 'Apache/2.4.10', 'Microsoft-HTTPAPI/2.0', nan, 'Apache/2',
               'nginx/1.10.1', 'Apache', 'Apache/2.2.15 (Red Hat)',
               'Apache/2.4.23 (Unix) OpenSSL/1.0.1e-fips mod_bwlimited/1.4',
               'openresty/1.11.2.1', 'Apache/2.2.22', 'Apache/2.4.7 (Ubuntu)',
               'nginx/1.12.0',
               'Apache/2.4.12 (Unix) OpenSSL/1.0.1e-fips mod_bwlimited/1.4',
               'Oracle-iPlanet-Web-Server/7.0', 'cloudflare-nginx', 'nginx/1.6.2',
               'openresty', 'Heptu web server', 'Pepyaka/1.11.3', 'nginx/1.8.0',
               'nginx/1.10.1 + Phusion Passenger 5.0.30',
               'Apache/2.2.29 (Amazon)', 'Microsoft-IIS/7.5', 'LiteSpeed',
               'Apache/2.4.25 (cPanel) OpenSSL/1.0.1e-fips mod_bwlimited/1.4',
               'tsa_c', 'Apache/2.2.0 (Fedora)', 'Apache/2.2.22 (Debian)',
               'Apache/2.2.15 (CentOS)', 'Apache/2.4.25',
               'Apache/2.4.25 (Amazon) PHP/7.0.14', 'GSE',
               'Apache/2.4.23 (Unix) OpenSSL/0.9.8e-fips-rhel5 mod_bwlimited/1.4',
               'Apache/2.4.25 (Amazon) OpenSSL/1.0.1k-fips',
               'Apache/2.2.22 (Ubuntu)', 'Tengine',
               'Apache/2.4.18 (Unix) OpenSSL/0.9.8e-fips-rhel5 mod_bwlimited/1.4',
               'Apache/2.4.10 (Debian)', 'Apache/2.4.6 (CentOS) PHP/5.6.8',
               'Sun-ONE-Web-Server/6.1',
               'Apache/2.4.18 (Unix) OpenSSL/1.0.2e Communique/4.1.10',
               'AmazonS3',
               'Apache/1.3.37 (Unix) mod_perl/1.29 mod_ssl/2.8.28 OpenSSL/0.9.7e-p1',
               'ATS', 'Apache/2.2.27 (CentOS)',
               'Apache/2.2.29 (Unix) mod_ssl/2.2.29 OpenSSL/1.0.1e-fips DAV/2 mod_bwlimite
        d/1.4',
               'CherryPy/3.6.0', 'Server', 'KHL',
               'Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/5.4.16 mod_j
        k/1.2.40',
               'Apache/2.2.3 (CentOS)', 'Apache/2.4',
               'Apache/1.3.27 (Unix)  (Red-Hat/Linux) mod_perl/1.26 PHP/4.3.3 FrontPage/5.
        0.2 mod_ssl/2.8.12 OpenSSL/0.9.6b',
               'mw2114.codfw.wmnet',
               'Apache/2.2.31 (Unix) mod_ssl/2.2.31 OpenSSL/1.0.1e-fips mod_bwlimited/1.4
        mod_perl/2.0.8 Perl/v5.10.1',
               'Apache/1.3.34 (Unix) PHP/4.4.4', 'Apache/2.2.31 (Amazon)',
               'Jetty(9.0.z-SNAPSHOT)', 'Apache/2.2.31 (CentOS)',
               'Apache/2.4.12 (Ubuntu)', 'HTTPDaemon',
               'Apache/2.2.29 (Unix) mod_ssl/2.2.29 OpenSSL/1.0.1e-fips mod_bwlimited/1.
        4',
               'MediaFire', 'DOSarrest', 'mw2232.codfw.wmnet',
               'Sucuri/Cloudproxy', 'Apache/2.4.23 (Unix)', 'nginx/0.7.65',
               'mw2260.codfw.wmnet', 'Apache/2.2.32', 'mw2239.codfw.wmnet',
               'DPS/1.1.8', 'Apache/2.0.52 (Red Hat)',
               'Apache/2.2.25 (Unix) mod_ssl/2.2.25 OpenSSL/0.9.8e-fips-rhel5 mod_bwlimite
        d/1.4',
               'Apache/1.3.31 (Unix) PHP/4.3.9 mod_perl/1.29 rus/PL30.20',
               'Apache/2.2.13 (Unix) mod_ssl/2.2.13 OpenSSL/0.9.8e-fips-rhel5 mod_auth_pas
        sthrough/2.1 mod_bwlimited/1.4 PHP/5.2.10',
               'nginx/1.1.19', 'ATS/5.3.0', 'Apache/2.2.3 (Red Hat)',
               'nginx/1.4.3',
               'Apache/2.2.29 (Unix) mod_ssl/2.2.29 OpenSSL/1.0.1e-fips mod_bwlimited/1.4
        PHP/5.4.35',
               'Apache/2.2.14 (FreeBSD) mod_ssl/2.2.14 OpenSSL/0.9.8y DAV/2 PHP/5.2.12 wit
        h Suhosin-Patch',
               'Apache/2.2.14 (Unix) mod_ssl/2.2.14 OpenSSL/0.9.8e-fips-rhel5',
               'Apache/1.3.39 (Unix) PHP/5.2.5 mod_auth_passthrough/1.8 mod_bwlimited/1.4
        mod_log_bytes/1.2 mod_gzip/1.3.26.1a FrontPage/5.0.2.2635 DAV/1.0.3 mod_ssl/2.8.30
        OpenSSL/0.9.7a',
               'SSWS', 'Microsoft-IIS/8.0', 'Apache/2.4.18 (Ubuntu)',
               'Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16 mod_apreq2-20090110/
        2.8.0 mod_perl/2.0.10 Perl/v5.24.1',
               'Apache/2.2.20 (Unix)', 'YouTubeFrontEnd', 'nginx/1.11.3',
               'nginx/1.11.2', 'nginx/1.10.0 (Ubuntu)', 'nginx/1.8.1',
               'nginx/1.11.10', 'Squeegit/1.2.5 (3_sir)',
               'Virtuoso/07.20.3217 (Linux) i686-generic-linux-glibc212-64  VDB',
               'Apache-Coyote/1.1', 'Yippee-Ki-Yay', 'mw2165.codfw.wmnet',
               'mw2192.codfw.wmnet', 'Apache/2.2.23 (Amazon)',
               'nginx/1.4.6 (Ubuntu)', 'nginx + Phusion Passenger',
               'Proxy Pandeiro UOL', 'mw2231.codfw.wmnet', 'openresty/1.11.2.2',
               'mw2109.codfw.wmnet', 'nginx/0.8.54', 'Apache/2.4.6',
               'mw2225.codfw.wmnet', 'Apache/1.3.27 (Unix) PHP/4.4.1',
               'mw2236.codfw.wmnet', 'mw2101.codfw.wmnet', 'Varnish',
               'Resin/3.1.8', 'mw2164.codfw.wmnet', 'Microsoft-IIS/8.5',
               'mw2242.codfw.wmnet',
               'Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.5.38',
               'mw2175.codfw.wmnet', 'mw2107.codfw.wmnet', 'mw2190.codfw.wmnet',
               'Apache/2.4.6 (CentOS)', 'nginx/1.13.0', 'barista/5.1.3',
               'mw2103.codfw.wmnet', 'Apache/2.4.25 (Debian)', 'ECD (fll/0790)',
               'Pagely Gateway/1.5.1', 'nginx/1.10.3',
               'Apache/2.4.25 (FreeBSD) OpenSSL/1.0.1s-freebsd PHP/5.6.30',
               'mw2097.codfw.wmnet', 'mw2233.codfw.wmnet', 'fbs',
               'mw2199.codfw.wmnet', 'mw2255.codfw.wmnet', 'mw2228.codfw.wmnet',
               'Apache/2.2.31 (Unix) mod_ssl/2.2.31 OpenSSL/1.0.1e-fips mod_bwlimited/1.4
        mod_fcgid/2.3.9',
               'gunicorn/19.7.1',
               'Apache/2.2.31 (Unix) mod_ssl/2.2.31 OpenSSL/0.9.8e-fips-rhel5 mod_bwlimite
        d/1.4',
               'Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16',
               'mw2241.codfw.wmnet',
               'Apache/1.3.33 (Unix) mod_ssl/2.8.24 OpenSSL/0.9.7e-p1 PHP/4.4.8',
               'lighttpd', 'mw2230.codfw.wmnet',
               'Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips', 'AkamaiGHost',
               'mw2240.codfw.wmnet', 'nginx/1.10.2', 'PWS/8.2.0.7', 'nginx/1.2.1',
               'nxfps',
               'Apache/2.2.16 (Unix) mod_ssl/2.2.16 OpenSSL/0.9.8e-fips-rhel5 mod_auth_pas
        sthrough/2.1 mod_bwlimited/1.4',
               'Play', 'mw2185.codfw.wmnet',
               'Apache/2.4.10 (Unix) OpenSSL/1.0.1k',
               'Apache/Not telling (Unix) AuthTDS/1.1',
               'Apache/2.2.11 (Unix) PHP/5.2.6', 'Scratch Web Server',
               'marrakesh 1.12.2', 'nginx/0.8.35', 'mw2182.codfw.wmnet',
               'squid/3.3.8', 'nginx/1.10.0', 'Nginx (OpenBSD)',
               'Zope/(2.13.16; python 2.6.8; linux2) ZServer/1.1',
               'Apache/2.2.26 (Unix) mod_ssl/2.2.26 OpenSSL/0.9.8e-fips-rhel5 mod_bwlimite
        d/1.4 PHP/5.4.26',
               'Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8e-fips-rhel5 PHP/5.3.10',
               'Apache/2.2.27 (Unix) OpenAM Web Agent/4.0.1-1 mod_ssl/2.2.27 OpenSSL/1.0.1
        p PHP/5.3.28',
               'mw2104.codfw.wmnet', '.V01 Apache', 'mw2110.codfw.wmnet',
               'Apache/2.4.6 (Unix) mod_jk/1.2.37 PHP/5.5.1 OpenSSL/1.0.1g mod_fcgid/2.3.
        9',
               'mw2176.codfw.wmnet', 'mw2187.codfw.wmnet', 'mw2106.codfw.wmnet',
               'Microsoft-IIS/7.0',
               'Apache/1.3.42 Ben-SSL/1.60 (Unix) mod_gzip/1.3.26.1a mod_fastcgi/2.4.6 mod
        _throttle/3.1.2 Chili!Soft-ASP/3.6.2 FrontPage/5.0.2.2635 mod_perl/1.31 PHP/4.4.
        9',
               'Aeria Games & Entertainment', 'nginx/1.6.3 + Phusion Passenger',
               'Apache/2.4.10 (Debian) PHP/5.6.30-0+deb8u1 mod_perl/2.0.9dev Perl/v5.20.
        2',
               'mw2173.codfw.wmnet',
               'Apache/2.4.6 (Red Hat Enterprise Linux) OpenSSL/1.0.1e-fips mod_fcgid/2.3.
        9 Communique/4.2.0',
               'Apache/2.2.15 (CentOS) DAV/2 mod_ssl/2.2.15 OpenSSL/1.0.1e-fips PHP/5.3.
        3',
               'Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/7.0.14',
               'mw2198.codfw.wmnet', 'mw2172.codfw.wmnet', 'nginx/1.2.6',
               'Apache/2.4.6 (Unix) mod_jk/1.2.37',
               'Apache/2.4.25 (Unix) OpenSSL/1.0.1e-fips mod_bwlimited/1.4',
               'nginx/1.4.4', 'Cowboy', 'mw2113.codfw.wmnet',
               'Apache/2.2.14 (Unix) mod_ssl/2.2.14 OpenSSL/0.9.8a',
               'Apache/2.4.10 (Ubuntu)', 'mw2224.codfw.wmnet',
               'mw2171.codfw.wmnet', 'mw2257.codfw.wmnet', 'mw2226.codfw.wmnet',
               'DMS/1.0.42', 'nginx/1.6.3', 'Application-Server',
               'Apache/2.4.6 (CentOS) mod_fcgid/2.3.9 PHP/5.6.30',
               'mw2177.codfw.wmnet', 'lighttpd/1.4.28', 'mw2197.codfw.wmnet',
               'Apache/2.2.31 (FreeBSD) PHP/5.4.15 mod_ssl/2.2.31 OpenSSL/1.0.2d DAV/2',
               'Apache/2.2.26 (Unix) mod_ssl/2.2.26 OpenSSL/1.0.1e-fips DAV/2 mod_bwlimite
        d/1.4',
               'Apache/2.2.24 (Unix) DAV/2 PHP/5.3.26 mod_ssl/2.2.24 OpenSSL/0.9.8y',
               'mw2178.codfw.wmnet', '294', 'Microsoft-IIS/6.0', 'nginx/1.7.4',
               'Apache/2.2.22 (Debian) mod_python/3.3.1 Python/2.7.3 mod_ssl/2.2.22 OpenSS
        L/1.0.1t',
               'Apache/2.4.16 (Ubuntu)', 'www.lexisnexis.com  9999',
               'nginx/0.8.38', 'mw2238.codfw.wmnet', 'Pizza/pepperoni',
               'XXXXXXXXXXXXXXXXXXXXXX', 'MI', 'Roxen/5.4.98-r2',
               'Apache/2.2.31 (Unix) mod_ssl/2.2.31 OpenSSL/1.0.1e-fips mod_bwlimited/1.
        4',
               'nginx/1.9.13', 'mw2180.codfw.wmnet', 'Apache/2.2.14 (Ubuntu)',
               'ebay server', 'nginx/0.8.55', 'Apache/2.2.10 (Linux/SUSE)',
               'nginx/1.7.12',
               'Apache/2.0.63 (Unix) mod_ssl/2.0.63 OpenSSL/0.9.8e-fips-rhel5 mod_auth_pas
        sthrough/2.1 mod_bwlimited/1.4 PHP/5.3.6',
               'Boston.com Frontend', 'My Arse', 'IdeaWebServer/v0.80',
               'Apache/2.4.17 (Unix) OpenSSL/1.0.1e-fips PHP/5.6.19',
               'Microsoft-IIS/7.5; litigation_essentials.lexisnexis.com  9999',
               'Apache/2.2.16 (Debian)'], dtype=object)
```

In [ ]:

```
#Fill in nulls with unknown.
df_copy['server'].fillna("Unknown", inplace=True)
```

In [ ]:

```
#Clean up the 'charset' column: replace the dashes with spaces and make it Title ca
df_copy['server'] = df_copy['server'].str.lower()
```

In [ ]:

```
#Grouping values with the least count into one bin "Other" to reduce number of uniq
series = pd.value_counts(df_copy.server)
```

```
mask = (series/series.sum() * 100).lt(1)
df_copy['server'] = np.where(df_copy['server'].isin(series[mask].index),'other',df_
```

In [ ]:

```
#Inspecting changes.
df_copy['server'].unique()
```

Out[ ]:

```
array(['nginx', 'other', 'microsoft-httpapi/2.0', 'unknown', 'apache',
       'nginx/1.12.0', 'cloudflare-nginx', 'microsoft-iis/7.5',
       'apache/2.2.15 (centos)', 'gse', 'ats', 'server',
       'youtubefrontend', 'apache-coyote/1.1'], dtype=object)
```

In [ ]:

```
 # Further bucketing and cleaning up the server column.

def standardize_server(server_string):
    if 'apache' in server_string:
        return 'apache'
    if 'nginx' in server_string:
        return 'nginx'
    if 'microsoft' in server_string:
        return 'microsoft-IIS'

    return server_string

#Applying function.
df_copy['standardized_server'] = df_copy['server'].apply(standardize_server)
df_copy['standardized_server'] = df_copy['standardized_server'].str.replace('-', '

print(df_copy[['server', 'standardized_server']])
```

```
                     server standardized_server
0                     nginx               nginx
1                     other               other
2     microsoft-httpapi/2.0       microsoft IIS
3                     nginx               nginx
4                   unknown             unknown
...                     ...                 ...
1776                 apache              apache
1777                 apache              apache
1778                  other               other
1779       cloudflare-nginx               nginx
1780                  other               other

[1781 rows x 2 columns]
```

In [ ]:

```
#Final inspection of data.
df_copy['standardized_server'].unique()
```

Out[ ]:

```
array(['nginx', 'other', 'microsoft IIS', 'unknown', 'apache', 'gse',
       'ats', 'server', 'youtubefrontend'], dtype=object)
```

In [ ]:

```
#Checking values and their amounts.
df_copy['standardized_server'].value_counts()
```

Out[ ]:

```
standardized_server
other               499
apache              431
nginx               341
unknown             176
microsoft IIS       164
gse                  49
server               49
youtubefrontend      42
ats                  30
Name: count, dtype: int64
```

In [ ]:

```
#Inspecting other columns to clean.
df_copy.isna().sum()
```

Out[ ]:

```
url                           0
url_length                    0
number_special_characters     0
server                        0
content_length              812
whois_country               306
whois_statepro              362
whois_regdate               127
whois_updated_date          139
tcp_conversation_exchange     0
dist_remote_tcp_port          0
remote_ips                    0
app_bytes                     0
source_app_packets            0
remote_app_packets            0
source_app_bytes              0
remote_app_bytes              0
app_packets                   0
dns_query_times               1
type                          0
charset_imputed               0
standardized_server           0
dtype: int64
```

In [ ]:

```
#Content length has the most null values and has a very large spread. Filling with
#The median may also not work well since its 1853, and imputing with KNN may not wo
df_copy[['content_length']].describe()
```

Out[ ]:

	content_length
count	969.000000
mean	11726.927761
std	36391.809051
min	0.000000
25%	324.000000
50%	1853.000000
75%	11323.000000
max	649263.000000

In [ ]:

```
df_copy['content_length'] = df_copy['content_length'].interpolate()
```

In [ ]:

```
df_copy[['content_length']].describe()
```

Out[ ]:

	content_length
count	1781.000000
mean	13497.243964
std	38415.552697
min	0.000000
25%	603.000000
50%	4714.750000
75%	12578.500000
max	649263.000000

In [ ]:

```
#Inspecting other columns to clean and seeing if we filled in content_length succes
df_copy.isna().sum()
```

Out[ ]:

```
url                            0
url_length                     0
number_special_characters      0
server                         0
content_length                 0
whois_country                306
whois_statepro               362
whois_regdate                127
whois_updated_date           139
tcp_conversation_exchange      0
dist_remote_tcp_port           0
remote_ips                     0
app_bytes                      0
source_app_packets             0
remote_app_packets             0
source_app_bytes               0
remote_app_bytes               0
app_packets                    0
dns_query_times                1
type                           0
charset_imputed                0
standardized_server            0
dtype: int64
```

In [ ]:

```
#Inspecting the whois_country. There is some cleaning that should be done - multipl
df_copy['whois_country'].unique()
```

Out[ ]:

