CT127-3-2-PFDA

Programming for Data Analysis

Group Assignment – Web Hacking Trends

Lecturer Name : Dr. Kulothunkan A/L Palasundram

Hand out Date : 5 December 2024

Hand in Date : 16 February 2025

Intake Code : APU2F2411CS(DA), APD2F2411CS(DA)

Group Members

Student ID	Student Name
TP068495	Lim Beng Rhui (*Group Leader*)
TP068983	Chan Chun Ming
TP068580	Eu Jun Hong
TP067678	Ng Xuan Jack

Table of Contents

Introduction

In today's digitalized world, where online activities and services flourish globally, cybersecurity awareness is integral to protecting one's online identity and privacy. As cybercrime investigators, our team aims to identify trends and patterns commonly found in web defacement attacks so that we can recommend actions for individuals and organizations to implement preventative measures, reducing their risks in the digital world.

Data Description

The analysis uses the `hackingData.csv` dataset, with its attributes listed in Table 1.

Table 1

Attributes in the `hackingData.csv` *Dataset*

Attribute	Description
Date	The date where the website defacement incident happened
Year *	The year of the occurrence of the website defacement incident
Month *	The month of the occurrence of the website defacement incident
Day *	The day of the occurrence of the website defacement incident
Notify	People or group who reported this incident
URL	URL link of the victim website
Protocol_Type *	Protocol type implemented by the website (HTTP or HTTPS)
Sector *	Sector / Domain category of the website, for example, • .gov is represented as Government.
URL_File_Type *	The file format linked to the website, categorized into several attributes, where • JPEG, PNG, JPG and similar files are classified as Media.
IP	IP address of the victim server
IP_Class *	The class of the IP address (Class A to E)
Country	Hosting country of the server encountering web defacement attacks
Continent *	Continent where the server's country is located at, such as Asia and Africa
OS	Operating system of the victim server
OS_Family *	The family of the operating system of the victim server
OS_Version *	The version of the operating system of the victim server
WebServer	Server used to host the website
WebServer_Version *	Version of the website server
Encoding	Encoding technique used in defacement message
Lang	Language of the victim website
Ransom	Amount paid to hackers in thousands
DownTime	Number of days that the system is down
Loss	Business revenue loss caused by the web defacement

Note. * means the attribute is a new attribute that is extracted based on another attribute.

Assumptions

Throughout the data analytics process, our group has made a few assumptions where

- it is assumed that all data are collected and recorded accurately without any human or measurement error,
- the missing values in the Loss attribute are treated as instances where data was not recorded, rather than indicating no loss, and
- no sampling bias is involved when the data is collected.

Hypothesis and Objectives

Our team has defined the following hypothesis for our investigation:

Null hypothesis (H_0)

Websites running Apache web servers on a Linux operating system will experience the same amount of loss as websites with baseline configurations.

Alternative hypothesis (H_1)

Websites running Apache web servers on a Linux operating system will experience a reduction in loss by at least 10% if compared to baseline configuration, with the reduction effect amplifying as the downtime increases.

To identify the relevant attributes for the hypothesis, our team has assigned the following objectives among ourselves:

Chan Chun Ming (TP068983)

I intend to investigate the impact of server downtime of different OS families towards revenue loss across years when facing web defacement attacks.

- How does OS family, downtime and year correlate with revenue loss?
- What is the proportion of each OS family that contributed to revenue loss due to web defacement attacks over the years?
- What is the trend of loss for each OS Family across years and which OS Family improves the most?
- How OS families react differently to loss when correlated with downtime?

Eu Jun Hong (TP068580)

I wish to analyse if the combination of specific OS and web server leads to higher loss.

- What is the relationship between OS family and loss?
- What is the relationship between web server and loss?
- What is the relationship between OS family and web server?

Lim Beng Rhui (TP068495)

I aim to study how the loss incurred due to web defacement attacks evolves across different timeframes based on website infrastructure characteristics derived from URLs and IP addresses.

- What is the correlation between temporal factors (year, month, and day), website information (protocol type, sector, file type, IP class), and web defacement loss?
- How does the average web defacement losses vary for different protocol types across different years?
- What is the relative contribution of each URL sector to the total web defacement loss across 5-year periods?
- What monthly trends exist in the loss distribution among different IP classes?

Ng Xuan Jack (TP067678)

I have in view on how year and country affect downtime due to web defacement attacks.

- What patterns/ trends can be observed via average loss caused by web defacement attacks over the years?
- What were the differences existing in average loss caused by web defacement attacks across various countries?
- Does relationship exist or associated across year, country, and the average loss due to web defacement attacks?

Data Preparation

Data Import

Data preparation begins by saving the dataset into the `hacking_data` variable using the `read.csv()` method after setting the working directory and loading all necessary libraries.