```
array([nan, 'US', 'SC', 'GB', 'UK', 'RU', 'AU', 'CA', 'PA', 'se', 'IN',
       'LU', 'TH', "[u'GB'; u'UK']", 'FR', 'NL', 'UG', 'JP', 'CN', 'SE',
       'SI', 'IL', 'ru', 'KY', 'AT', 'CZ', 'PH', 'BE', 'NO', 'TR', 'LV',
       'DE', 'ES', 'BR', 'us', 'KR', 'HK', 'UA', 'CH', 'United Kingdom',
       'BS', 'PK', 'IT', 'Cyprus', 'BY', 'AE', 'IE', 'UY', 'KG'],
      dtype=object)
```

In [ ]:

```
df_copy['whois_country'].value_counts()
```

Out[ ]:

```
whois_country
US                 1103
CA                   84
ES                   63
AU                   35
PA                   21
GB                   19
JP                   11
UK                   10
CN                   10
IN                   10
FR                    9
CZ                    9
NL                    6
CH                    6
[u'GB'; u'UK']        5
KR                    5
PH                    4
BS                    4
ru                    4
AT                    4
HK                    3
us                    3
TR                    3
BE                    3
DE                    3
SC                    3
KY                    3
SE                    3
BR                    2
UY                    2
Cyprus                2
SI                    2
UA                    2
RU                    2
IL                    2
NO                    2
KG                    2
TH                    1
se                    1
LV                    1
LU                    1
United Kingdom        1
UG                    1
PK                    1
IT                    1
BY                    1
AE                    1
IE                    1
Name: count, dtype: int64
```

In [ ]:

```python
#Function to replace the strange values in the column.
def replace(x):
    if x == "[u'GB'; u'UK']"or x=="United Kingdom" or x=="UK":
        return "GB"
    elif x == "Cyprus":
        return "CY"
    elif x == "us":
        return "US"
    elif x == "ru":
        return "RU"
    elif x == "se":
        return "SE"
    else:
        return x

df_copy["whois_country"] = list(map(lambda x: replace(x), df_copy["whois_country"])
```

In [ ]:
```
#Sanity check
df_copy['whois_country'].unique()
```

Out[ ]:
```
array([nan, 'US', 'SC', 'GB', 'RU', 'AU', 'CA', 'PA', 'SE', 'IN', 'LU',
       'TH', 'FR', 'NL', 'UG', 'JP', 'CN', 'SI', 'IL', 'KY', 'AT', 'CZ',
       'PH', 'BE', 'NO', 'TR', 'LV', 'DE', 'ES', 'BR', 'KR', 'HK', 'UA',
       'CH', 'BS', 'PK', 'IT', 'CY', 'BY', 'AE', 'IE', 'UY', 'KG'],
      dtype=object)
```

In [ ]:
```
#Filling the NA as 'other' category.
df_copy['whois_country'].fillna("Other", inplace=True)
```

In [ ]:
```
#Sanity check
df_copy['whois_country'].unique()
```

Out[ ]:
```
array(['Other', 'US', 'SC', 'GB', 'RU', 'AU', 'CA', 'PA', 'SE', 'IN',
       'LU', 'TH', 'FR', 'NL', 'UG', 'JP', 'CN', 'SI', 'IL', 'KY', 'AT',
       'CZ', 'PH', 'BE', 'NO', 'TR', 'LV', 'DE', 'ES', 'BR', 'KR', 'HK',
       'UA', 'CH', 'BS', 'PK', 'IT', 'CY', 'BY', 'AE', 'IE', 'UY', 'KG'],
      dtype=object)
```

In [ ]:
```
#Inspecting other columns to clean and seeing if we filled in whois_country success
df_copy.isna().sum()
```

Out[ ]:

```
url                            0
url_length                     0
number_special_characters      0
server                         0
content_length                 0
whois_country                  0
whois_statepro               362
whois_regdate                127
whois_updated_date           139
tcp_conversation_exchange      0
dist_remote_tcp_port           0
remote_ips                     0
app_bytes                      0
source_app_packets             0
remote_app_packets             0
source_app_bytes               0
remote_app_bytes               0
app_packets                    0
dns_query_times                1
type                           0
charset_imputed                0
standardized_server            0
dtype: int64
```

In [ ]:

```python
#statepro
df_copy['whois_statepro'].unique()
```

```
Out[ ]: array([nan, 'AK', 'TX', 'Mahe', 'CO', 'FL', 'Kansas',
               'Novosibirskaya obl.', 'CA', 'Tennessee', 'Vi', 'OR', 'Texas',
               'ALBERTA', 'PANAMA', 'Arizona', 'WI', 'Oregon', 'Andhra Pradesh',
               'AB', 'Tamil Nadu', 'VA', 'NY', 'quebec', 'MA', 'ON', 'New Mexico',
               'British Columbia', 'Massachusetts', 'California', 'bangkok',
               'WEST MIDLANDS', 'TEXAS', 'WC1N', 'Kentucky', 'MD', 'NEW YORK',
               'Washington', 'Colorado', 'PA', 'LA', 'WA', 'Queensland', 'MOSCOW',
               'UK', 'P', 'NH', 'Pennsylvania', 'UTTAR PRADESH', 'NC', 'kireka',
               'IL', 'Missouri', 'Osaka', 'QC', 'Michigan', 'Maryland', 'Ontario',
               'South Carolina', 'Zhejiang', 'New York', 'QLD', 'NJ', 'GA', 'MO',
               'HR', 'ab', 'Greater London', 'Illinois', '--', 'Fukuoka', 'BC',
               'AL', 'Krasnoyarsk', 'MAINE', 'Virginia', 'MH', 'GRAND CAYMAN',
               'Austria', 'DE', 'shandong', 'AZ', 'PRAHA', 'beijingshi',
               'liaoningsheng', 'North Carolina', 'OH', 'Manila', 'Utah', 'MI',
               'NSW', 'UT', 'New South Wales', 'WV', 'Ohio', 'RIX', 'TR', 'nj',
               'Panama', 'SK', 'ca', 'Alicante', 'New Jersey', 'Vic', 'ME',
               'worcs', 'Maine', 'London', 'Karnataka', 'Quebec', 'Indiana',
               'NEW SOUTH WALES', '6110021', 'Not Applicable', 'Peterborough',
               'CT', 'Minnesota', 'NOT APPLICABLE', 'VIC', 'Noord-Holland',
               'CALIFORNIA', 'Nevada', 'Nebraska', 'ILOCOS NORTE R3', 'NV', 'MB',
               'Florida', 'Central', 'Maharashtra', 'widestep@mail.ru', 'VERMONT',
               'ZH', 'hunansheng', 'NONE', 'Wisconsin', 'UTAH', 'Utr', 'Bei Jing',
               '-', 'Manitoba', 'ALABAMA', 'New Providence', 'Punjab', 'qc',
               'Connecticut', 'il', 'Berlin', 'INDAL', 'RM', 'va', 'ny',
               'MAHARASHTR', 'ONTARIO', 'Haryana', 'MIDDLESEX', 'Rogaland',
               'District of Columbia', 'DC', 'HANTS', 'Zug', 'VT', 'TN',
               'ANTWERP', 'CH', 'TOKYO-TO', 'Saskatchewan', 'Alabama', 'Tottori',
               'Arkansas', 'OK', 'Dubai', 'KS', 'Barcelona', 'CO. DUBLIN',
               'Metro Manila', 'Montevideo', 'KG', 'FLORIDA', 'Other', 'QUEBEC',
               'bc', 'Paris'], dtype=object)
```

```
In [ ]: #Cleaning up data and nulls.
        def replace_state(x):
            if x == "California"or x=="CALIFORNIA":
                return "CA"
            elif x == "Arizona":
                return "AZ"
            elif x == "New York" or x=="NEW YORK":
                return "NY"
            elif x == "Ohio":
                return "OH"
            elif x == "Utah":
                return "UT"
            elif x == "None":
                return "NA"
            elif x == "Texas":
                return "TX"
            elif x == "Washington":
                return "WA"
            elif x == "va":
                return "VA"
            elif x == "Illinois" or x=="il":
                return "IL"
            elif x == "District of Columbia" or x=="DC" or x=="Maryland":
                return "MD"
            elif x == "New Jersey":
```

```
        return "NJ"
    elif x == "Maine" or x=="MAINE":
        return "ME"
    elif x == "Quebec" or x=="QUEBEC" or x=="qc" or x=="quebec":
        return "QC"
    elif x == "Missouri":
        return "MO"
    elif x == "Nevada":
        return "NV"
    elif x == "WC1N" or x=="Greater London" or x=="UK" or x=="WEST MIDLANDS" or x==
        return "England"
    elif x == "Pennsylvania":
        return "PA"
    elif x == "Florida" or x=="FLORIDA":
        return "FL"
    elif x == "PANAMA":
        return "Panama"
    else:
        return x

df_copy["whois_statepro"] = list(map(lambda x: replace_state(x), df_copy["whois_sta
```

In [ ]:

```
#Inspecting top 21 values.
df_copy["whois_statepro"].value_counts()[:21]
```

Out[ ]:

```
whois_statepro
CA           430
NY            87
WA            75
FL            67
AZ            64
Barcelona     62
ON            45
UT            42
NV            33
IL            28
PA            28
CO            24
England       22
MO            22
MA            22
Panama        21
OH            21
TX            19
VA            18
NJ            16
QC            15
Name: count, dtype: int64
```

In [ ]:

```
#Grouping values with the least count into one bin "Other" to reduce number of uniq
counts = df_copy['whois_statepro'].value_counts()
df_copy['whois_statepro'] = np.where(df_copy['whois_statepro'].isin(counts[counts <
```

In [ ]:

```
#Fill null's with "Other".
df_copy['whois_statepro'].fillna("Other", inplace=True)
```

In [ ]:
```
df_copy['whois_statepro'].unique()
```

Out[ ]:
```
array(['Other', 'TX', 'CO', 'FL', 'CA', 'Panama', 'AZ', 'VA', 'NY', 'QC',
       'MA', 'ON', 'England', 'WA', 'PA', 'IL', 'MO', 'NJ', 'OH', 'UT',
       'NV', 'Barcelona'], dtype=object)
```

In [ ]:
```
#Inspecting other columns to clean and seeing if we filled in whois_country success
df_copy.isna().sum()
```

Out[ ]:
```
url                            0
url_length                     0
number_special_characters      0
server                         0
content_length                 0
whois_country                  0
whois_statepro                 0
whois_regdate                127
whois_updated_date           139
tcp_conversation_exchange      0
dist_remote_tcp_port           0
remote_ips                     0
app_bytes                      0
source_app_packets             0
remote_app_packets             0
source_app_bytes               0
remote_app_bytes               0
app_packets                    0
dns_query_times                1
type                           0
charset_imputed                0
standardized_server            0
dtype: int64
```

In [ ]:
```
#Inspect the regdate column
df_copy['whois_regdate'].value_counts()
```

Out[ ]:
```
whois_regdate
17/09/2008 0:00    62
13/01/2001 0:12    59
31/07/2000 0:00    47
15/02/2005 0:00    41
29/03/1997 0:00    33
                   ..
23/11/1994 0:00     1
30/08/2015 0:00     1
30/04/2009 0:00     1
27/11/2006 0:00     1
14/11/2008 0:00     1
Name: count, Length: 890, dtype: int64
```

In [ ]:
```
#Inspection
df_copy['whois_regdate'].unique()
```

```
Out[ ]:  array(['10/10/2015 18:21', nan, '7/10/1997 4:00', '12/05/1996 0:00',
                '3/08/2016 14:30', '29/07/2002 0:00', '18/03/1997 0:00',
                '8/11/2014 7:41', '14/09/2007 0:00', '22/11/2016 0:00',
                '11/10/2002 0:00', '14/11/2002 0:00', '16/07/2000 0:00',
                '25/05/2013 0:00', '9/08/1999 0:00', '15/09/2013 0:00',
                '3/07/1999 0:00', '2/11/2003 0:00', '12/08/2008 22:10',
                '21/05/2009 0:00', '1/08/2002 0:00', '13/01/2005 0:00',
                '18/05/2005 19:41', '4/01/2001 0:00', '28/02/2008 10:58',
                '8/12/2006 0:00', '16/06/2000 0:00', '13/10/2000 0:00',
                '31/12/1999 0:00', '30/07/1996 0:00', '9/05/2008 0:00',
                '23/04/1999 0:00', '4/02/1997 0:00', '13/02/2003 0:00',
                '17/05/2008 0:00', '30/05/2002 0:00', '20/10/2005 0:00',
                '7/01/2006 0:00', '5/03/1996 5:00', '23/03/1995 0:00',
                '10/01/1998 0:00', '27/04/2016 0:00', '7/04/2011 0:00',
                '26/02/2009 0:00', '3/07/2002 0:00', '21/02/1995 0:00',
                '4/07/2007 0:00', '2/07/1998 0:00', '10/03/2005 0:00',
                '15/12/2004 0:00', '25/02/2008 0:00', '27/01/2005 0:00',
                '14/09/2006 0:00', '30/04/2010 14:12', '22/04/1997 0:00',
                '16/07/2016 0:00', '27/11/2016 19:09', '30/10/2009 0:00',
                '12/03/2009 21:00', '9/03/2000 17:50', '30/05/2008 0:00',
                '25/09/2000 0:00', '9/04/2002 0:00', '11/01/1997 0:00',
                '11/06/2000 0:00', '13/02/2002 19:55', '19/12/2007 0:00',
                '6/11/1997 0:00', '21/01/2000 0:00', '27/04/2009 0:00',
                '11/10/2000 0:00', '4/08/1998 0:00', '31/05/2000 0:00',
                '23/10/1999 0:00', '23/06/2010 0:00', '9/03/2000 0:00',
                '13/04/1994 0:00', '9/06/2010 0:00', '29/04/2009 0:00',
                '19/01/2015 0:00', '11/11/2015 0:00', '22/03/2017 0:00',
                '3/11/2009 0:00', '19/07/2010 20:03', '28/04/1997 0:00',
                '4/03/1996 0:00', '24/10/2007 0:00', '21/10/2004 0:00',
                '2002-03-20T23:59:59.0Z', '29/03/1997 0:00', '10/06/2008 0:00',
                '30/11/1999 0:00', '30/08/2004 0:00', '11/11/1996 0:00',
                '2/10/1995 4:00', '28/06/2011 0:00', '16/08/2016 0:00',
                '9/05/2000 17:31', '31/07/2000 0:00', '14/05/1999 0:00',
                '24/04/2009 0:00', '6/08/1998 0:00', '22/02/1996 0:00',
                '15/06/2007 0:00', '21/09/2009 0:00', '20/01/1995 0:00',
                '28/03/2006 0:00', '28/09/2007 16:06', '4/02/2017 0:00',
                '7/03/1996 0:00', '4/04/2003 0:00', '26/08/2015 0:00',
                '22/07/2004 0:00', '30/11/2004 0:00', '5/03/2008 0:00',
                '8/05/2003 0:00', '2/11/2010 21:52', '23/05/2006 0:00',
                '2/12/1998 0:00', '15/04/1999 0:00', '30/05/2003 0:00',
                '10/01/2002 0:57', '17/03/2006 0:00', '1/03/1999 0:00',
                '1/07/2015 0:00', '22/01/2006 0:00', '6/03/1996 5:00',
                '8/02/2013 0:00', '15/04/2003 0:00', '24/10/1996 0:00',
                '30/04/2016 0:00', '30/12/1999 0:00', '13/07/2001 0:00',
                '29/05/1996 0:00', '25/02/2004 0:00', '18/01/1995 0:00',
                '25/10/2014 0:00', '20/05/1996 0:00', '14/07/2000 0:00',
                '1/12/1996 5:00', '11/11/2000 0:00', '4/09/2004 0:00',
                '9/09/2005 13:44', '7/08/1996 0:00', '30/12/2005 0:00',
                '4/04/2006 0:00', '25/05/2000 0:00', '5/01/1996 0:00',
                '9/07/2008 0:00', '10/05/2006 20:00', '29/07/2006 0:00',
                '27/09/2010 0:00', '31/10/2008 0:00', '3/03/2000 0:00',
                '2/04/2008 0:00', '6/09/2016 0:00', '11/02/1997 0:00',
                '6/08/1995 0:00', '23/10/2008 0:00', '19/02/2009 0:00',
                '30/08/1998 0:00', '14/12/2006 0:00', '22/03/2005 3:36',
                '26/06/2009 0:00', '9/02/2000 14:17', '8/06/2007 0:00',
                '1/11/1994 0:00', '8/08/2013 0:00', '20/04/2000 0:00',
                '12/11/2016 0:00', '8/02/2000 0:00', '25/09/2003 0:00',
                '13/01/2001 0:12', '10/05/2002 0:00', '7/06/2005 0:00',
                '19/08/2012 0:00', '28/04/2003 0:00', '12/03/1999 0:00',
                '30/03/2011 0:00', '14/01/2008 0:00', '26/03/2000 0:00',
                '2/11/2002 0:00', '10/04/2010 0:00', '6/05/2010 16:44',
                '17/08/1999 12:43', '4/10/2006 0:00', '9/03/2007 0:00',
                '31/05/1996 0:00', '29/07/2004 0:00', '17/01/2005 0:00',
                '1/06/2005 0:00', '22/12/1999 0:00', '11/08/2002 0:00',
                '17/11/2003 0:00', '14/08/2008 0:00', '22/08/2010 0:00',
                '8/07/1996 0:00', '21/03/2014 0:00', '22/10/1998 0:00',
                '8/07/2010 0:00', '17/07/2009 0:00', '24/05/2001 20:47',
                '27/06/2004 0:00', '4/03/2004 0:00', '1/06/2004 0:00',
                '13/12/1995 0:00', '10/11/2000 0:00', '30/10/2003 0:00',
                '9/05/1998 0:00', '5/10/2006 0:00', '14/10/2005 0:00',
                '8/09/2003 0:00', '13/07/2008 0:00', '22/08/1996 0:00',
                '1/05/2002 0:00', '10/06/2005 0:00', '7/09/2000 0:00',
                '31/12/2015 0:00', '8/12/2007 0:00', '6/04/2008 0:00',
                '16/07/1998 0:00', '9/02/2001 0:00', '28/12/1999 0:00',
                '21/01/2008 0:00', '27/02/1996 0:00', '7/10/2006 0:00',
                '18/12/2004 0:00', '9/02/2009 0:00', '4/11/1998 0:00',
                '4/11/2016 0:00', '15/04/2004 0:00', '2/07/2010 0:00',
                '12/11/2007 20:49', '22/02/2006 0:00', '5/06/2000 0:00',
                '9/12/1996 0:00', '14/10/1999 0:00', '10/05/2006 23:10',
                '22/02/2001 0:00', '11/09/1997 4:00', '16/10/2000 0:00',
                '8/06/2006 0:00', '18/04/2001 0:00', '5/08/1993 0:00',
                '17/02/2011 0:00', '24/10/1999 0:00', '2/11/1996 0:00',
                '14/02/2000 0:00', '14/09/2009 8:39', '24/05/2000 0:00',
                '18/07/2005 0:00', '28/10/1999 0:00', '25/02/2006 0:00',
                '12/06/2007 0:00', '18/04/1995 0:00', '25/04/2000 0:00',
                '5/12/1997 0:00', '24/09/1996 0:00', '10/01/1992 0:00',
                '17/07/2001 0:00', '14/07/2011 0:00', '21/09/2001 0:00',
                '5/06/2003 0:00', '5/12/1996 0:00', '19/02/2002 1:02',
                '28/11/1999 0:00', '26/05/1997 0:00', '5/07/2008 0:00',
                '7/04/2007 0:00', '9/04/2011 16:13', '29/01/2004 16:01',
                '17/02/1999 5:00', '27/02/2000 15:05', '21/07/1995 0:00',
                '8/08/2002 0:00', '15/02/2005 0:00', '8/01/2009 10:56',
                '1/07/2007 5:40', '21/10/2006 0:00', '30/08/2000 0:00',
                '14/09/1995 0:00', '14/09/2000 8:47', '17/02/2006 0:00',
                '10/07/2002 0:00', '4/09/1996 4:00', '15/05/2005 0:00',
                '8/01/2004 0:00', '10/11/2009 0:00', '26/11/1996 0:00',
                '2/08/2010 0:00', '24/08/2003 0:00', '19/07/2010 0:00',
                '28/10/2008 0:00', '24/06/2003 0:00', '28/02/2008 0:00',
                '22/09/1993 0:00', '29/09/1994 0:00', '7/11/2016 0:00',
                '30/07/2009 4:01', '10/01/2007 15:27', '19/10/2005 0:00',
                '13/09/2001 23:03', '2/10/2005 0:00', '27/08/2002 0:00',
                '28/09/1996 0:00', '5/06/2006 0:00', '22/03/2006 0:00',
                '20/10/1998 0:00', '23/04/2002 18:56', '26/09/2005 0:00',
                '4/01/1995 0:00', '30/07/2004 0:00', '5/11/2008 0:00',
                '29/12/2005 0:00', '17/02/2009 0:00', '2/12/2005 0:00',
                '8/11/1998 0:00', '12/02/2002 0:00', '8/04/1996 0:00',
                '8/05/1996 0:00', '2/03/1999 0:00', '17/12/2007 0:00',
                '3/06/2000 0:00', '21/11/2016 0:00', '8/09/2007 0:00',
                '25/10/2006 0:00', '18/10/1994 0:00', '23/10/1996 0:00',
                '4/11/2003 0:00', '30/07/2002 0:00', '9/12/2008 0:00',
                '9/03/2010 0:00', '4/07/2001 23:21', '8/03/2016 0:00',
                '23/02/2010 0:00', '6/07/2016 14:30', '16/01/2016 0:00',
                '13/06/1997 0:00', '22/07/2005 18:52', '7/04/2002 20:00',
                '5/03/2013 20:51', '24/05/2004 0:00', '30/01/2006 0:00',
                '19/12/1997 0:00', '30/11/2006 0:00', '14/04/1999 0:00',
                '27/05/2008 0:00', '8/05/2004 20:53', '16/08/1999 0:00',
                '4/08/1995 0:00', '18/07/2002 0:00', '23/11/1994 0:00',
                '30/08/2015 0:00', '30/04/2009 0:00', '27/11/2006 0:00',
                '13/02/2017 0:00', '29/09/2015 0:00', '19/08/2004 0:00',
                '18/03/2016 0:00', '6/12/2010 0:00', '23/06/2003 0:00',
                '3/08/2004 0:00', '14/05/2002 0:00', '26/04/2001 0:00',
                '9/05/2004 18:06', '5/11/1997 5:00', '8/12/1995 0:00',
                '27/06/2001 0:00', '13/11/2002 0:00', '30/01/1996 0:00',
                '16/07/1999 0:00', '23/06/2006 0:00', '26/09/2003 0:00',
                '10/04/2002 0:00', '11/11/2006 0:00', '20/04/1995 4:00',
                '3/05/2015 0:00', '4/02/2005 0:00', '16/09/1996 4:00',
                '20/07/2006 0:00', '9/12/2003 0:00', '12/03/2008 0:00',
                '13/09/2000 0:00', '5/02/1999 0:00', '10/05/2000 0:00',
                '25/01/1995 0:00', '5/10/2005 0:00', '28/06/2006 0:00',
                '25/06/1997 0:00', '9/01/1998 0:00', '17/03/2000 0:00',
                '9/10/2016 17:08', '29/02/2008 0:00', '18/05/2001 0:00',
                '30/10/2002 0:00', '19/12/2000 0:00', '13/05/1998 0:00',
                '9/08/2009 0:00', '29/07/2007 0:00', '15/07/2009 0:00',
                '27/09/2000 0:00', '8/08/2001 0:00', '27/11/2002 0:00',
                '12/03/2009 1:58', '8/11/1993 0:00', '25/07/1998 0:00',
                '16/11/1999 0:00', '20/07/1998 4:00', '7/04/2003 0:00',
                '4/06/2009 0:00', '29/11/2000 0:00', '3/02/1999 0:00',
                '18/10/2000 0:00', '6/10/2005 0:00', '22/09/2009 0:00',
                '18/01/1994 0:00', '13/09/1996 4:00', '21/07/2008 0:00',
                '30/01/2007 0:00', '11/03/2017 0:00', '25/08/2004 0:00',
                '16/05/2001 0:00', '22/11/2003 0:00', '14/10/2016 0:00',
                '26/07/1995 0:00', '22/04/1999 0:00', '26/05/2006 0:00',
                '6/08/2002 0:00', '7/03/2000 0:00', '8/07/2000 0:00',
                '3/12/1999 0:00', '15/11/1998 0:00', '11/11/2007 0:00',
                '19/07/2004 0:00', '2/09/2003 0:00', '28/08/1998 0:00',
                '4/10/2003 0:00', '2/10/2007 0:00', '24/04/2007 0:00',
                '15/12/2010 0:00', '1/06/2001 0:00', '22/08/2007 0:00',
                '26/01/2001 0:00', '6/06/2007 0:00', '19/09/2002 0:00',
                '28/07/1995 0:00', '8/05/2001 0:00', '18/10/2002 0:00',
                '18/10/2004 0:00', '5/11/1997 0:00', '28/03/2000 0:00',
                '12/04/2000 0:00', '28/11/2006 0:00', '24/02/1996 0:00',
                '21/12/2005 0:00', '1/07/1998 0:00', '5/05/2004 0:00',
                '2/12/2004 0:00', '10/08/1998 0:00', '13/04/2005 0:00',
                '14/08/2000 0:00', '25/06/2002 0:00', '22/06/2014 0:00',
                '24/05/2015 0:00', '5/08/1999 0:00', '21/08/1998 0:00',
                '28/08/2003 0:00', '9/03/2004 0:00', '3/01/2002 0:00',
                '22/03/2009 0:00', '3/07/2000 17:33', '2/02/1996 0:00',
                '19/02/1997 0:00', '23/08/2010 0:00', '23/09/2009 0:00',
                '7/10/2005 16:25', '20/09/1995 0:00', '30/01/2003 0:00',
                '22/11/2000 0:00', '6/11/2001 1:42', '11/11/1999 0:00',
                '26/07/1990 0:00', '26/07/2003 0:00', '16/05/1995 0:00',
                '26/04/2004 0:00', '9/03/2006 0:00', '23/07/2011 0:00',
                '9/12/2002 0:00', '2/04/1999 0:00', '21/05/2007 21:12',
                '4/10/1996 0:00', '1/05/1996 0:00', '31/01/1995 5:00',
                '22/02/2007 0:00', '6/12/1994 0:00', '2/01/2003 0:00',
                '6/10/1999 0:00', '11/06/2003 0:00', '30/12/1996 0:00',
                '27/02/1998 5:00', '11/01/1999 0:00', '4/01/2007 0:00',
                '4/11/2004 22:27', '21/01/2007 22:31', '20/11/2007 0:00',
                '12/12/2003 0:00', '4/05/2001 0:00', '9/09/2002 0:00',
                '15/02/2000 0:00', '1/03/2005 0:00', '21/02/1992 0:00',
                '15/10/2005 0:00', '1/06/1996 0:00', '5/03/2005 0:00',
                '14/04/1998 0:00', '2/06/1994 0:00', '31/05/1995 0:00',
                '2/03/1995 0:00', '18/01/2008 0:00', '22/03/1999 0:00',
                '27/05/1998 0:00', '17/09/1998 0:00', '7/05/1998 0:00',
                '5/11/2003 0:00', '23/10/1994 0:00', '24/05/2010 0:00',
                '14/10/2007 12:24', '25/11/2016 0:00', '30/09/2002 0:00',
                '9/02/2007 0:00', '10/02/1999 0:00', '7/08/1995 0:00',
                '21/05/1997 0:00', '10/01/2006 0:00', '31/08/2004 0:00',
                '5/12/2005 0:00', '27/05/2002 0:00', '5/05/2001 0:00',
                '1/03/2008 0:00', '16/02/2005 0:00', '17/12/2005 0:00',
                '2/10/2002 0:00', '6/03/1998 0:00', '29/09/2005 0:00',
                '17/07/2000 0:00', '27/06/2015 0:00', '19/02/2005 0:00',
                '7/11/1996 0:00', '14/12/1995 5:00', '6/11/1998 0:00',
                '29/09/2000 0:00', '30/12/2002 0:00', '15/09/2003 0:00',
                '15/09/1997 0:00', '21/05/1998 0:00', '4/01/1997 0:00',
                '25/07/2003 18:21', '25/10/2005 0:00', '29/12/1999 0:00',
                '1/09/1995 4:00', '16/03/2003 8:22', '22/07/2008 0:00',
                '27/09/2013 0:00', '5/11/1999 0:00', 'b', '13/08/2002 0:00',
                '1/09/1998 0:00', '8/12/1997 0:00', '16/12/1993 5:00',
                '16/02/2001 9:00', '25/02/1999 0:00', '6/08/2004 0:00',
                '18/05/2003 7:22', '4/02/2003 0:00', '9/08/2002 18:13',
                '29/12/2008 0:00', '31/07/2006 0:00', '16/11/1998 0:00',
                '18/04/2000 0:00', '20/04/2001 0:00', '28/09/2005 0:00',
                '31/05/2005 0:00', '3/08/1998 0:00', '28/08/2007 0:00',
                '19/12/2006 0:00', '16/01/2001 0:00', '12/12/1990 0:00',
                '19/07/2016 0:00', '13/11/2003 0:00', '2/12/2000 0:00',
                '7/04/2006 16:53', '6/05/2006 0:00', '12/09/2009 21:54',
                '4/10/2010 0:00', '18/04/2005 22:11', '1/10/2001 0:00',
                '14/07/1998 4:00', '15/08/1995 0:00', '8/09/2004 0:00',
                '26/01/2001 12:11', '17/07/2006 0:00', '17/05/2001 0:00',
                '7/09/1993 0:00', '7/06/2001 0:00', '13/08/2004 0:00',
                '16/06/1996 0:00', '6/11/2007 0:00', '14/06/1995 0:00',
                '23/03/1998 0:00', '21/04/2002 0:00', '7/07/2007 0:00',
                '31/12/2005 0:00', '17/03/1996 5:00', '19/02/2002 0:00',
                '28/04/2005 0:00', '21/06/1997 0:00', '24/04/1999 0:00',
                '24/01/2001 0:00', '23/05/1997 0:00', '12/09/2002 0:00',
                '20/02/2006 0:00', '21/11/1997 0:00', '19/03/1999 0:00',
                '1/09/2016 0:00', '9/08/1995 0:00', '15/04/2010 0:00',
                '4/12/2014 0:00', '16/07/1998 16:08', '10/06/1999 0:00',
                '28/08/2001 0:00', '12/10/2003 0:00', '14/10/2001 0:00',
                '29/09/1998 0:00', '13/06/1995 0:00', '24/09/2006 0:00',
                '13/08/2003 0:00', '12/01/1995 0:00', '10/09/2003 0:00',
                '9/04/2008 0:00', '25/10/2002 0:00', '21/05/1995 0:00',
                '20/01/1998 0:00', '9/08/1996 0:00', '27/01/1999 0:00',
                '15/08/2013 0:00', '10/08/1999 0:00', '27/05/1997 0:00',
                '13/03/2000 0:00', '3/04/2011 0:00', '16/12/2004 0:00',
                '4/06/2010 0:00', '11/10/2005 0:00', '25/10/2003 0:00',
                '22/12/2003 0:00', '26/01/2007 0:00', '19/10/2006 0:00',
                '21/06/2004 6:33', '3/12/1996 0:00', '9/02/2005 0:00',
                '27/03/1998 0:00', '2/05/2005 0:00', '20/03/2002 0:00',
                '15/03/2005 0:00', '5/12/1996 5:00', '10/10/2006 0:00',
                '1/08/2007 0:00', '3/06/1993 0:00', '22/11/1996 0:00',
                '27/04/2007 0:00', '7/04/2006 10:47', '30/03/1994 0:00',
                '12/02/1999 0:00', '16/09/1996 0:00', '27/05/2010 0:00',
                '24/07/2006 17:52', '5/03/1998 0:00', '28/01/2006 0:00',
                '9/10/2014 0:00', '1/02/2003 16:44', '24/10/2000 0:00',
                '29/08/2001 0:00', '26/07/2005 0:00', '8/02/1996 0:00',
                '2/10/2001 13:00', '12/10/1997 4:00', '16/09/2004 0:00',
                '27/07/2003 0:00', '15/07/1999 0:00', '28/07/2007 0:00',
                '10/02/2000 0:00', '19/07/2004 14:56', '2/05/1996 0:00',
                '24/09/2009 0:00', '7/03/2001 0:00', '2/01/2001 0:00',
                '26/01/2006 21:09', '21/12/2006 0:00', '16/11/2016 0:00',
                '13/05/2011 0:00', '26/05/1998 0:00', '8/04/2000 0:00',
                '5/11/2007 0:00', '19/04/2006 0:00', '1/06/2006 0:00', '0',
                '10/03/1998 0:00', '23/01/2004 0:00', '3/03/1999 0:00',
                '3/06/1998 0:00', '11/01/2005 0:00', '2/06/2004 0:00',
                '11/06/2002 0:00', '15/11/1995 0:00', '15/12/1994 0:00',
                '23/07/1999 0:00', '17/08/1995 0:00', '7/09/1995 0:00',
                '29/01/2003 23:45', '3/08/1999 0:00', '27/01/2010 0:00',
                '28/12/1998 0:00', '18/01/1996 5:00', '17/11/2006 0:00',
                '21/05/2006 0:00', '24/11/2001 0:00', '6/09/1997 0:00',
                '31/10/1997 0:00', '22/09/2002 0:00', '1/05/2009 0:00',
                '8/04/2009 0:00', '10/05/2007 19:20', '8/04/1998 0:00',
                '20/10/2003 0:00', '5/08/2008 0:00', '29/03/2008 0:00',
                '15/12/2007 0:00', '3/09/1996 4:00', '2/08/2000 0:00',
                '5/11/2007 15:14', '9/01/2007 0:00', '29/03/2000 0:00',
                '20/03/2005 0:00', '3/07/2000 0:00', '3/10/1997 0:00',
                '31/01/1996 0:00', '20/08/2014 0:00', '16/11/1994 0:00',
                '16/05/2000 0:00', '14/12/2004 0:00', '2/10/2000 0:00',
                '16/06/2014 0:00', '7/02/2006 0:00', '29/04/1994 0:00',
                '7/12/2006 0:00', '14/04/2017 0:00', '10/08/1995 0:00',
                '18/10/2005 0:00', '23/06/1999 0:00', '3/05/2006 0:00',
                '26/05/1994 0:00', '17/08/2010 0:00', '2/03/2000 0:00',
                '26/01/2002 0:00', '4/02/1996 0:00', '23/07/2004 0:00',
                '17/05/2004 0:00', '19/11/1998 0:00', '18/06/2003 0:00',
                '7/07/1999 0:00', '25/02/2005 0:00', '22/08/2003 0:00',
                '20/08/2000 0:00', '3/03/2004 0:00', '10/11/1994 0:00',
                '9/07/1995 0:00', '22/09/2000 0:00', '17/04/1996 0:00',
                '25/11/1995 0:00', '2/04/2009 0:00', '10/12/2010 0:00',
                '5/11/1996 5:00', '8/03/2003 0:00', '12/08/2005 0:00',
                '23/11/1998 0:00', '14/03/2004 0:00', '27/06/1996 0:00',
                '19/05/2016 0:00', '24/05/2006 0:00', '12/02/2000 0:00',
                '3/05/2004 0:00', '15/12/1998 0:00', '15/04/2002 0:00',
                '5/02/2009 0:00', '7/07/1997 0:00', '5/03/1994 0:00',
                '22/05/1995 0:00', '23/04/1996 0:00', '25/03/2010 0:00',
                '27/01/1995 0:00', '17/09/2008 0:00', '9/08/1994 0:00',
                '26/03/2002 0:00', '26/04/1996 0:00', '19/05/2003 0:00',
                '20/10/2000 0:00', '2/07/1999 0:00', '17/03/2009 0:00',
                '27/05/2006 0:00', '25/07/1995 0:00', '11/08/2009 0:00',
                '24/03/2005 0:00', '29/03/1994 0:00', '1/02/1994 0:00',
                '2/01/2016 0:00', '24/02/2008 18:32', '2/12/1997 0:00',
                '17/10/2008 0:00', '16/02/2010 0:00', '10/05/1996 0:00',
                '27/01/2007 0:00', '8/06/2009 3:48', '19/04/2003 0:00',
                '20/12/1999 0:00', '29/12/2007 0:00', '10/01/2000 0:00',
                '17/05/2007 0:00', '10/08/2016 0:00', '23/12/1995 0:00',
                '3/01/2009 0:00', '22/11/1995 0:00', '26/06/2007 0:00',
                '1/03/1994 0:00', '13/12/1993 5:00', '14/12/2009 0:00',
                '1/04/2008 22:47', '28/11/1994 0:00', '9/12/1998 0:00',
                '6/09/2005 0:00', '7/11/2003 0:00', '18/03/1999 0:00',
                '8/02/2005 0:00', '8/12/2016 0:00', '21/11/1996 0:00',
                '3/09/1996 0:00', '29/09/1993 0:00', '19/12/1995 0:00',
                '23/03/1999 0:00', '27/01/2011 0:00', '13/01/1996 0:00',
                '28/02/1994 5:00', '26/06/1996 0:00', '16/05/1995 4:00',
                '16/07/2007 0:00', '22/05/2009 0:00', '10/03/2006 0:00',
                '18/12/2006 0:00', '10/09/1998 0:00', '26/06/1995 0:00',
                '28/12/2007 0:00', '28/07/1999 0:00', '28/02/2002 0:00',
                '8/02/1999 0:00', '16/06/2000 17:03', '27/02/1995 0:00',
                '13/02/1999 0:00', '12/06/2001 20:58', '23/05/1995 0:00',
                '16/01/2008 0:00', '31/07/1997 0:00', '4/03/2006 0:00',
                '1/11/2000 0:00', '18/12/2008 0:00', '31/08/2003 0:00',
                '3/06/1997 0:00', '29/04/1999 0:00', '22/06/1998 0:00',
                '14/12/1999 0:00', '28/01/2004 0:00', '28/03/2007 0:00',
                '27/07/1995 0:00', '15/06/2006 0:00', '13/07/1998 0:00',
                '8/01/1997 0:00', '29/07/1998 4:00', '14/08/1997 0:00',
                '23/11/2010 0:00', '20/12/2008 0:00', '26/06/1997 0:00',
                '15/02/1999 0:00', '1/04/1998 0:00', '14/11/2008 0:00'],
               dtype=object)
```

In [ ]:

```
#Going to look at the dates now and clean up the format on those.