Figure 1

Code Snippet for Setting Working Directory, Importing Libraries and Retrieve Dataset

```
# Set working directory (modify if needed)
setwd("~/Documents/RStudio/PFDA Assignment")

# Load packages into session
library(MASS)          # For transformation
library(tidyverse)     # For data manipulation
library(dplyr)         # For data cleaning
library(VIM)           # Using kNN to predict missing value
library(countrycode)   # Standardize country name
library(stringdist)    # Fuzzy matching to match mispelled or similar value
library(lubridate)     # Helps to handle date-related issues
library(moments)       # To calculate skewness
library(mice)          # To impute empty data

# Resolve conflicts for functions (due to MASS library)
conflicted::conflict_prefer("select", "dplyr")
conflicted::conflict_prefer("filter", "dplyr")

# Load dataset to RStudio
hacking_data <- read.csv("hackingData.csv")
```

Figure 2

Output of Initial Dataset

	Date	Notify	URL IP	Country	OS	WebServer	Encoding	Lang	Ransom	DownTime	Loss
1	2/1/1998	Team CodeZero	http://www.janet-jackson.com	UNKNOWN	Unknown	Unknown	utf-8	NULL	NA	18	678.0
2	3/1/1998	Feliz	http://cariari.ucr.ac.cr	COSTARICA	Unknown	Unknown	NULL	NULL	NA	31	1988.0
3	4/1/1998	Optiklenz(LOU)	http://marin.k12.ca.us	UNITED STATES	Unknown	Unknown	iso-8859-1	Helvetica	2034	35	12204.0
4	4/1/1998	Team CodeZero	http://www.dm.af.mil	AFGHANISTAN	Unknown	Unknown	utf-8	NULL	NA	58	NA
5	4/1/1998	Team CodeZero	http://www.bolling.af.mil	AFGHANISTAN	Unknown	Unknown	windows-1252	Times New Roman	NA	59	NA
6	4/1/1998	foxy man	http://hope.hinet.net	UNKNOWN	Unknown	Unknown	utf-8	NULL	2316	34	13896.0
7	5/1/1998	Team CodeZero	http://www.nic.ad	ANDORRA	Unknown	Unknown	utf-8	NULL	195	21	390.0
8	5/1/1998	Zyklon	http://www.tamu-commerce.edu		Unknown		NULL	NULL	NA	37	1895.0
9	5/1/1998	net_	http://prOn.prizon.net	UNKNOWN	Unknown	Unknown	utf-8	NULL	2877	28	11508.0
10	7/1/1998	D.A.M.M.	http://www.unicef.org	UNKNOWN	Unknown	Unknown	windows-1252	NULL	1070	21	2140.0
11	8/1/1998	Optiklenz(LOU)	http://www.wvlc.wvnet.edu	UNKNOWN	Unknown	Unknown	utf-8	NULL	2760	33	16560.0
12	9/1/1998	viRtU3	http://www.trsystems.com		Unknown		NULL	NULL	NA	17	443.0
13	10/1/1998	Magica de Bin	http://www.webbnet.com	UNKNOWN	Unknown	Unknown	utf-8	NULL	NA	43	3509.0
14	11/1/1998	Optiklenz	http://www.easysoftware.com	UNKNOWN	Unknown	Unknown	utf-8	Helvetica	2406	15	1684.2
15	14/01/1998	Magica de Bin	http://www.itp.berkeley.edu	UNKNOWN	Unknown	Unknown	utf-8	NULL	902	39	5412.0
16	15/01/1998	Duncan Silver of L.O.U	http://www.emergent.com	UNKNOWN	Unknown	Unknown	utf-8	NULL	NA	50	2296.0
17	17/01/1998	I-h4k4	http://www.thebuzz.com	UNKNOWN	Unknown	Unknown	NULL	NULL	1898	34	11388.0
18	18/01/1998	Zyklon	http://www.kvtc.mtcs.tec.me.us		Unknown		NULL	NULL	NA	55	NA
19	19/01/1998	Older Generation	http://www.njcool.net	UNKNOWN	Unknown	Unknown	utf-8	NULL	NA	13	97.0
20	19/01/1998	SQuIrL & TdAwG	http://www.kalnet.com	UNKNOWN	Unknown	Unknown	utf-8	NULL	1809	40	10854.0

Data Preprocessing and Validation

With the dataset, we begin by removing unused attributes and eliminating duplicate records before replacing different empty values with the `NA` character to ensure consistency.

Figure 3

Code Snippet for Removing Unused Attributes, Remove Duplicates and Impute Empty Values

```
# Function that replace all invalid values with NA
replace_missing <- function(dataSet, columns) {
  dataSet %>%
    mutate(across(
      all_of(columns),
      ~ if_else(
        str_detect(as.character(.), "(?i)^Unkno|^null") | as.character(.) == "" | is.na(.),
        NA_character_,
        .
    )))
}

# Step 1: Only keep the attributes that we investigate
hacking_data_preprocessed <- hacking_data %>%
  select(
    "Date", "URL", "IP", "Country", "OS", "WebServer", "DownTime", "Loss"
  )

# Step 2: Remove duplicates
nrow(hacking_data_preprocessed) - n_distinct(hacking_data_preprocessed)   # Count duplicate
hacking_data_preprocessed <- distinct(hacking_data_preprocessed)          # Remove

# Step 3: Replace empty values (except DownTime and Loss) with NA
hacking_data_preprocessed <- replace_missing(
  hacking_data_preprocessed,
  c("Date", "URL", "IP", "Country", "OS", "WebServer")
)
```

Figure 4

Output After Removing Unused Attributes, Removing Duplicates and Imputing Empty Values