#Make function to clean up data column.
def date_cleaner(datetime_str):
    if datetime_str in [np.nan, "b", "0", "None"]:  # these are the missing values
        return np.nan

    if "T" in datetime_str:
        split_datetime = datetime_str.split("T")
    else:
        split_datetime = datetime_str.split()

    date = split_datetime[0]
    date_with_slash = date.replace("-", "/")

    if date_with_slash == "2002/03/20":  # only one instance of this.
        date_with_slash = "20/03/2002"
    return date_with_slash
```

In [ ]:

```
#Going to apply the cleaner format to both regdate and updated date columns
df_copy.whois_regdate = df_copy.whois_regdate.apply(date_cleaner)
df_copy["whois_regdate"] = pd.to_datetime(df_copy.whois_regdate, format="%d/%m/%Y",

#Update the updated_date column
df_copy.whois_updated_date = df_copy.whois_updated_date.apply(date_cleaner)
df_copy["whois_updated_date"] = pd.to_datetime(df_copy.whois_updated_date, format="

#Filling null values with the median. The reason being is it is impossible to know
df_copy["whois_regdate"].fillna(df_copy["whois_regdate"].median(), inplace=True)
df_copy["whois_updated_date"].fillna(df_copy["whois_updated_date"].median(), inplac
```

In [ ]:

```
#Deal with 1 last null value in dns_query_times.
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1781 entries, 0 to 1780
Data columns (total 22 columns):
 #   Column                     Non-Null Count  Dtype         
---  ------                     --------------  -----         
 0   url                        1781 non-null   object        
 1   url_length                 1781 non-null   int64         
 2   number_special_characters  1781 non-null   int64         
 3   server                     1781 non-null   object        
 4   content_length             1781 non-null   float64       
 5   whois_country              1781 non-null   object        
 6   whois_statepro             1781 non-null   object        
 7   whois_regdate              1781 non-null   datetime64[ns]
 8   whois_updated_date         1781 non-null   datetime64[ns]
 9   tcp_conversation_exchange  1781 non-null   int64         
 10  dist_remote_tcp_port       1781 non-null   int64         
 11  remote_ips                 1781 non-null   int64         
 12  app_bytes                  1781 non-null   int64         
 13  source_app_packets         1781 non-null   int64         
 14  remote_app_packets         1781 non-null   int64         
 15  source_app_bytes           1781 non-null   int64         
 16  remote_app_bytes           1781 non-null   int64         
 17  app_packets                1781 non-null   int64         
 18  dns_query_times            1780 non-null   float64       
 19  type                       1781 non-null   int64         
 20  charset_imputed            1781 non-null   object        
 21  standardized_server        1781 non-null   object        
dtypes: datetime64[ns](2), float64(2), int64(12), object(6)
memory usage: 306.2+ KB
```

In [ ]:

```
#I will interpolate the dns query column. It's one value, so filling in the null va
df_copy['dns_query_times'] = df_copy['dns_query_times'].interpolate()
```

In [ ]:

```
#Sanity check - clear.
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1781 entries, 0 to 1780
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype         
---  ------                      --------------  -----         
 0   url                         1781 non-null   object        
 1   url_length                  1781 non-null   int64         
 2   number_special_characters   1781 non-null   int64         
 3   content_length              1781 non-null   float64       
 4   whois_country               1781 non-null   object        
 5   whois_statepro              1781 non-null   object        
 6   whois_regdate               1781 non-null   datetime64[ns]
 7   whois_updated_date          1781 non-null   datetime64[ns]
 8   tcp_conversation_exchange   1781 non-null   int64         
 9   dist_remote_tcp_port        1781 non-null   int64         
 10  remote_ips                  1781 non-null   int64         
 11  app_bytes                   1781 non-null   int64         
 12  source_app_packets          1781 non-null   int64         
 13  remote_app_packets          1781 non-null   int64         
 14  source_app_bytes            1781 non-null   int64         
 15  remote_app_bytes            1781 non-null   int64         
 16  app_packets                 1781 non-null   int64         
 17  dns_query_times             1781 non-null   int32         
 18  type                        1781 non-null   int64         
 19  charset_imputed             1781 non-null   object        
 20  standardized_server         1781 non-null   object        
dtypes: datetime64[ns](2), float64(1), int32(1), int64(12), object(5)
memory usage: 285.4+ KB
```

In [ ]:

```
#Change column to int type for cleaner clarity.
df_copy['dns_query_times'] = df_copy['dns_query_times'].astype(int)
```

In [ ]:

```
#Sanity check
df_copy['dns_query_times'].value_counts()
```

Out[ ]:

```
dns_query_times
0     976
4     309
6     213
2     143
8     105
10     19
12     12
14      2
20      1
9       1
Name: count, dtype: int64
```

In [ ]:

```
#Drop the server column since we have it standardized and ready to go.
df_copy = df_copy.drop(columns=["server"])
```

In [ ]:

```
#Data is now clean and ready to go!
df_copy.head()
```

Out[ ]:

	url	url_length	number_special_characters	content_length	whois_country	whois_sta
0	M0_109	16	7	263.0	Other	
1	B0_2314	16	6	15087.0	Other	
2	B0_911	16	6	324.0	Other	
3	B0_113	17	6	162.0	US	
4	B0_403	17	6	124140.0	US	

5 rows × 21 columns

Section 2: Descriptive Questions

1. How many unique URL's are in the dataset?

Answer: There are 1781 unique websites examined in the dataset.

In [ ]:

```
df_copy['url'].nunique()
```

Out[ ]:

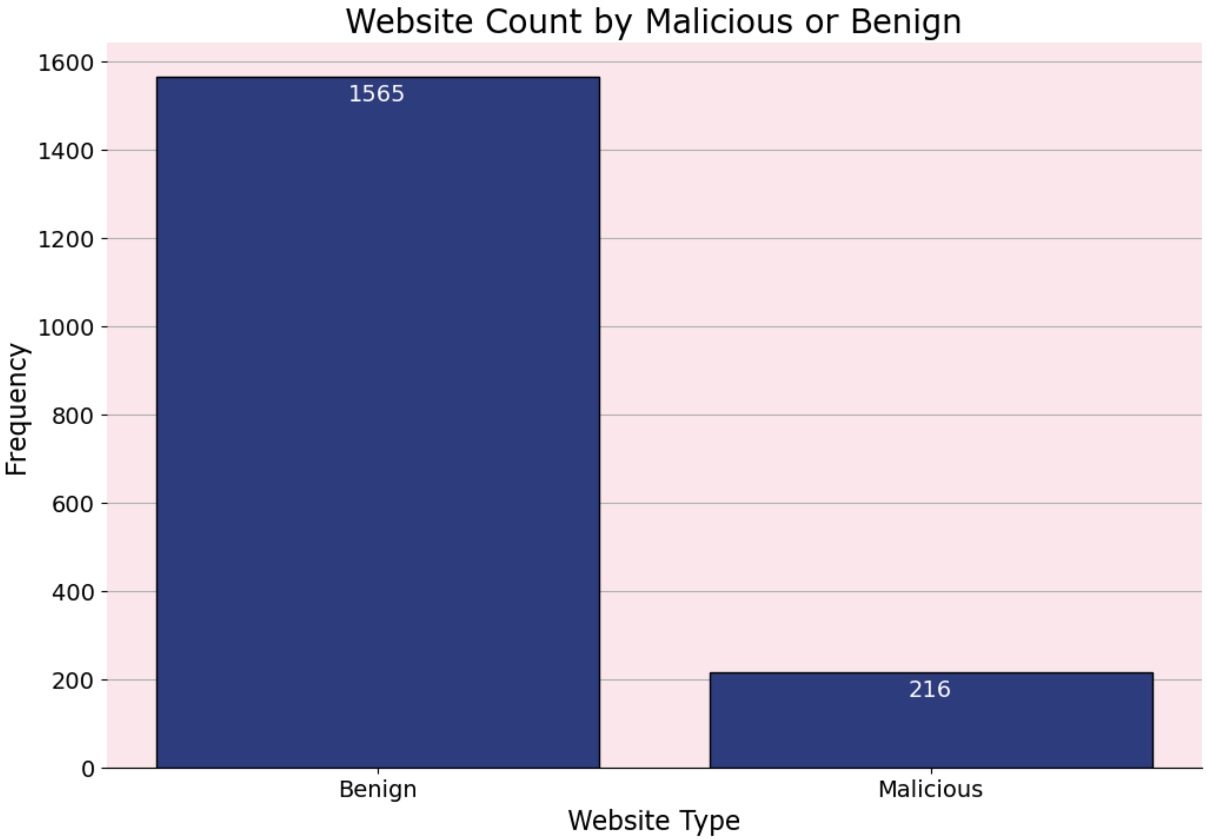
```
1781
```

2. How many websites are malicious and how many are benign?

In [ ]:

```
def plot_hist(data, plot_title, x_name, y_name, bin_amount=10, bar_color='#CC313D',

    #Custom Fonts
    font1 = {'family':'verdana','color':'#000000','size':20}
    font2 = {'family':'verdana','color':'#000000','size':16}

    #Create the plot, set x & y axis titles, and graph title.
    fig, ax = plt.subplots(figsize=(12,8))
    n, bins, patches = ax.hist(x=data, bins=bin_amount, color=bar_color, edgecolor=
    ax.set_title(plot_title, fontdict=font1)
    ax.set_xlabel(x_name, fontdict=font2)
    ax.set_ylabel(y_name,fontdict=font2)

    #Plot Styling for axes ticks
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    ax.set_facecolor(face_color)

    ax.spines["right"].set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["top"].set_visible(False)


    for i in range(len(patches)):
        ax.text((bins[i] + bins[i+1]) / 2,  # Midpoint of the bin
                n[i] + annotate_placement,  # Height of the bin
                f'{n[i]:.0f}',  # Annotation text
                ha='center',
                va='top',
                color=annotate_color,
                fontsize=annotate_font)

    ax.grid(axis='y')
    plt.xticks(ha='center')

    plt.show()


def plot_bar(x_data, y_data, plot_title, x_name, y_name, bar_color='#CC313D', face_

    #Custom Fonts
    font1 = {'family':'verdana','color':'#000000','size':20}
    font2 = {'family':'verdana','color':'#000000','size':16}

    #Create the plot, set x & y axis titles, and graph title.
    fig, ax = plt.subplots(figsize=(12,8))
    ax.bar(x=x_data, height=y_data, color=bar_color, edgecolor='black', zorder=3)
    ax.set_title(plot_title, fontdict=font1)
    ax.set_xlabel(x_name, fontdict=font2)
    ax.set_ylabel(y_name,fontdict=font2)

    ax.spines["right"].set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["top"].set_visible(False)

    #Plot Styling for axes ticks
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)
    ax.set_facecolor(face_color)



    for bar in ax.patches:
        height = bar.get_height()
        ax.text(bar.get_x() + bar.get_width() / 2,
            height + annotate_placement,
            f'{height:.0f}',
            ha='center',
            va='top',
            color=annotate_color,
            fontsize=annotate_font)

    ax.grid(axis='y')
    plt.xticks(ha='center')

    plt.show()
```

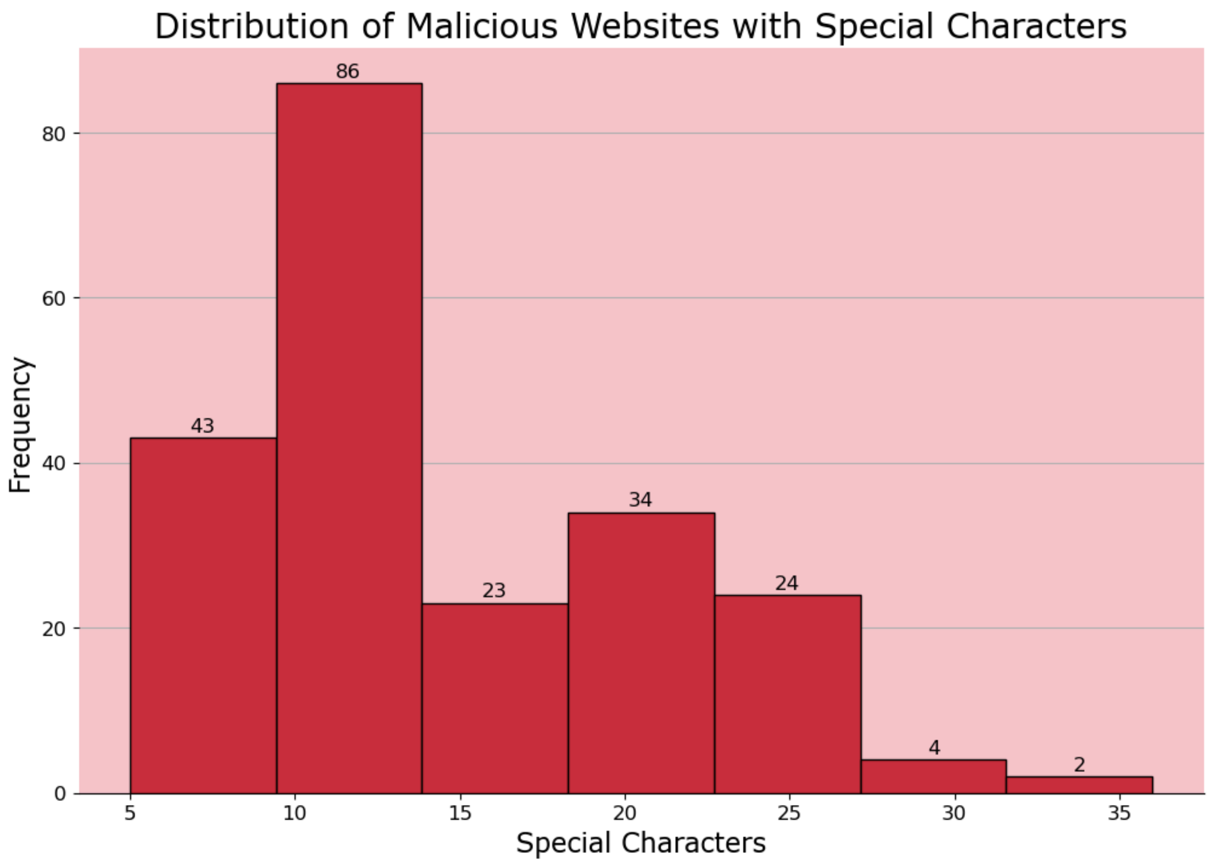
```
In [ ]: counts = df_copy['type'].value_counts()
        counts = counts.rename(index={0: 'Benign', 1: 'Malicious'})

        plot_bar(counts.index, counts.values, "Website Count by Malicious or Benign", "Webs
```



3. Do malicious websites have many special characters in their URL?

```
In [ ]: #Create Data frame masks.
        MALICIOUS_WEBSITES = df_copy['type'] == 1
        BENIGN_WEBSITES = df_copy['type'] == 0
```

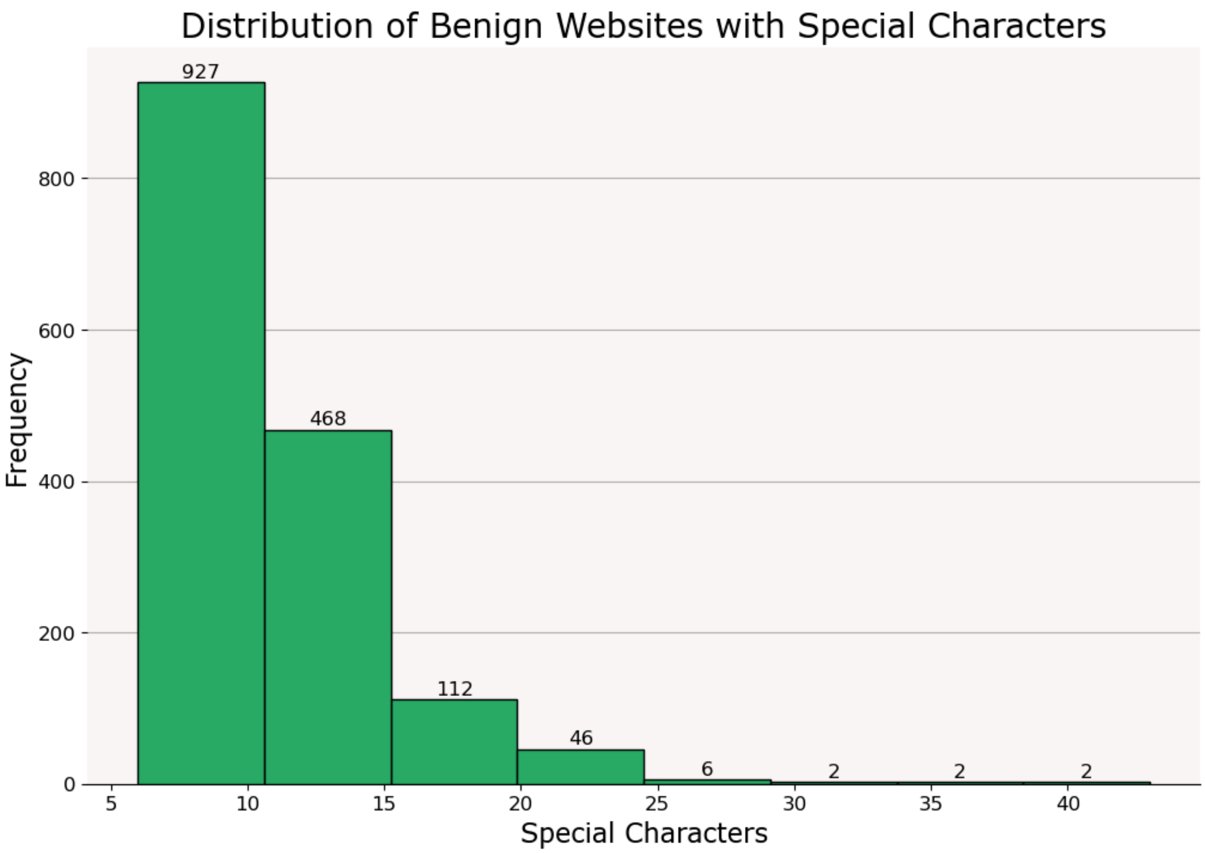
```
In [ ]: #Data sorting to set up for plotting.
        special_characters_malicious = df_copy[MALICIOUS_WEBSITES]['number_special_characte