	Date	URL	IP	Country	OS	WebServer	DownTime	Loss
1	2/1/1998	http://www.janet-jackson.com	<NA>	<NA>	<NA>	<NA>	18	678.0
2	3/1/1998	http://cariari.ucr.ac.cr	<NA>	COSTARICA	<NA>	<NA>	31	1988.0
3	4/1/1998	http://marin.k12.ca.us	<NA>	UNITED STATES	<NA>	<NA>	35	12204.0
4	4/1/1998	http://www.dm.af.mil	<NA>	AFGHANISTAN	<NA>	<NA>	58	NA
5	4/1/1998	http://www.bolling.af.mil	<NA>	AFGHANISTAN	<NA>	<NA>	59	NA
6	4/1/1998	http://hope.hinet.net	<NA>	<NA>	<NA>	<NA>	34	13896.0
7	5/1/1998	http://www.nic.ad	<NA>	ANDORRA	<NA>	<NA>	21	390.0
8	5/1/1998	http://www.tamu-commerce.edu	<NA>	<NA>	<NA>	<NA>	37	1895.0
9	5/1/1998	http://pr0n.prizon.net	<NA>	<NA>	<NA>	<NA>	28	11508.0
10	7/1/1998	http://www.unicef.org	<NA>	<NA>	<NA>	<NA>	21	2140.0
11	8/1/1998	http://www.wvlc.wvnet.edu	<NA>	<NA>	<NA>	<NA>	33	16560.0
12	9/1/1998	http://www.trsystems.com	<NA>	<NA>	<NA>	<NA>	17	443.0
13	10/1/1998	http://www.webbnet.com	<NA>	<NA>	<NA>	<NA>	43	3509.0
14	11/1/1998	http://www.easysoftware.com	<NA>	<NA>	<NA>	<NA>	15	1684.2
15	14/01/1998	http://www.itp.berkeley.edu	<NA>	<NA>	<NA>	<NA>	39	5412.0

After preliminary cleaning, we address each attribute one by one, starting with Date. Using the `lubridate` package, we converted the date into the correct format before extracting its corresponding year, month and date.

Figure 5

Code Snippet for Converting Date into Consistent Format and Extract Year, Month and Day

```
# For Date, the issues are:
# - Date is not consistent, contains date in format of 2/1/1998 and 14/01/1998
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(across(
    all_of("Date"),              # Get each value in the date column
    dmy                          # Helps to standardize date in date-month-year format
  ))

# Add additional attributes for year, month and date
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    Year = year(Date),
    Month = month(Date, label = TRUE),
    Day = day(Date)
  )
```

Figure 6

Output After Converting Date and Extracting Year, Month and Day

	Date	URL	IP	Country	OS	WebServer	DownTime	Loss	Year	Month	Day
1	1998-01-02	http://www.janet-jackson.com	<NA>	<NA>	<NA>	<NA>	18	678.0	1998	Jan	2
2	1998-01-03	http://cariari.ucr.ac.cr	<NA>	COSTARICA	<NA>	<NA>	31	1988.0	1998	Jan	3
3	1998-01-04	http://marin.k12.ca.us	<NA>	UNITED STATES	<NA>	<NA>	35	12204.0	1998	Jan	4
4	1998-01-04	http://www.dm.af.mil	<NA>	AFGHANISTAN	<NA>	<NA>	58	NA	1998	Jan	4
5	1998-01-04	http://www.bolling.af.mil	<NA>	AFGHANISTAN	<NA>	<NA>	59	NA	1998	Jan	4
6	1998-01-04	http://hope.hinet.net	<NA>	<NA>	<NA>	<NA>	34	13896.0	1998	Jan	4
7	1998-01-05	http://www.nic.ad	<NA>	ANDORRA	<NA>	<NA>	21	390.0	1998	Jan	5
8	1998-01-05	http://www.tamu-commerce.edu	<NA>	<NA>	<NA>	<NA>	37	1895.0	1998	Jan	5
9	1998-01-05	http://prOn.prizon.net	<NA>	<NA>	<NA>	<NA>	28	11508.0	1998	Jan	5
10	1998-01-07	http://www.unicef.org	<NA>	<NA>	<NA>	<NA>	21	2140.0	1998	Jan	7
	...	...	...	...	...	...	...	...	...	...	...
11	2002-03-22	Freestyler??http://hebust.edu.cn	202.206.64.33	<NA>	<NA>	SolarisSunOS	24	2848.0	2002	Mar	22
12	2002-03-30	fs - Freestyler??http://www.unse.edu.ar	170.210.224.3	<NA>	<NA>	SolarisSunOS	12	100.0	2002	Mar	30
	...	...	...	...	...	...	...	...	...	...	...

Note. The yellow highlighted records contain URLs in an incorrect format.

Next, the URL attribute shows minimal errors; some URLs have words before them. The `sub()` function is implemented to remove the characters before the `http` characters.