        #Plot data.
        plot_hist(special_characters_malicious,
                "Distribution of Malicious Websites with Special Characters",
                "Special Characters",
                "Frequency",
                bin_amount=7,
                annotate_placement=2.5,
                annotate_color="black",
                annotate_font=12)
```



4. Do benign websites have many special characters in their url?

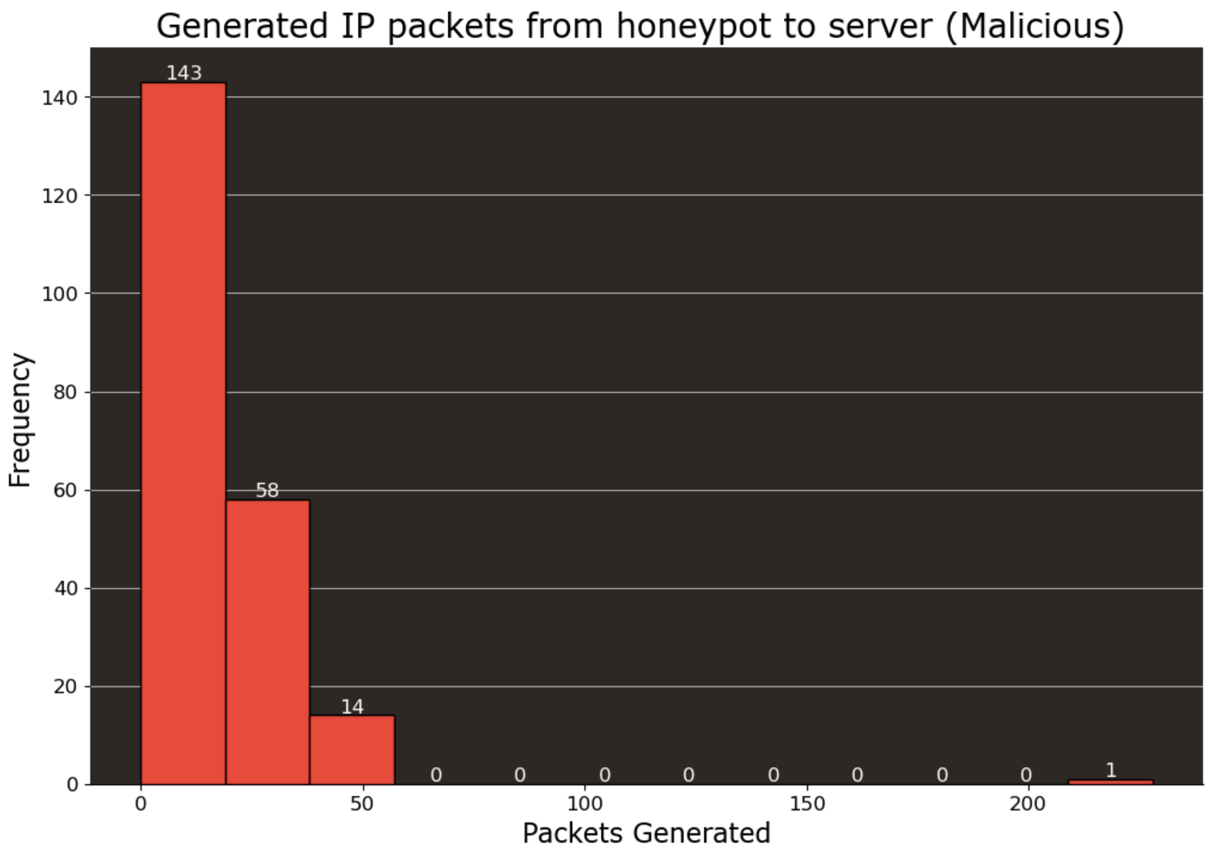
In [ ]:

```
#Data sorting to set up for plotting.
special_characters_benign = df_copy[BENIGN_WEBSITES]['number_special_characters'].v

#Plot data.
plot_hist(special_characters_benign,
        "Distribution of Benign Websites with Special Characters",
        "Special Characters", "Frequency",
         bin_amount=8,
         bar_color="#2BAE66FF",
         face_color="#FCF6F5FF",
         annotate_placement=25,
         annotate_color="black",
         annotate_font=12)
```



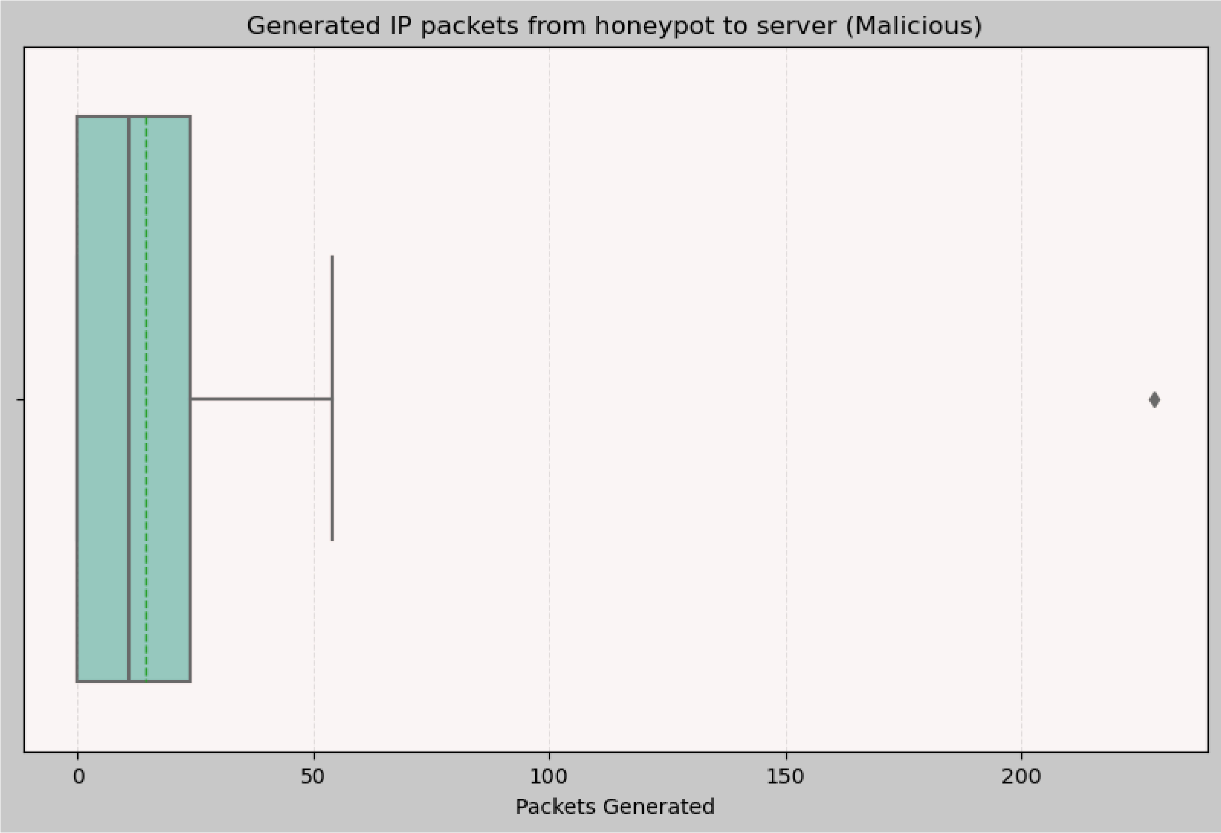
5. Do malicious websites generate many IP packets when communicating between a honeypot and the server?

In [ ]:
```
#Data sorting for plotting
malicious_df = df_copy[MALICIOUS_WEBSITES]
malicious_app_packets = df_copy[MALICIOUS_WEBSITES]['app_packets'].values

#Plotting
plot_hist(malicious_app_packets,
        "Generated IP packets from honeypot to server (Malicious) ",
        "Packets Generated", "Frequency",
        bin_amount=12,
        bar_color="#E94B3CFF",
        face_color="#2D2926FF",
        annotate_placement=3.5,
        annotate_color="white",
        annotate_font=12)
```



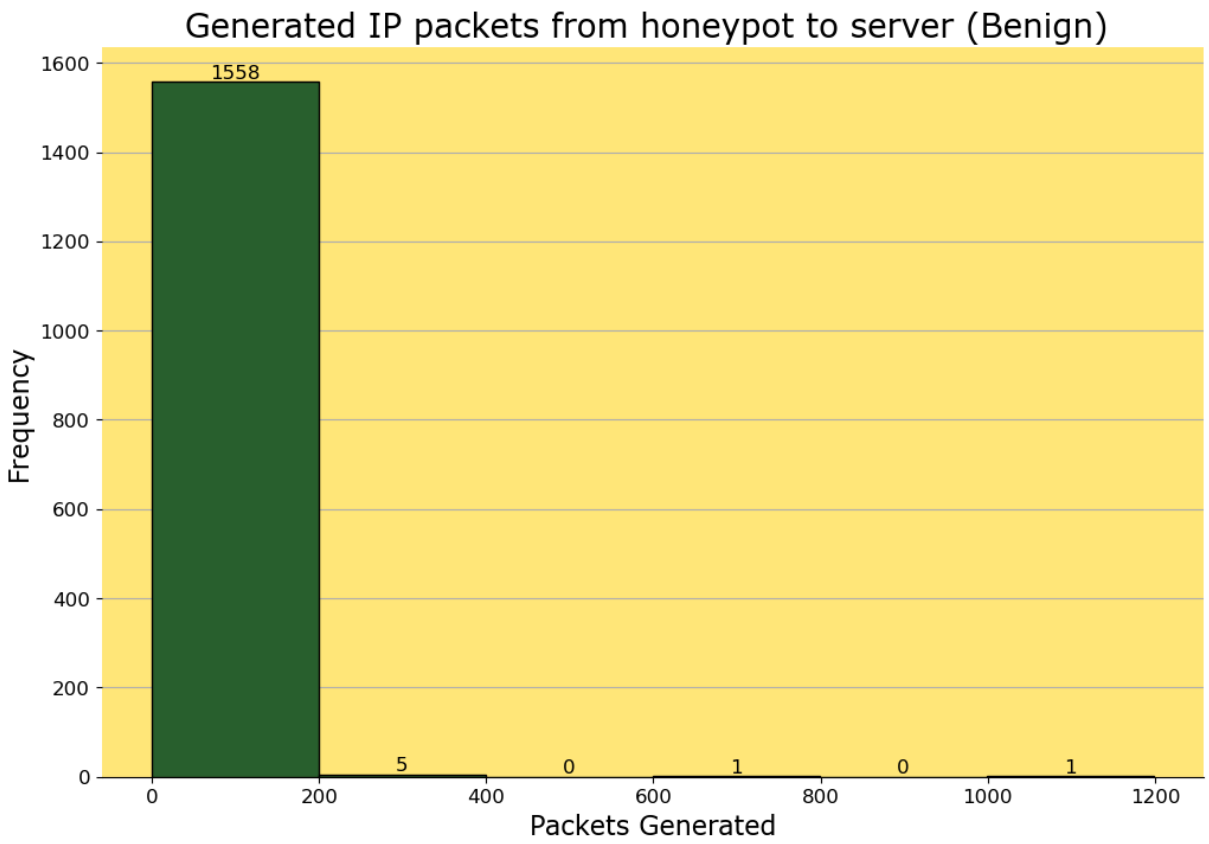
In [ ]:

```
#Boxplot of the distribution

plt.figure(figsize=(10, 6), facecolor="#cccccc")
sns.boxplot(x=malicious_app_packets,
            palette="Set3",
            zorder=2,
            showmeans=True,
            meanline=True)
plt.title('Generated IP packets from honeypot to server (Malicious)')
plt.xlabel('Packets Generated')

#Set background of grid to custom color and add y-axis gridlines.
ax = plt.gca()
ax.set_facecolor('#FCF6F5')
plt.grid(True, axis='x', linestyle='--', linewidth=0.7, color='gray', alpha=0.2, zo

plt.show()
```



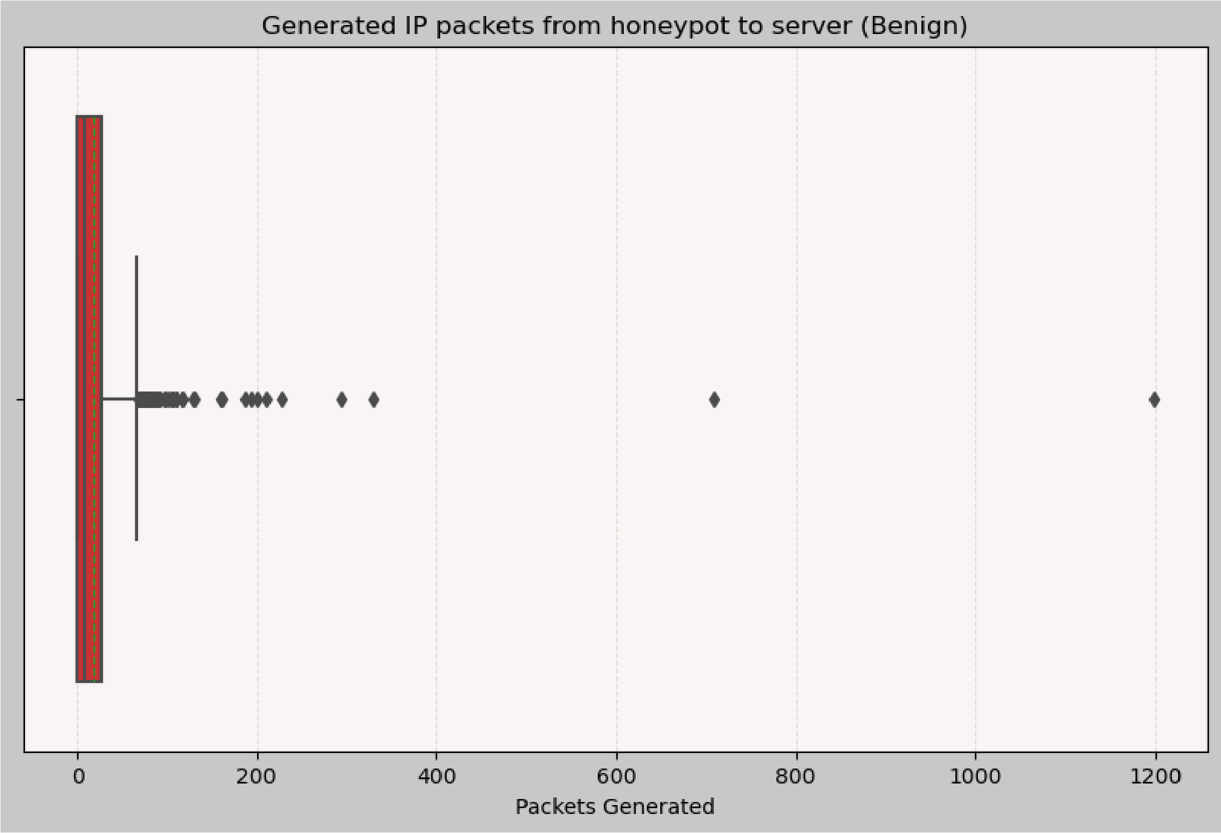
6. Do benign websites generate many IP packets when communicating between a honeypot and the server?

```
In [ ]: #Data sorting for plotting
benign_app_packets = df_copy[BENIGN_WEBSITES]['app_packets'].values

#Plot data.
plot_hist(benign_app_packets,
        "Generated IP packets from honeypot to server (Benign) ",
        "Packets Generated",
        "Frequency",
        bin_amount=6,
        bar_color="#2C5F2DFF",
        face_color="#FFE77AFF",
        annotate_placement=40,
        annotate_color="black",
        annotate_font=12)
```



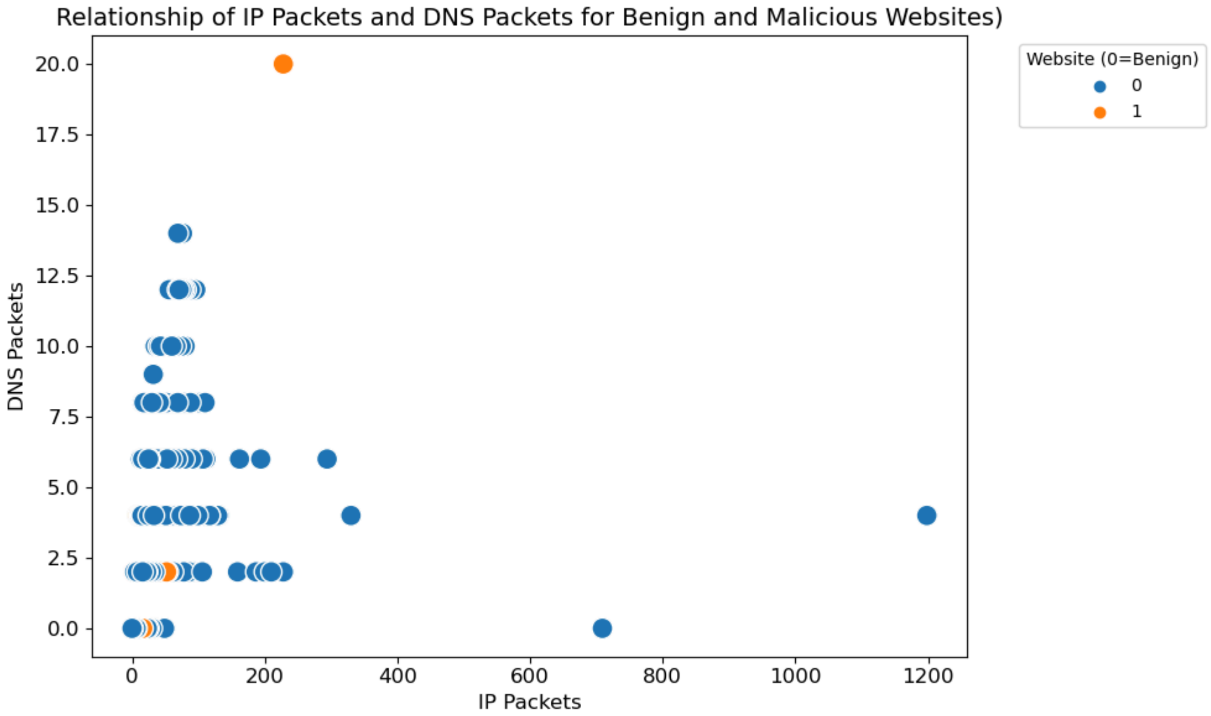
In [ ]:

```
#Boxplot of the distribution
plt.figure(figsize=(10, 6), facecolor="#cccccc")
sns.boxplot(x=benign_app_packets,
            palette="Set1",
            zorder=2,
            showmeans=True,
            meanline=True)
plt.title('Generated IP packets from honeypot to server (Benign)')
plt.xlabel('Packets Generated')