Figure 7

Code Snippet for Removing Characters before `http` *Characters*

```
# For URL, the issues are:
# - Empty spaces at the front of data
# - Contains uncleaned URL (e.g. "Freestyler??http://hebust.edu.cn")
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    URL = sub(".*?(http)", "\\1", URL)         # Remove text before "http"
  )
```

Figure 8

Output After Text at Front of `http` *is Removed*

```
       Date                      URL              IP  Country           OS     WebServer DownTime Loss Year Month Day
        ...                      ...             ...      ...          ...           ...      ...  ...  ...   ... ...
1 2002-03-22     http://hebust.edu.cn   202.206.64.33     <NA>         <NA>  SolarisSunOS       24 2848 2002   Mar  22
2 2002-03-30   http://www.unse.edu.ar   170.210.224.3     <NA>  SolarisSunOS          <NA>       33 2916 2002   Mar  30
        ...                      ...             ...      ...          ...           ...      ...  ...  ...   ... ...
```

Figure 9

Code Snippet for Applying the `get_formatted_ip()` *Function on IP Attribute*

```
# For IP, no issues is found at first glance.
# - Try check if there is any invalid IP (if any octet does not fall between 0 to 255)
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    IP = sapply(IP, get_formatted_ip)        # Check for IP address validity
  )
```

Moving on to the IP attribute, no noticeable errors are found at first glance. However, we applied a function to check the octets to ensure validity. Fortunately, all IP addresses are in the correct format. Hence, no cleaning is necessary for the IP attribute.

Figure 10

The `get_formatted_ip()` *Function to Format IP Address*

```
# Function to check if IP address is in a valid form
get_formatted_ip <- function(ip_address) {

  # For NA, return the same value
  if (is.na(ip_address) || ip_address == "")  return (NA)

  # First check if the IP is in the correct format
  format <- "^\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}$"
  if (!str_detect(ip_address, format)) {
    return (NA)
  }

  # Split the input into octets based on the dots
  splitted_ip <- str_split(ip_address, "\\.", simplify = TRUE)
  splitted_ip <- as.numeric(splitted_ip)

  # Check if each octet is between 0 to 255
  if (all(splitted_ip >= 0 & splitted_ip <= 255)) return (ip_address)
  else return (NA)
}
```

For the Country attribute, due to its varying representation, we duplicate the data and use the `standardize_country()` method to calculate the Jaro-Winkler distance and return the best match. The results show that some countries were not appropriately transformed, so we manually handled the inaccuracies before cleaning the dataset. We also created a new Continent attribute for data representing continents instead of countries to prevent data loss.

Figure 11

Output of Using Duplicated Data for Transformation

	Country	New_Country
1	<NA>	<NA>
2	COSTARICA	Costa Rica
3	UNITED STATES	United States
4	AFGHANISTAN	Afghanistan
...	...	...
150	BANGLADESH	Bangladesh
151	GREENLAND	Greenland
152	CONGO	Togo
153	VIRGINISLANDS(U.S.)	Finland
154	FAROE ISLANDS	Faroe Islands
155	NAMIBIA	Namibia
156	KYRGYZSTAN	Kyrgyzstan
157	MICRONESIA	Indonesia
158	FIJI	Fiji
159	EASTTIMOR	Austria
160	CAPEVERDE	Cape Verde
161	AZERBAIJAN	Azerbaijan
162	TAJIKISTAN	Tajikistan
163	SAOTOMEANDPRINCIPE	San Marino
164	TURKMENISTAN	Turkmenistan
...	...	...

Note. The records highlighted in yellow represents misclassified values.

Figure 12

The `standardize_country()` *Method to Estimate Country Name*

```
# Function to standardize country name
standardize_country <- function(country) {

  # Return NA if input is missing
  if (is.na(country)) return(NA)

  # Use Jaro-Winkler distance to calculate match index
  distances <- stringdist::stringdist(
    tolower(country),                    # String to be compared with
    tolower(country_name_list),          # The list to be compared from
    method = "jw"                        # Method used is Jaro-Winkler string distance
  )

  # Retrieve the country name with smallest distance (highest similarity)
  best_index <- which.min(distances)
  best_match <- country_name_list[best_index]

  # Return the most matching country name
  return(best_match)
}
```

Figure 13

Code Snippet to Clean the Country Attribute

```
# For Country, the issues are:
# - Name written in different formats, e.g. "Taiwan", "TAIWAN", and "Taiwan, Prov"
# - Some does not represent country, e.g. "AMERICANSAMOA", "ASIA/PACIFIC REGION")
# - There are anonymous data, i.e. "Anonymous Proxy"

# First try to observe if the countries are transformed properly
country_test <- hacking_data_preprocessed %>%
  select("Country") %>%
  mutate(
    New_Country = sapply(Country, standardize_country)
  )
distinct(country_test)

# Notice that some countries are transformed incorrectly, and manual intervention is required
# Capitalize the countries
hacking_data_preprocessed$Country <- str_to_title(hacking_data_preprocessed$Country)

# We first create a new column to store continents (to avoid data loss)
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    Continent = case_when(
      Country %in% c("Europe", "European Uni", "European Union", "Westeuro", "Easteuro") ~ "Europe",
      Country %in% c("Asia", "Asia/Pacific Region", "Middleeast") ~ "Asia",
      Country == "Oseania" ~ "Oceania",
      Country == "Southamerica" ~ "Americas",
      Country == "Africa" ~ "Africa",
      TRUE ~ NA_character_
  ))

# Next, modify the country for those that cannot be detected
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(Country = case_when(
    Country == "Yugoslavia" ~ "Serbia",
    Country == "Virginislands(British)" ~ "British Virgin Islands",
    Country == "Macedonia" | Country == "Macedonia, T" ~ "North Macedonia",
    Country == "Macau" ~ "Macao SAR China",
    Country == "Ascensionisland" ~ "United Kingdom",
    Country == "Congo" ~ "Congo - Kinshasa",
    Country == "Virginislands(U.s.)" ~ "U.S. Virgin Islands",
    Country == "Micronesia" ~ "Micronesia (Federated States of)",
    Country == "Easttimor" ~ "Timor-Leste",
    Country == "Saotomeandprincipe" ~ "São Tomé & Príncipe",
    Country == "Syrian Arab Republic" ~ "Syria",
    Country == "Korea, Repub" | Country == "Korea" ~ "South Korea",
    Country == "America" ~ "United States",
    Country %in% c("Europe", "Asia/Pacific Region", "Anonymous Proxy", "Satellite Provider", "European Uni",
                   "European Union", "Virgin Islands", "Westeuro", "Easteuro", "Southamerica", "Asia",
                   "Oseania", "Middleeast", "Africa") ~ NA_character_,
    TRUE ~ Country
  ))

# Now use the standardize_country function to transform country
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    Country = sapply(Country, standardize_country)
  )

# Then, we map countries to continents using countrycode but skip NA
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    Continent = case_when(
      (!is.na(Country) & is.na(Continent)) ~ countrycode(Country, "country.name", "continent"),
      TRUE ~ Continent  # Keep existing values
    )
  )
```

Figure 14

Output Before and After Cleaning Country Attribute

Before

	Date	URL	IP	Country	OS	WebServer	DownTime	Loss	Year	Month	Day
	...	...	...	...	...	...	...	...	...	...	...
1	2011-06-11	http://www.i4cd.co.kr/help.html	112.216.93.62	Korea, Repub	Win 2	Apache	17	632.0	2011	Jun	11
2	2012-06-21	http://miiconsulting.com	92.61.152.68	European Uni	Linux	Apache	31	12342.0	2012	Jun	21
3	2013-10-06	http://www.johnbowenturf.com	141.101.116.58	European Union	Linux	cloudflare-nginx	19	4135.6	2013	Oct	6
4	2013-10-06	http://banque-togo.net/x.htm	208.91.197.46	Virgin Islands	F5 Big-IP	Apache	37	4608.0	2013	Oct	6
5	2013-10-09	http://www.alberah.com/main/alberah_index.php	90.153.255.252	Syrian Arab Republic	Linux	Apache	10	57.0	2013	Oct	9
6	2013-10-10	http://wapka.mobi	70.39.184.115	Satellite Provider	F5 Big-IP	nginx	32	1728.0	2013	Oct	10
7	2013-10-10	http://www.romapcexpress.com	130.117.233.105	European Union	Linux	Apache	22	997.0	2013	Oct	10
8	2013-10-10	http://www.lict.sy//	213.178.235.108	Syrian Arab Republic	Win 2008	Apache	21	910.0	2013	Oct	10
9	2013-10-10	http://www.bargaindvdset.com/blog/	208.91.197.101	Virgin Islands	F5 Big-IP	Apache	7	NA	2013	Oct	10
10	2013-10-10	http://www.woodworkersjournal.com/WIW_forum/	2.23.82.9	European Union	Win 2003	IIS/6.0	56	NA	2013	Oct	10
11	2013-10-10	http://anythingoes.in/wp-login.php	141.101.117.180	European Union	Linux	cloudflare-nginx	58	NA	2013	Oct	10
12	2013-10-10	http://www.2nbb.com	141.101.117.237	European Union	Linux	cloudflare-nginx	31	1292.0	2013	Oct	10
13	2013-10-11	http://gpfree.wapka.me	70.39.184.115	Satellite Provider	F5 Big-IP	nginx	32	1537.0	2013	Oct	11
14	2013-10-11	http://www.wepy.com/.php	141.101.116.225	European Union	Linux	cloudflare-nginx	27	831.0	2013	Oct	11
15	2013-10-11	http://islamicschoolsystem.com	208.91.197.54	Virgin Islands	F5 Big-IP	Apache	17	1352.0	2013	Oct	11
16	2013-10-11	http://almadmon.com	91.144.0.29	Syrian Arab Republic	Linux	nginx	7	NA	2013	Oct	11
	...	...	...	...	...	...	...	...	...	...	...