#Set background of grid to custom color and add y-axis gridlines.
ax = plt.gca()
ax.set_facecolor('#FCF6F5')
plt.grid(True, axis='x', linestyle='--', linewidth=0.7, color='gray', alpha=0.2, zo

plt.show()
```



In [ ]:

```
# Relationship analysis for DNS Query Times and APP Packetsd
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_copy, x=df_copy['app_packets'], y=df_copy['dns_query_times'
plt.title("Relationship of IP Packets and DNS Packets for Benign and Malicious Webs
plt.xlabel('IP Packets', fontsize=12)
plt.ylabel('DNS Packets', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title='Website (0=Benign)', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

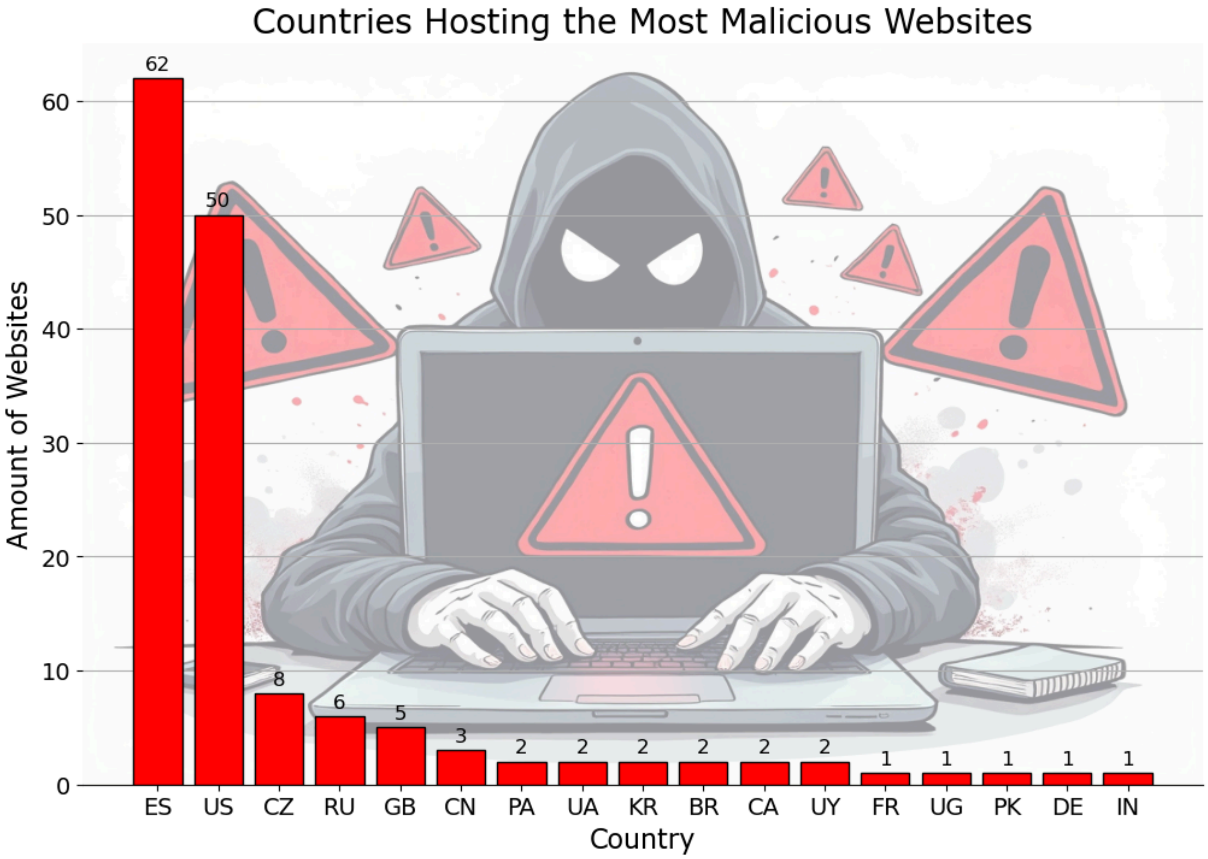


7. Which countries host the most malicious websites?

In [ ]:

```
#Sort Data
countries_malicious_df = df_copy[MALICIOUS_WEBSITES]
MASK_COUNTRY = countries_malicious_df['whois_country'] != "Other"
country_malicious = countries_malicious_df[MASK_COUNTRY]['whois_country'].value_cou

#Custom Fonts
font1 = {'family':'verdana','color':'#000000','size':20}
font2 = {'family':'verdana','color':'#000000','size':16}

#Create the plot, set x & y axis titles, and graph title.
fig, ax = plt.subplots(figsize=(12,8))
ax.bar(x=country_malicious.index, height=country_malicious.values, color='red', edg
ax.set_title('Countries Hosting the Most Malicious Websites', fontdict=font1)
ax.set_xlabel('Country', fontdict=font2)
ax.set_ylabel('Amount of Websites',fontdict=font2)

ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Plot Styling for axes ticks
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
ax.set_facecolor("white")


# Get the current axis limits.
x_min, x_max = ax.get_xlim()
y_min, y_max = ax.get_ylim()
```

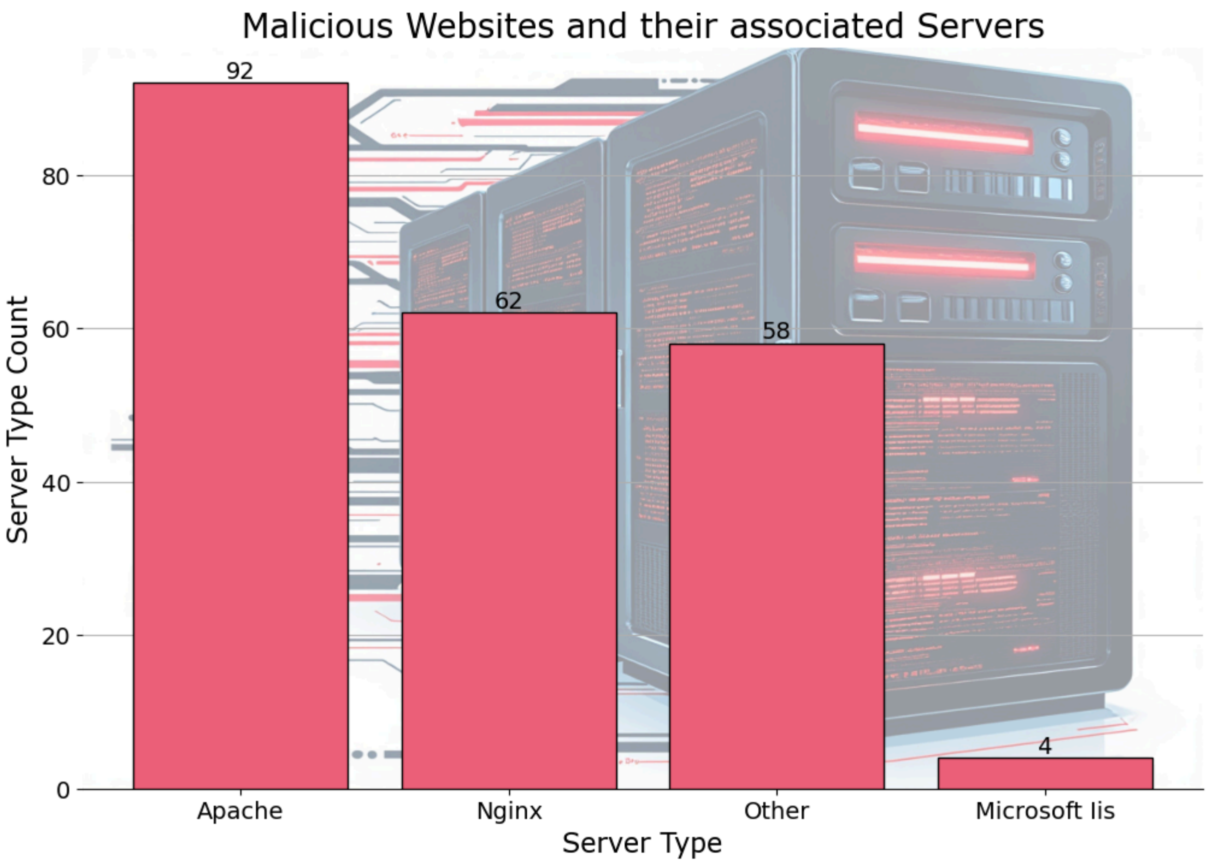
```
#Customize the graph, set image background
background = plt.imread(r'C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_
ax.imshow(background, extent=[x_min, x_max, y_min, y_max], aspect='auto', alpha=0.4

for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + 2,
        f'{height:.0f}',
        ha='center',
        va='top',
        color="black",
        fontsize=12)

ax.grid(axis='y', zorder=3)
plt.xticks(ha='center')

plt.show()
```



8. Of the malicious websites, what is the most common server type?

```
In [ ]: #Sort Data
country_malicious = countries_malicious_df['standardized_server'].value_counts()
country_malicious.index = country_malicious.index.str.title()

#Custom Fonts
font1 = {'family':'verdana','color':'#000000','size':20}
```

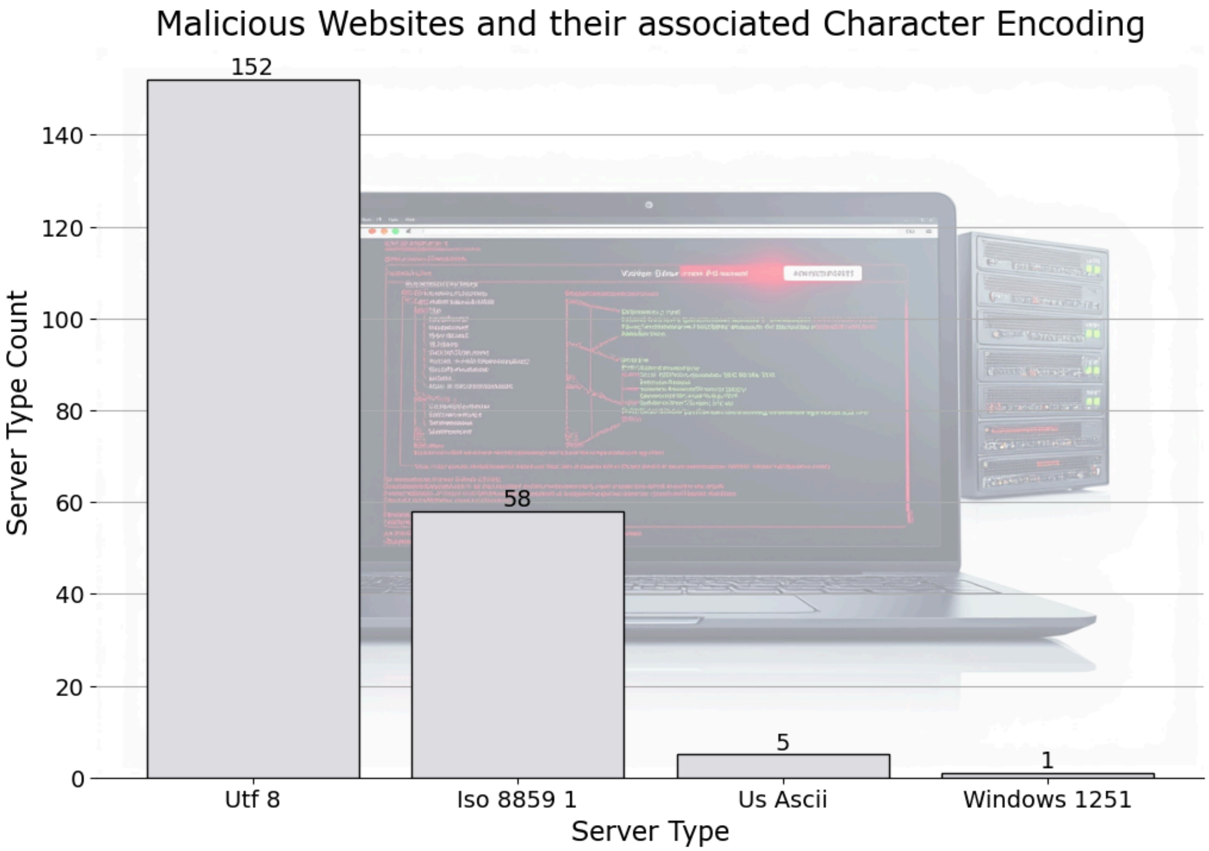
```
font2 = {'family':'verdana','color':'#000000','size':16}

#Create the plot, set x & y axis titles, and graph title.
fig, ax = plt.subplots(figsize=(12,8))
ax.bar(x=country_malicious.index, height=country_malicious.values, color='#EF6079FF
ax.set_title('Malicious Websites and their associated Servers', fontdict=font1)
ax.set_xlabel('Server Type', fontdict=font2)
ax.set_ylabel('Server Type Count',fontdict=font2)

ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Plot Styling for axes ticks
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
ax.set_facecolor("white")


# Get the current axis limits.
x_min, x_max = ax.get_xlim()
y_min, y_max = ax.get_ylim()

#Customize the graph, set image background
background = plt.imread(r'C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_
ax.imshow(background, extent=[x_min, x_max, y_min, y_max], aspect='auto', alpha=0.4

for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + 3,
        f'{height:.0f}',
        ha='center',
        va='top',
        color="black",
        fontsize=14)

ax.grid(axis='y', zorder=3)
plt.xticks(ha='center')

plt.show()
```



9. Of the malicious websites, what is the most common server character encoding?

```
In [ ]: #Sort Data
country_malicious = countries_malicious_df['charset_imputed'].value_counts()
country_malicious.index = country_malicious.index.str.title()

#Custom Fonts
font1 = {'family':'verdana','color':'#000000','size':20}
font2 = {'family':'verdana','color':'#000000','size':16}

#Create the plot, set x & y axis titles, and graph title.
fig, ax = plt.subplots(figsize=(12,8))
ax.bar(x=country_malicious.index, height=country_malicious.values, color='#DFDCE5FF
ax.set_title('Malicious Websites and their associated Character Encoding', fontdict
ax.set_xlabel('Server Type', fontdict=font2)
ax.set_ylabel('Server Type Count',fontdict=font2)

ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Plot Styling for axes ticks
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
ax.set_facecolor("white")
```

```
# Get the current axis limits.
x_min, x_max = ax.get_xlim()
y_min, y_max = ax.get_ylim()

#Customize the graph, set image background
background = plt.imread(r'C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_
ax.imshow(background, extent=[x_min, x_max, y_min, y_max], aspect='auto', alpha=0.4

for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + 5,
        f'{height:.0f}',
        ha='center',
        va='top',
        color="black",
        fontsize=14)

ax.grid(axis='y', zorder=3)
plt.xticks(ha='center')

plt.show()
```



Section 3: Inferential Analysis (All Tests 95% Significance Level)

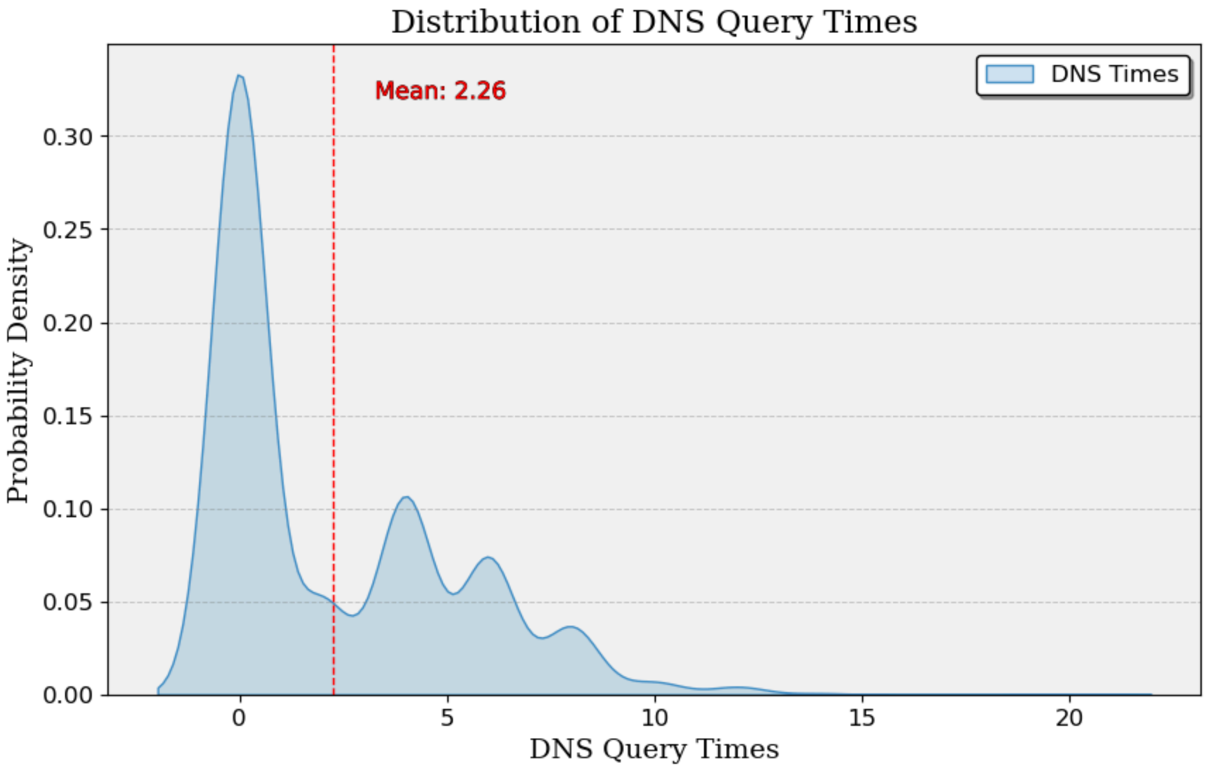
In [ ]:

```
#Function for testing our hypothesis.
def test_outcome(pvalue, alpha=0.05):
    if pvalue < alpha:
        return "Reject the null hypothesis."
    else:
        return "Fail to reject the null hypothesis."

#KDE Function for beautified graphs.
def plot_kde(column, title, x_name, label, x_val, y_val, color='#408EC6', legend_lo
    #Font dictionaries for custom styling

    font1 = {'family':'serif','color':'black','size':16}
    font2 = {'family':'serif','color':'black','size':14}

    #Set up the plot
    fig, ax = plt.subplots(figsize=(10, 6))

    #KDE plot
    sns.kdeplot(data=df_copy, x=column, color=color, label=label, fill=True)

    #Labeling axes, customizing font sizes and styles, adjust tick sizes, and setti
    plt.title(title, fontdict=font1)
    plt.xlabel(x_name, fontdict=font2)
    plt.ylabel('Probability Density', fontdict=font2)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.grid(axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.4)
    ax.set_facecolor('#f0f0f0')

    # Add a vertical line at the mean
    mean_points = df_copy[column].mean()
    plt.axvline(mean_points, color='red', linestyle='--', linewidth=1)
    plt.text(mean_points - x_val, y_val, f'Mean: {mean_points:.2f}', color='red', f
                             path_effects.Normal()])

    # Add legend
    plt.legend(loc=legend_loc, fontsize=12, frameon=True, fancybox=True, shadow=Tru

    plt.show()

#Function for hypothesis testing.
def multi_sample_test(dataset1, dataset2, x1, x2, label1, label2, title, xname, yna
                      dataset3=None, x3=None, label3=None,
                      dataset4=None, x4=None, label4=None,
                      kde_color1='#fbb30b', kde_color2='#9ce15b', kde_color3='#5d8a

    font3 = {'family':'fantasy','color':'black','size':20}
    font4 = {'family':'fantasy','color':'black','size':14}

    # Set up the plot
    plt.figure(figsize=(12 if dataset4 is None else 14, 7))

    # KDE plots
    sns.kdeplot(data=dataset1, x=x1, color=kde_color1, label=label1, fill=True, alp
    sns.kdeplot(data=dataset2, x=x2, color=kde_color2, label=label2, fill=True, alp
    if dataset3 is not None:
        sns.kdeplot(data=dataset3, x=x3, color=kde_color3, label=label3, fill=True,

    if dataset4 is not None:
        sns.kdeplot(data=dataset4, x=x4, color=kde_color4, label=label4, fill=True,

    # Labeling axes, customizing font sizes and styles, adjust tick sizes, and sett
    plt.title(title, fontdict=font3)
    plt.xlabel(xname, fontdict=font4)
    plt.ylabel(yname, fontdict=font4)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    ax = plt.gca()
    ax.set_facecolor('#f0f0f0')

    # Add legend
    plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='gr
    plt.legend(loc='upper right', fontsize=12, frameon=True, fancybox=True, shadow=
               facecolor='white', edgecolor='black', ncol=1 if dataset4 is None els

    plt.show()
```