After

	Date	URL	IP	Country	OS	WebServer	DownTime	Loss	Year	Month	Day	Continent
	...	...	...	...	...	...	...	...	...	...	...	...
1	2011-06-11	http://www.i4cd.co.kr/help.html	112.216.93.62	South Korea	Win 2	Apache	17	632.0	2011	Jun	11	Asia
2	2012-06-21	http://miiconsulting.com	92.61.152.68	<NA>	Linux	Apache	31	12342.0	2012	Jun	21	Europe
3	2013-10-06	http://www.johnbowenturf.com	141.101.116.58	<NA>	Linux	cloudflare-nginx	19	4135.6	2013	Oct	6	Europe
4	2013-10-06	http://banque-togo.net/x.htm	208.91.197.46	<NA>	F5 Big-IP	Apache	37	4608.0	2013	Oct	6	<NA>
5	2013-10-09	http://www.alberah.com/main/alberah_index.php	90.153.255.252	Syria	Linux	Apache	10	57.0	2013	Oct	9	Asia
6	2013-10-10	http://wapka.mobi	70.39.184.115	<NA>	F5 Big-IP	nginx	32	1728.0	2013	Oct	10	<NA>
7	2013-10-10	http://www.romapcexpress.com	130.117.233.105	<NA>	Linux	Apache	22	997.0	2013	Oct	10	Europe
8	2013-10-10	http://www.lict.sy//	213.178.235.108	Syria	Win 2008	Apache	21	910.0	2013	Oct	10	Asia
9	2013-10-10	http://www.bargaindvdset.com/blog/	208.91.197.101	<NA>	F5 Big-IP	Apache	7	NA	2013	Oct	10	<NA>
10	2013-10-10	http://www.woodworkersjournal.com/WIW_forum/	2.23.82.9	<NA>	Win 2003	IIS/6.0	56	NA	2013	Oct	10	Europe
11	2013-10-10	http://anythingoes.in/wp-login.php	141.101.117.180	<NA>	Linux	cloudflare-nginx	58	NA	2013	Oct	10	Europe
12	2013-10-10	http://www.2nbb.com	141.101.117.237	<NA>	Linux	cloudflare-nginx	31	1292.0	2013	Oct	10	Europe
13	2013-10-11	http://gpfree.wapka.me	70.39.184.115	<NA>	F5 Big-IP	nginx	32	1537.0	2013	Oct	11	<NA>
14	2013-10-11	http://www.wepy.com/.php	141.101.116.225	<NA>	Linux	cloudflare-nginx	27	831.0	2013	Oct	11	Europe
15	2013-10-11	http://islamicschoolsystem.com	208.91.197.54	<NA>	F5 Big-IP	Apache	17	1352.0	2013	Oct	11	<NA>
16	2013-10-11	http://almadmon.com	91.144.0.29	Syria	Linux	nginx	7	NA	2013	Oct	11	Asia
	...	...	...	...	...	...	...	...	...	...	...	...

Figure 15

Code Snippet for Extracting OS Family and Version

```
# First convert the data to small case and remove any empty spaces if there is
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(OS = str_to_lower(OS)) %>%
  mutate(OS = lapply(OS, trimws))

# Then retrieve the OS family from OS
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    OS_Family = case_when(
      str_detect(OS, "windows|win") ~ "Windows",
      str_detect(OS, "linux") ~ "Linux",
      str_detect(OS, "macos|macosx|osx") ~ "macOS",
      str_detect(OS, "bsd|freeb|openbsd|netbs") ~ "BSD",
      str_detect(OS, "solar|sunos") ~ "Solaris",
      str_detect(OS, "aix|hp-ux|tru64|irix|sco unix|unix") ~ "Unix",
      str_detect(OS, "ios|cisco|juniper|junos") ~ "Cisco",
      str_detect(OS, "netware") ~ "NetWare",
      str_detect(OS, "os2") ~ "OS/2",
      str_detect(OS, "embedded|wap|router|firewall|f5|ipxe|citri|crestron|hp|modem|adapter|device|sgos") ~ "Embedded",
      TRUE ~ NA_character_,
  ))

# Also retrieve the version from OS and remove the initial column
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(OS_Version = mapply(retrieve_os_version, OS, OS_Family)) %>%
  select(-OS)

# Observe the number of records for each OS
count(hacking_data_preprocessed, OS_Family)
# Convert OS with small quantity as others
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(OS_Family = if_else(str_detect(OS_Family, "(?i)Cisco|Embedded|macOS|NetWare|OS/2"),
                             "Others",
                             OS_Family))
```

Next, we used regex to detect keywords and retrieve the corresponding OS family for the OS attribute, then obtained their versions with the `retrieve_os_version()` function. OS family with few records have been combined under the "Others" category. Despite BSD and Solaris belonging to Unix, we provide distinct categories for them due to their significant numbers.

Figure 16

The `retrieve_os_version()` *Function to Extract OS Version*

```
# A function to retrieve ther version of operating system
retrieve_os_version <- function(os_string, family) {

  os_string <- trimws(os_string)                          # Remove white spaces of the string and retrieve the split list
  os_split <- str_split(os_string, " ")[[1]]

  regex <- switch(family, "Windows" = "windows|win",      # Define regex for each OS family
                          "Linux" = "linux",
                          "macOS" = "macos|macosx|osx",
                          "BSD" = "bsd|freeb|openbsd|netbs",
                          "Solaris" = "solar|sunos",
                          "Unix" = "aix|hp-ux|tru64|irix|sco unix|unix",
                          "Cisco" = "ios|cisco|juniper|junos",
                          "Android" = "android",
                          "NetWare" = "netware",
                          "OS/2" = "os2",
                          "Embedded" = "embedded|wap|router|firewall|f5 b|ipxe|citri|crestron|hp|modem|adapter|device|sgos",
                          NA)

  if (is.na(regex)) { return(NA_character_) }             # Return NA if family is NA

  for (index in 1:length(os_split)) {

    if(!is.na(str_match(os_split[index], regex))) {       # If the element matches the regex

      if (index < length(os_split)) {                     # Further check if the element is not the last one

        version <- os_split[index + 1]                    # Retrieve the version

        # Perform modification for different os family - modify for Windows family
        if (family == "Windows") {
          if (!is.na(str_match(os_string, "7"))) version = "7"
          else if (!is.na(str_match(os_string, "2003"))) version = "2003"
          else if (!is.na(str_match(os_string, "pocketpc/ce"))) version = NA_character_

        # Modify for BSD family
        } else if (family == "BSD") {
          match_result <- str_match(os_string, "(\\d\\.\\d\\.\\d)")
          if (!is.na(match_result[1, 1])) version <- match_result[1, 1]     # Extract the matched value from the matrix
          else version <- NA_character_                                     # Or some other default value

        # Modify for other family
        } else if (family == "Embedded" || family == "Cisco" || family == "Unix") {
          match_result <- str_match(os_string, "\\d+\\..")
          if (!is.na(match_result[1, 1])) version <- match_result[1, 1]
          else version <- NA_character_

        # Modify for solaris family
        } else if (family == "Solaris") {
          match_result <- str_match(version, "(\\d+)/(\\d+)")
          if (!is.na(match_result[1, 3]) && match_result[1, 3] != "") version_last_part <- match_result[1, 3]
        }

        # Return the version after modification
        return(version)
  } } }

  # Return NA if condition is not fulfilled
  return(NA_character_)
}
```

Figure 17

Output Before and After OS Family and Version is Extracted

Before

	Date	URL	IP	Country	OS	WebServer	DownTime	Loss	Year	Month	Day	Continent
	...	...	...	...	...	...	...	...	...	...	...	...
1	2000-04-23	http://www.vipfe.gov.bo	<NA>	Bolivia	win nt9x	<NA>	14	50.0	2000	Apr	23	Americas
2	2000-05-06	http://www.cfc.gob.mx	<NA>	Mexico	win nt9x	<NA>	41	2169.0	2000	May	6	Americas
3	2000-05-15	http://www.ssp.df.gov.br	<NA>	Brazil	win nt9x	<NA>	38	1905.0	2000	May	15	Americas
4	2001-03-10	http://www.chcc.nsw.gov.au	<NA>	Australia	win nt9x	<NA>	36	2018.0	2001	Mar	10	Oceania
5	2002-11-09	http://www.cbtmail.com	208.8.36.91	United States	solaris 8	<NA>	54	NA	2002	Nov	9	Americas
6	2002-11-09	http://webmail.cbtcompanies.com	208.8.36.91	United States	solaris 8	<NA>	55	NA	2002	Nov	9	Americas
7	2013-10-06	http://www.transasianaxis.com/ina.php	68.171.214.116	United States	juniper ive os 7.x	Apache	58	NA	2013	Oct	6	Americas
8	2013-10-06	http://www.mtsn-cilendek.web.id/win32.html	103.247.8.28	Indonesia	ipxe 1.x	Apache	55	NA	2013	Oct	6	Asia
9	2013-10-15	http://point.basesolucoes.com.br/xampp/lang.tmp	177.52.16.22	Brazil	cisco ios 12.x	Apache	21	791.0	2013	Oct	15	Americas
10	2013-10-16	http://realchianti.com/Ir.html	81.19.48.37	United Kingdom	cisco pix os 8.x|6.x	<NA>	45	2081.0	2013	Oct	16	Europe
	...	...	...	...	...	...	...	...	...	...	...	...

After

	Date	URL	IP	Country	WebServer	DownTime	Loss	Year	Month	Day	Continent	OS_Family	OS_Version
	...	...	...	...	...	...	...	...	...	...	...	...	...
1	2000-04-23	http://www.vipfe.gov.bo	<NA>	Bolivia	<NA>	14	50.0	2000	Apr	23	Americas	Windows	nt9x
2	2000-05-06	http://www.cfc.gob.mx	<NA>	Mexico	<NA>	41	2169.0	2000	May	6	Americas	Windows	nt9x
3	2000-05-15	http://www.ssp.df.gov.br	<NA>	Brazil	<NA>	38	1905.0	2000	May	15	Americas	Windows	nt9x
4	2001-03-10	http://www.chcc.nsw.gov.au	<NA>	Australia	<NA>	36	2018.0	2001	Mar	10	Oceania	Windows	nt9x
5	2002-11-09	http://www.cbtmail.com	208.8.36.91	United States	<NA>	54	NA	2002	Nov	9	Americas	Solaris	8
6	2002-11-09	http://webmail.cbtcompanies.com	208.8.36.91	United States	<NA>	55	NA	2002	Nov	9	Americas	Solaris	8
7	2013-10-06	http://www.transasianaxis.com/ina.php	68.171.214.116	United States	Apache	58	NA	2013	Oct	6	Americas	Others	7.x
8	2013-10-06	http://www.mtsn-cilendek.web.id/win32.html	103.247.8.28	Indonesia	Apache	55	NA	2013	Oct	6	Asia	Others	1.x
9	2013-10-15	http://point.basesolucoes.com.br/xampp/lang.tmp	177.52.16.22	Brazil	Apache	21	791.0	2013	Oct	15	Americas	Others	12.x
10	2013-10-16	http://realchianti.com/Ir.html	81.19.48.37	United Kingdom	<NA>	45	2081.0	2013	Oct	16	Europe	Others	8.x
	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 18