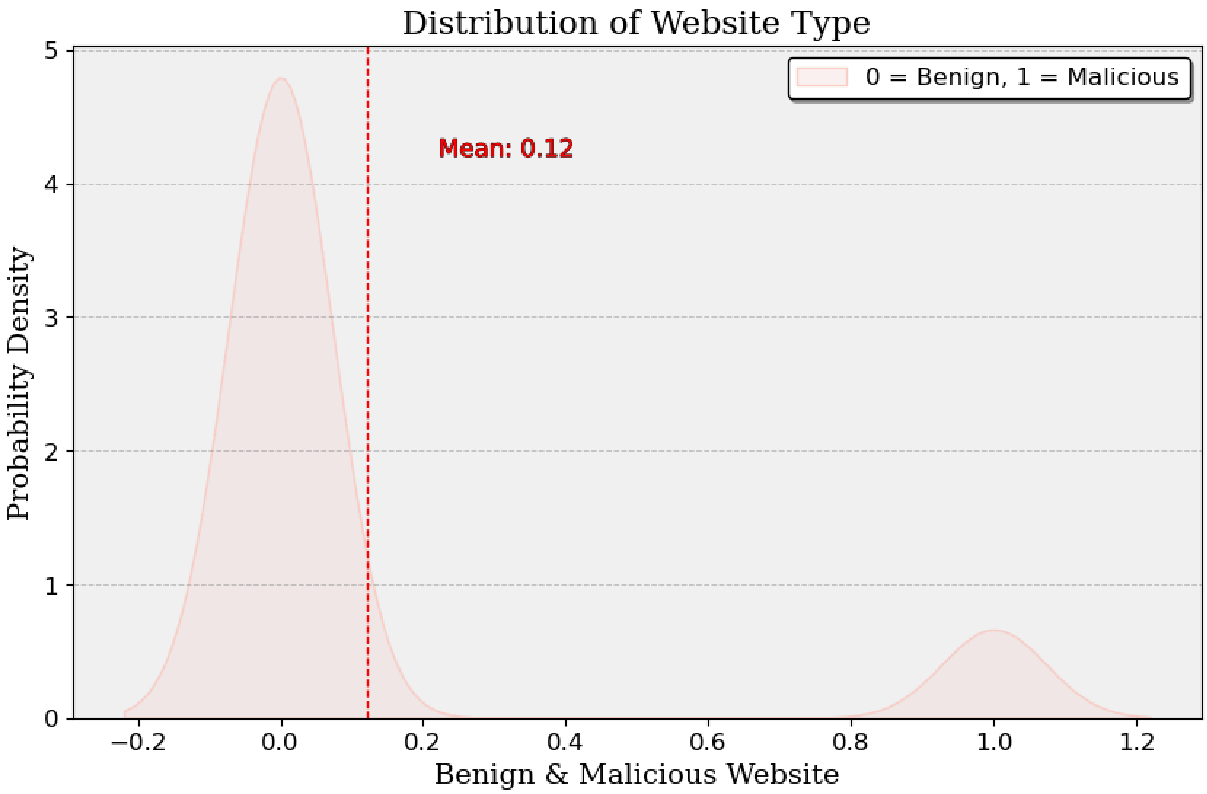
First I will inspect the distribution of my data via a KDE plot for numerous columns.

In [ ]:

```
plot_kde(column="dns_query_times",
        title="Distribution of DNS Query Times",
        x_name="DNS Query Times",
        label="DNS Times",
        x_val=-1,
        y_val=.32,
        color='#408EC6',
        legend_loc="upper right")
```

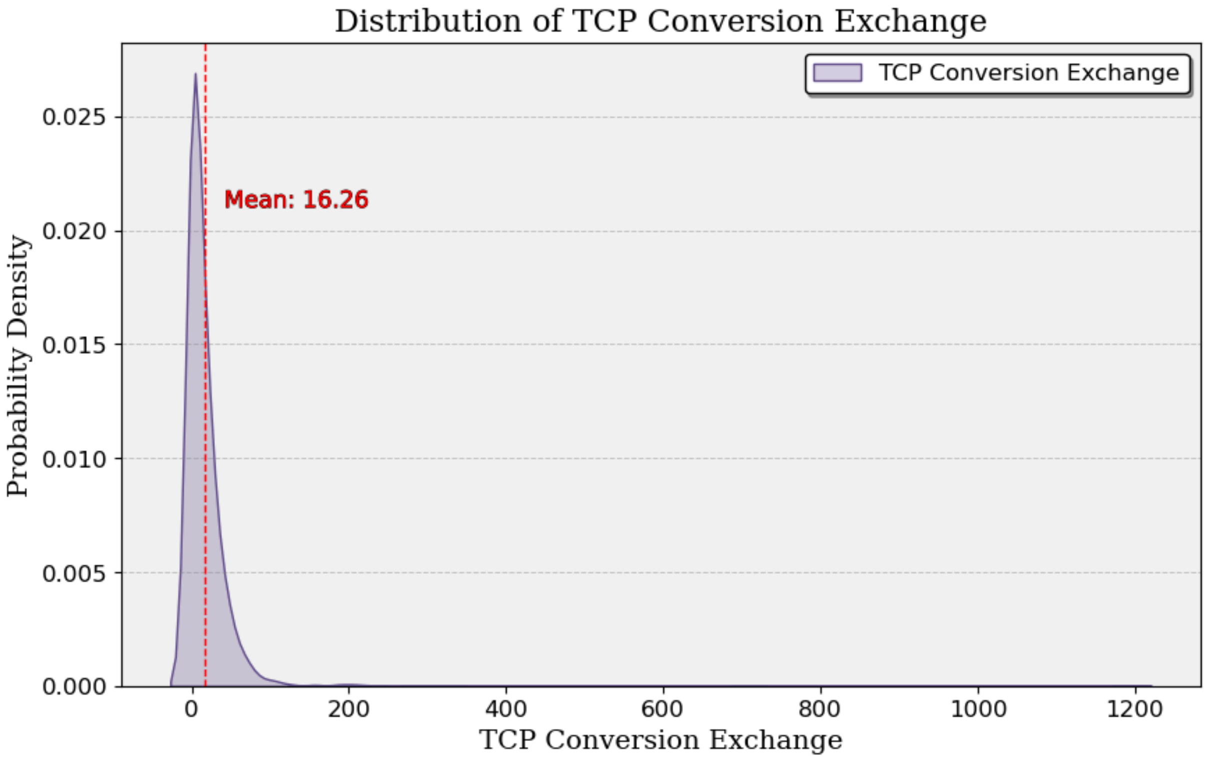


```
In [ ]: plot_kde(column="type",
                 title="Distribution of Website Type",
                 x_name="Benign & Malicious Website",
                 label="0 = Benign, 1 = Malicious",
                 x_val=-0.1,
                 y_val=4.2,
                 color='#FAD0C9FF',
                 legend_loc="upper right")
```

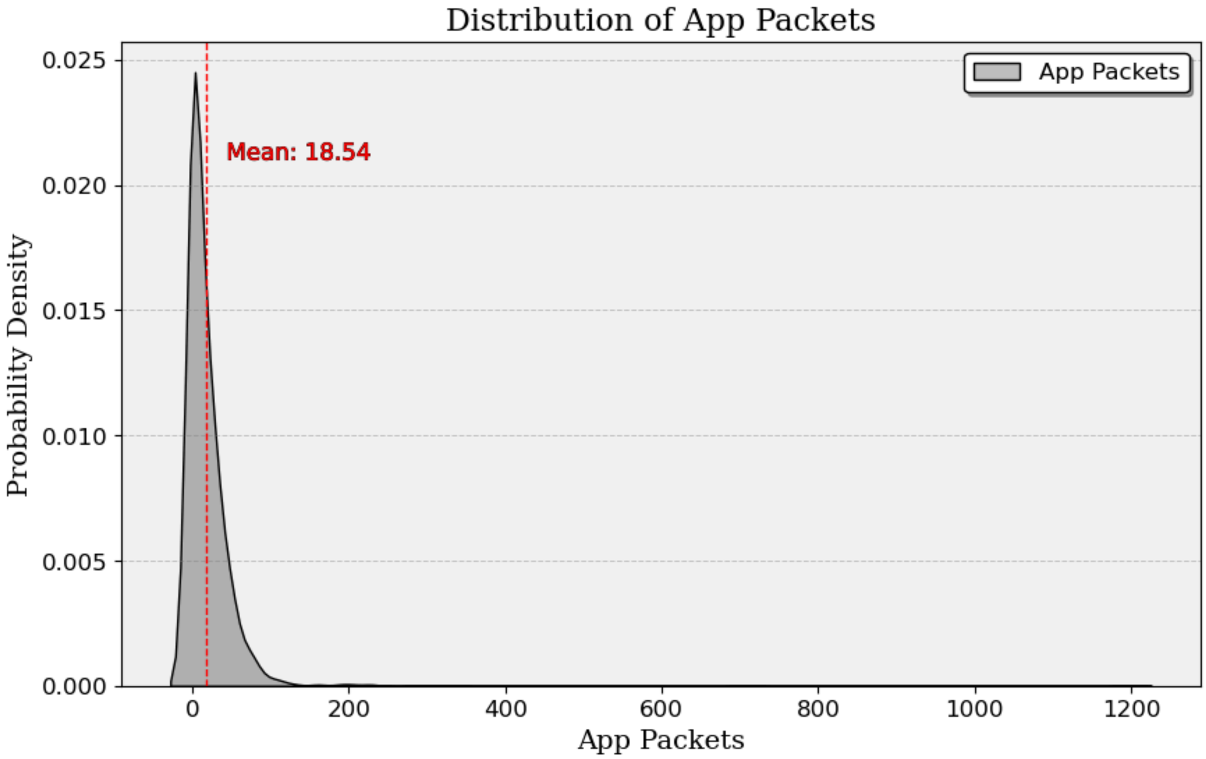


In [ ]:

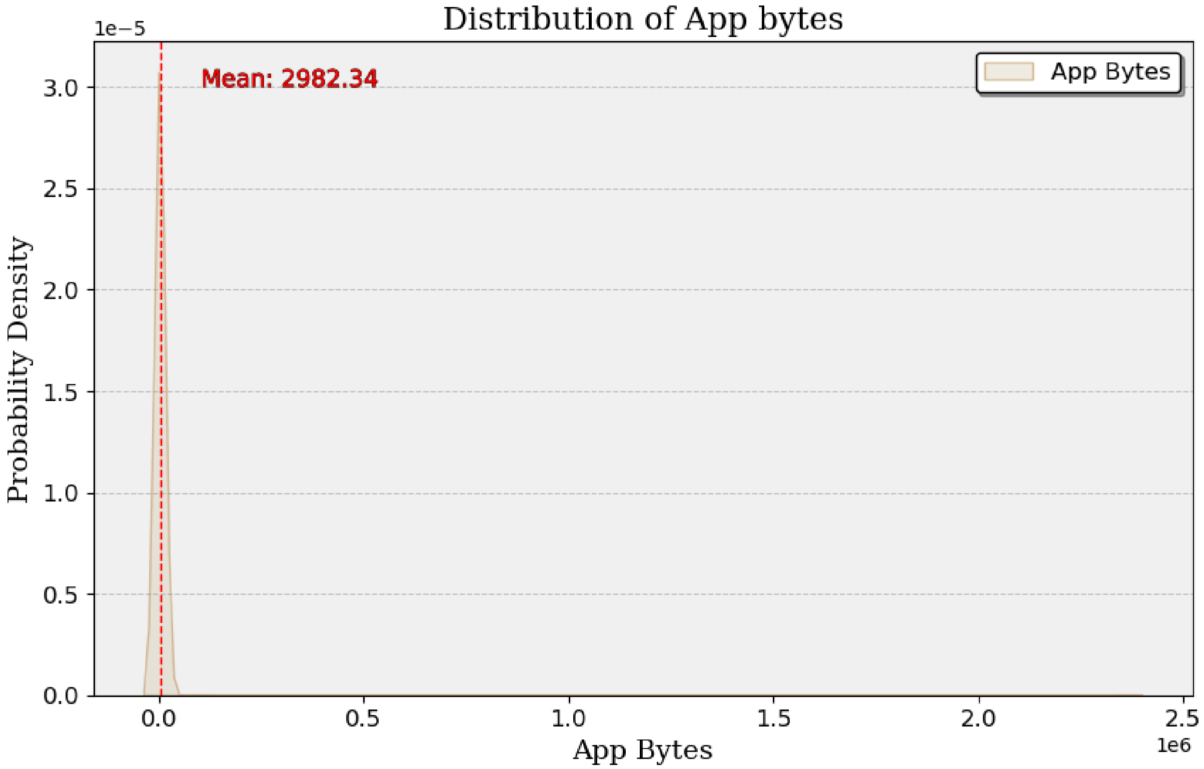
```
plot_kde(column="tcp_conversation_exchange",
        title="Distribution of TCP Conversion Exchange",
        x_name="TCP Conversion Exchange",
        label="TCP Conversion Exchange",
        x_val=-25,
        y_val=.021,
        color='#5F4B8BFF',
        legend_loc="upper right")
```



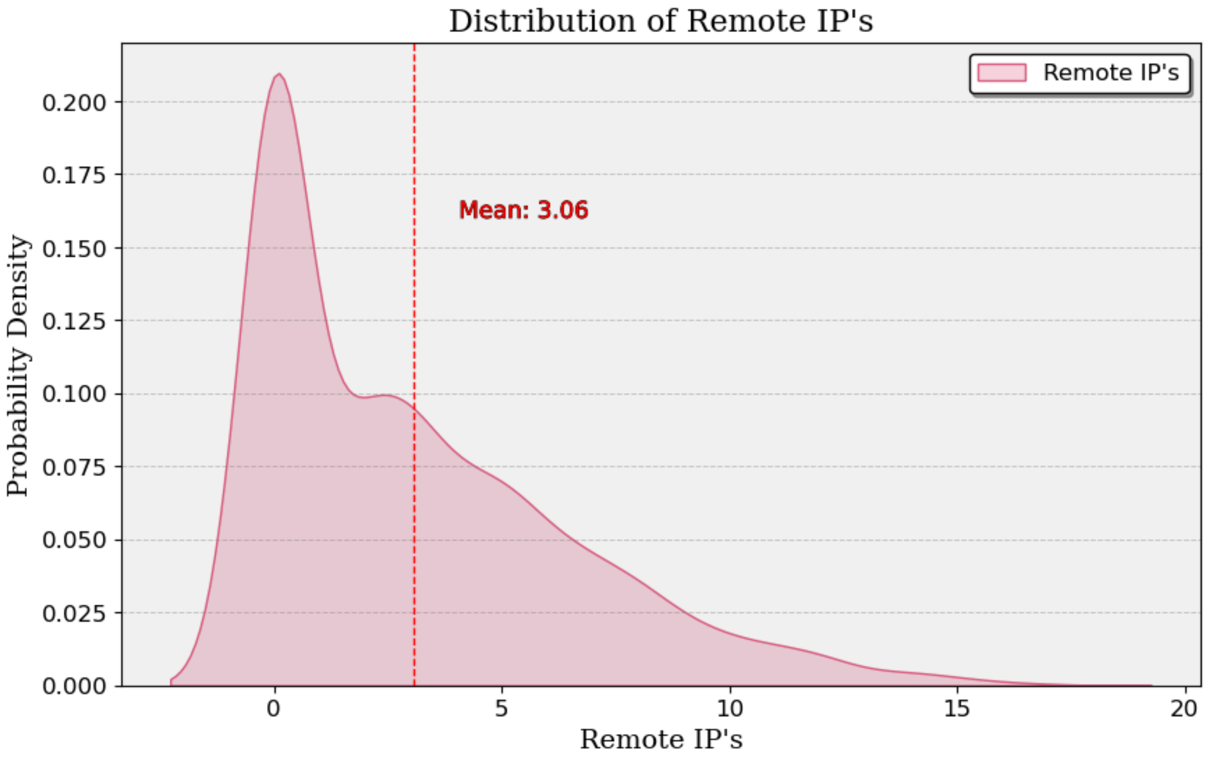
```
In [ ]: plot_kde(column="app_packets",
                 title="Distribution of App Packets",
                 x_name="App Packets",
                 label="App Packets",
                 x_val=-25,
                 y_val=0.021,
                 color='#000000FF',
                 legend_loc="upper right")
```



In [ ]:
```
plot_kde(column="app_bytes",
        title="Distribution of App bytes",
        x_name="App Bytes",
        label="App Bytes",
        x_val=-100000,
        y_val=0.00003,
        color='#D4B996FF',
        legend_loc="upper right")
```



In [ ]:
```
plot_kde(column="remote_ips",
        title="Distribution of Remote IP's",
        x_name="Remote IP's",
        label="Remote IP's",
        x_val=-1,
        y_val=0.16,
        color='#D85A7FFF',
        legend_loc="upper right")
```



Based off of all the analysis I have conducted so far, visually speaking, my data is not normally and in fact, it is positively skewed. However, just for good measure, let's perform a Shapiro-Wilkes test to test for normality on specific columns.

In [ ]:

```
#Performing normality test on app bytes column.

stat, p_val = stats.shapiro(df_copy['app_bytes'])
print(f"The P-Value calculated from the test is: {p_val}.")

# Interpret the results
alpha = 0.05
if p_val > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
The P-Value calculated from the test is: 0.0.
Sample does not look Gaussian (reject H0)
```

In [ ]:

```
#Performing normality test on remote_ips column.

stat, p_val = stats.shapiro(df_copy['remote_ips'])
print(f"The P-Value calculated from the test is: {p_val}.")

# Interpret the results
alpha = 0.05
if p_val > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
The P-Value calculated from the test is: 1.739388203384155e-38.
Sample does not look Gaussian (reject H0)
```

In [ ]:

```
#Performing normality test on type column.

stat, p_val = stats.shapiro(df_copy['type'])
print(f"The P-Value calculated from the test is: {p_val}.")

# Interpret the results
alpha = 0.05
if p_val > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
The P-Value calculated from the test is: 0.0.
Sample does not look Gaussian (reject H0)
```

In [ ]:

```
#Performing normality test on dns query times column.

stat, p_val = stats.shapiro(df_copy['dns_query_times'])
print(f"The P-Value calculated from the test is: {p_val}.")

# Interpret the results
alpha = 0.05
if p_val > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
The P-Value calculated from the test is: 8.407790785948902e-45.
Sample does not look Gaussian (reject H0)
```

After visually inspecting the data and performing Shapiro-Wilk tests on specific columns, it is safe to say we are dealing with non-normal data. As such, the following statistical tests that I will be performing will be for non-normal data. (IE: Mann-Whitney U Test, Kruskal-Wallis H Test, Etc.)

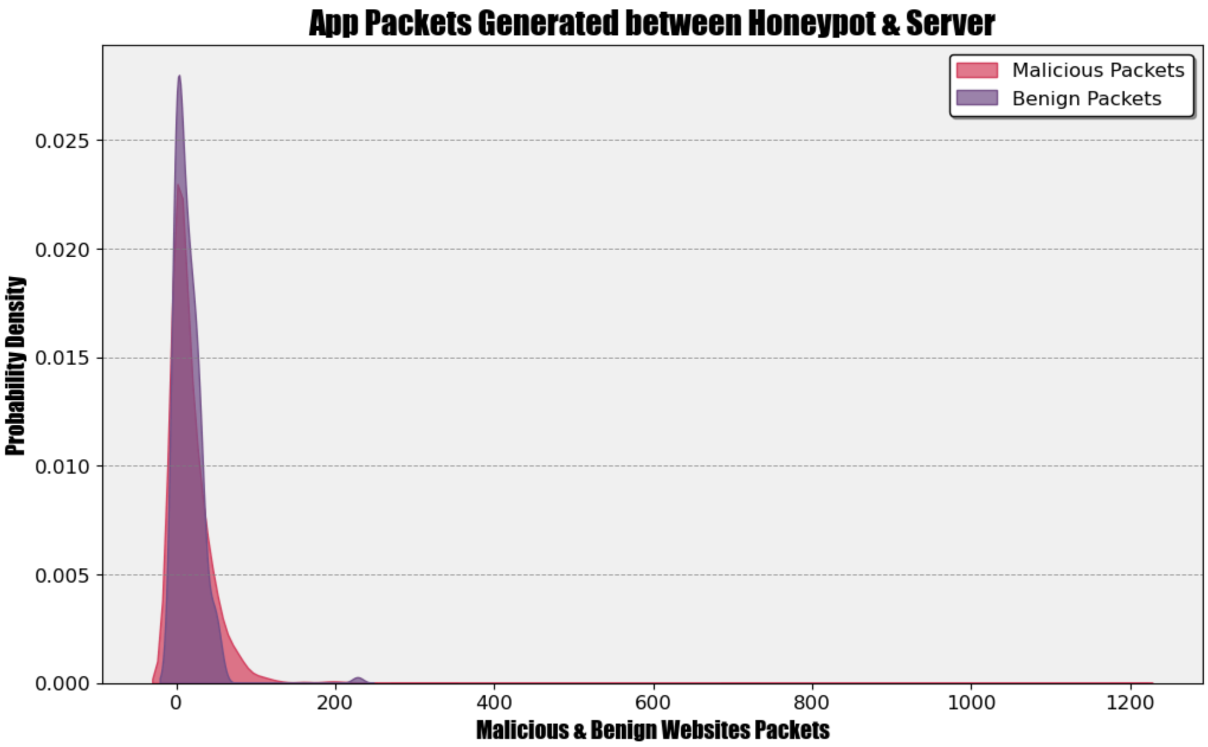
1. Is there a significant difference between benign websites and malicious websites total number of IP app packets generated during communication between the honeypot and server? (Mann-Whitney U Test)

H_0: *There is no difference between the total number of IP app packets generated during communication between the honeypot and server.*

H_1: *There is a difference between the total number of IP app packets generated during communication between the honeypot and server.*

In [ ]:

```
#Declaring variables to plot data.
benign_df = df_copy[BENIGN_WEBSITES]
benign_app_packets = benign_df['app_packets']
malicious_app_packets = malicious_df['app_packets']

#Call function to plot data.
multi_sample_test(dataset1=benign_app_packets,
                dataset2=malicious_app_packets,
                 x1=benign_app_packets,
                 x2=malicious_app_packets,
                 label1="Malicious Packets",
                 label2="Benign Packets",
                 title="App Packets Generated between Honeypot & Server",
                 xname="Malicious & Benign Websites Packets",
                 yname="Probability Density",
                      dataset3=None, x3=None, label3=None,
                      dataset4=None, x4=None, label4=None,
                      kde_color1='#D64161FF', kde_color2='#76528BFF', kde_color3='#
```



In [ ]:

```
#Call function to test outcome.
stat, p_val = mannwhitneyu(benign_app_packets, malicious_app_packets)
print(f"The P-Value calculated from the test is: {p_val}.")

alpha = 0.05

test_outcome(p_val, alpha=alpha)
```

```
The P-Value calculated from the test is: 0.6899548701963913.
```

Out[ ]:

```
'Fail to reject the null hypothesis.'
```

Interpretation

Based on our test outcome, we have found that there is no difference between the means of both groups of malicious and benign websites app packets generated.

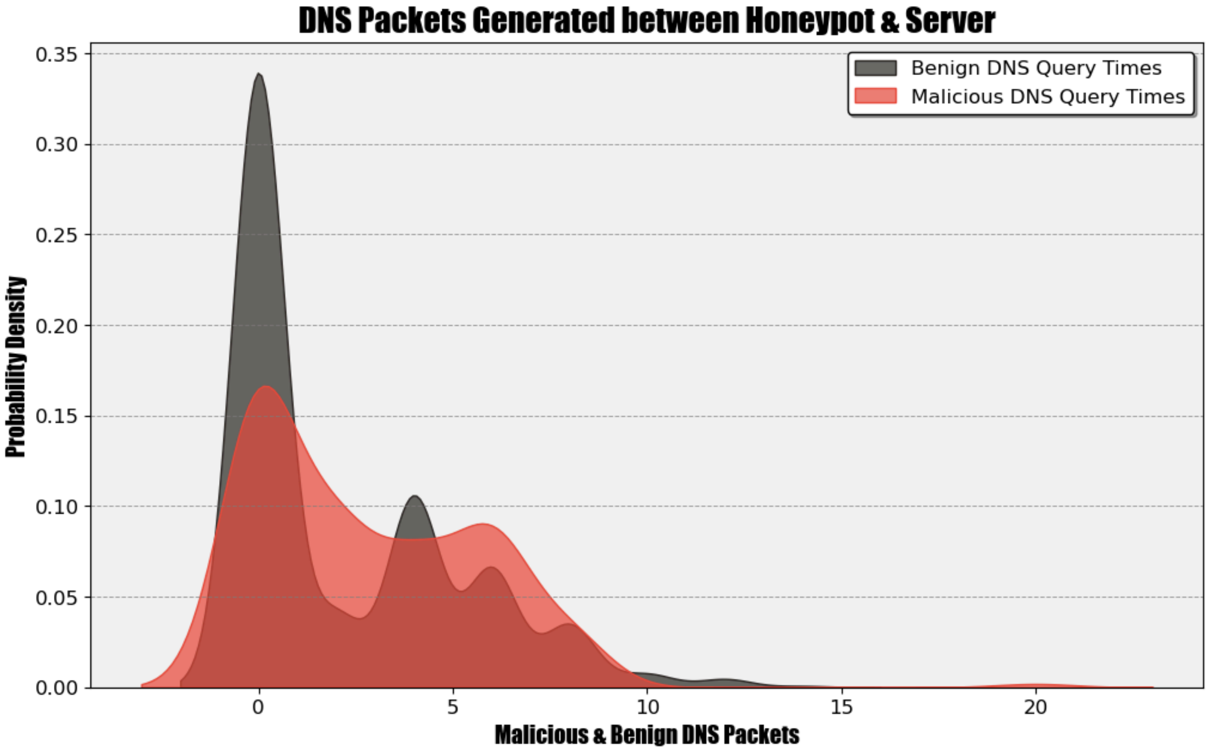
2. Is there a significant difference between benign websites and malicious websites DNS packets generated?

H_0: *There is no difference between the DNS packets generated for benign websites and malicious websites.*

H_1: *There is a difference between the DNS packets generated for benign websites and malicious websites.*

In [ ]:

```
#Declared variables to plot data
benign_query_times = benign_df['dns_query_times']
malicious_query_times = malicious_df['dns_query_times']

#Call function to plot data.
multi_sample_test(dataset1=benign_query_times,
                dataset2=malicious_query_times,
                 x1=benign_query_times,
                 x2=malicious_query_times,
                 label1="Benign DNS Query Times",
                 label2="Malicious DNS Query Times",
                 title="DNS Packets Generated between Honeypot & Server",
                 xname="Malicious & Benign DNS Packets",
                 yname="Probability Density",
                      dataset3=None, x3=None, label3=None,
                      dataset4=None, x4=None, label4=None,
                      kde_color1='#2D2926FF', kde_color2='#E94B3CFF', kde_color3='#
```



In [ ]:
```python
stat, p_val = mannwhitneyu(benign_query_times, malicious_query_times)
print(f"The P-Value calculated from the test is: {p_val}.")

alpha = 0.05

test_outcome(p_val, alpha=alpha)
```

```
The P-Value calculated from the test is: 8.972551409301633e-05.
```

Out[ ]:
```
'Reject the null hypothesis.'
```

Interpretation

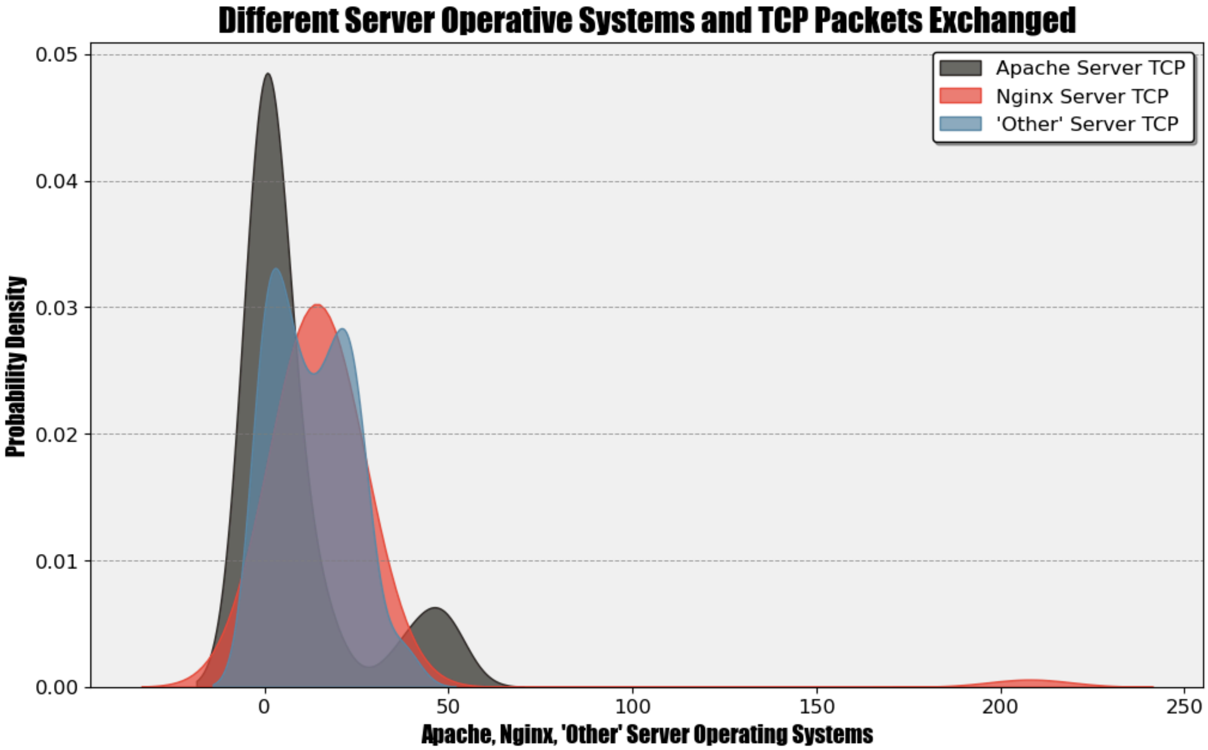
Based on our test outcome, we have found that there is a statistically significant difference between the means of both groups of malicious and benign websites DNS packets generated, and therefore warrants additional research.

3. For malicious websites, is there a significant difference for apache, nginx and other servers tcp packets exchanged? (Kruskal-Wallis H Test)

- $H_0 : \mu_{apache} = \mu_{nginx} = \mu_{other}$
- $H_a : H_0$ is not true

In [ ]:
```python
#Sort Data for plotting
malicious_apache_server = malicious_df[malicious_df["standardized_server"] == "apac
malicious_nginx_server = malicious_df[malicious_df["standardized_server"] == "nginx
```

```
malicious_other_server = malicious_df[malicious_df["standardized_server"] == "other

#Call function to plot data
multi_sample_test(dataset1=malicious_apache_server,
                dataset2=malicious_nginx_server,
                 x1=malicious_apache_server,
                 x2=malicious_nginx_server,
                 label1="Apache Server TCP",
                 label2="Nginx Server TCP",
                 title="Different Server Operative Systems and TCP Packets Exchange
                 xname="Apache, Nginx, 'Other' Server Operating Systems",
                 yname="Probability Density",
                      dataset3=malicious_other_server, x3=malicious_other_server, l
                      dataset4=None, x4=None, label4=None,
                      kde_color1='#2D2926FF', kde_color2='#E94B3CFF', kde_color3='#
```



In [ ]:

```
stat, p_value = kruskal(malicious_apache_server, malicious_nginx_server, malicious_
print(f"The P-Value calculated from the test is: {p_val}.")

test_outcome(p_value, alpha=alpha)
```

```
The P-Value calculated from the test is: 8.972551409301633e-05.
```

Out[ ]:

```
'Reject the null hypothesis.'
```

Interpretation

Based on our test outcome, we have found that the null hypothesis is not true and there is a statistically significant difference between the means of all groups, and therefore warrants additional research.

Section 4: Analysis & Conclusion

Analysis

The dataset was initially quite messy and required extensive data cleaning before it could be used for visualization. One of the significant tasks was standardizing the server column, which originally contained multiple versions for each specific operating system type. I cleaned and grouped these into a standardized format. Other columns also needed considerable attention. For instance, the whois_country column had inconsistent representations for countries, such as Great Britain being listed as "England," "Britain," and "Great Britain." Additionally, there were variations in text formats between uppercase and lowercase, necessitating further cleaning. After thorough data cleaning, the dataset was well-prepared for both descriptive and inferential analysis.

The descriptive analysis yielded interesting insights. The dataset revealed that there are nearly six times more benign websites than malicious ones. Spain hosts the most malicious websites, followed by the United States. In terms of operating systems, Apache and Nginx are the most popular among malicious websites. Nginx servers, in particular, are favored for their ability to handle the 'c10k' problem, which Apache's thread-based structure struggles with. This suggests that some malicious websites are designed to handle high traffic, possibly to maximize their impact. Additionally, UTF-8 was the main character encoding scheme used by malicious websites, followed by ISO-8859-1 (Latin-1). UTF-8 is a multibyte encoding that can represent any Unicode character, whereas ISO-8859-1 is a single-byte encoding. The use of UTF-8 indicates that malicious servers may require more versatile encoding capabilities.

For the inferential analysis, I formulated and tested three key questions. Initially, I generated Kernel Density Estimation (KDE) plots, which visually indicated that the data distributions were not normal and were mostly positively skewed. To confirm this, I conducted multiple Shapiro-Wilk tests, and in every case, the null hypothesis was rejected, confirming that the samples are not Gaussian. The first question I asked was whether there is a significant difference between the total number of IP app packets generated during communication between the honeypot and the server for benign and malicious websites. To test this, I conducted a Mann-Whitney U test, with the null hypothesis stating that there is no difference between the groups, and the alternative hypothesis stating that there is a difference. The result of the test led us to fail to reject the null hypothesis, indicating no significant difference between the means of both groups.

The second question I addressed was whether there is a significant difference between the DNS querying times for benign and malicious websites. Again, I used a Mann-Whitney U test, with the null hypothesis being that there is no difference in DNS querying times

> between the two groups, and the alternative hypothesis being that there is a difference. The test results led us to reject the null hypothesis, finding a statistically significant difference in DNS querying times between the two groups, which warrants further research.
>
> The final question I posed was whether there is a significant difference in TCP packets exchanged among Apache, Nginx, and other servers for malicious websites. For this, I conducted a Kruskal-Wallis H test. The null hypothesis stated that the mean TCP packets exchanged among these server types are equal, while the alternative hypothesis suggested otherwise. The results of the test led us to reject the null hypothesis, indicating a statistically significant difference in TCP packets exchanged among the three server types, which also warrants additional research.
>
> Overall, this exploratory data analysis was both challenging and enlightening. In the next section, I will explore machine learning techniques to see if we can develop a model that can predict whether a website is malicious or benign.

Resources / References

https://usa.kaspersky.com/resource-center/threats/what-is-a-honeypot

https://en.wikipedia.org/wiki/Honeypot_(computing)

https://www.cloudflare.com/learning/network-layer/what-is-a-packet/

https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/

https://www.ibm.com/topics/dns-protocol

https://www.nullhardware.com/blog/dns-basics/

https://www.hostinger.com/tutorials/what-is-apache

https://www.whois.com/whois/?srsltid=AfmBOoqzre7xOgzB3qPgqMSzAviQtvGHnpVXcqOHErNV3UPO5pthR73m

https://stackoverflow.com/questions/7048745/what-is-the-difference-between-utf-8-and-iso-8859-1