Code Snippet for Cleaning WebServer and Extracting WebServer Version

```
hacking_data_preprocessed <- hacking_data_preprocessed %>%           # Place url to the URL column if the URL column is empty
  mutate(URL = if_else(!is.na(WebServer) & is.na(URL) & !is.na(str_extract(WebServer, "(?i)http[s]?://\\S+|www.\\S+|\\S+\\.com|\\S+\\.net")),
                       str_extract(WebServer, "(?i)http[s]?://\\S+|www.\\S+|\\S+\\.com|\\S+\\.net"),
                       URL),
         WebServer = if_else(!is.na(WebServer) & str_detect(WebServer, "(?i)http[s]?://\\S+|www.\\S+|\\S+\\.com|\\S+\\.net"),
                             NA_character_,
                             WebServer))

hacking_data_preprocessed <- hacking_data_preprocessed %>%           # Place IP address to the correct column
  mutate(IP = if_else(is.na(IP) & !is.na(WebServer) & str_detect(WebServer, "(\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3})"),
                      WebServer,
                      IP),
         WebServer = if_else(!is.na(WebServer) & str_detect(WebServer, "(\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3})"),
                             NA_character_,
                             WebServer))

hacking_data_preprocessed <- hacking_data_preprocessed %>%           # Remove invalid data
  mutate(WebServer = if_else(!str_detect(WebServer, "[a-zA-Z0-9]") | !is.na(str_match(WebServer, "&quot")),
                             NA_character_,
                             WebServer))

hacking_data_preprocessed <- hacking_data_preprocessed %>%           # Extract version in format of x.x.x to new list
  mutate(WebServer_Version = str_extract(WebServer, "(\\d+(\\.\\d+)+)"),
         WebServer = str_replace_all(WebServer, "v?(\\d+(\\.\\d+)+)(\\s*\\S+)*", ""))

hacking_data_preprocessed <- hacking_data_preprocessed %>%           # Remove all ending "/" and "-"
  mutate(WebServer = sapply(str_replace_all(WebServer, "[/-]+$", ""), trimws))

hacking_data_preprocessed <- hacking_data_preprocessed %>%           # Standardize the common webservers
  mutate(
    WebServer = case_when(
      str_detect(WebServer, "(?i)^apache") ~ "Apache",
      str_detect(WebServer, "(?i)microsoft.*iis|^iis|microsoftoffice|pws") ~ "Microsoft-IIS",
      str_detect(WebServer, "(?i)nginx|ngx_openresty|chamber88") ~ "nginx",
      str_detect(WebServer, "(?i)sun.*web|sun.*one|oracle.*web|oracle.*iplanet|glassfish|solarissunos|oracle") ~ "Sun-Oracle-Web-Server",
      str_detect(WebServer, "(?i)^litespeed$") ~ "LiteSpeed",
      str_detect(WebServer, "(?i)^zeus$") ~ "Zeus",
      str_detect(WebServer, "(?i)^lighttpd$") ~ "lighttpd",
      str_detect(WebServer, "(?i)^akamaighost$") ~ "AkamaiGHost",
      str_detect(WebServer, "(?i)^varnish$") ~ "Varnish",
      str_detect(WebServer, "(?i)gws|ghs") ~ "Google-Web-Server",
      str_detect(WebServer, "(?i)cloudflare") ~ "Cloudflare",
      str_detect(WebServer, "(?i)lotus.*domino") ~ "Lotus-Domino",
      str_detect(WebServer, "(?i)thttpd") ~ "thttpd",
      is.na(WebServer) ~ NA_character_,
      TRUE ~ "Other"
  ) )
```

Like the OS attribute, we categorized the WebServer attribute and extracted its version. We also noticed that some attributes contain URLs and IP addresses, so the information has been reorganized and placed back in the appropriate columns.

Figure 19

Output Before and After WebServer is Cleaned and WebServer_Version is Extracted

Before

	Date	URL	IP	Country	WebServer	DownTime	Loss	Year	Month	Day	Continent	OS_Family	OS_Version
	...	...	...	...	...	...	...	...	...	...	...	...	...
1	2001-11-04	http://www.xiangtan.gov.cn	<NA>	China	IIS/4.0	29	1009	2001	Nov	4	Asia	Windows	nt9x
2	2001-11-07	http://www.ferrari.co.za	<NA>	<NA>	IIS/4.0	44	3409	2001	Nov	7	<NA>	Windows	nt9x
5	2003-10-08	http://lisa.datalab.telge.kth.se	130.237.77.254	Sweden	IIS 5.0	47	3146	2003	Oct	8	Europe	Windows	2000
6	2003-10-08	http://mirteknoloji.com/isko.htm	212.98.197.249	Turkey	IIS 5.0	27	1029	2003	Oct	8	Asia	Windows	2000
4	2013-10-06	http://aemaq.cl	69.174.241.60	United States	lighttpd/1.4.28	26	1345.0	2013	Oct	6	Americas	Linux	<NA>
5	2013-10-09	http://orleans.sc.gov.br/RooteD.html	189.38.91.14	Brazil	lighttpd/1.4.26	54	NA	2013	Oct	9	Americas	Linux	<NA>
6	2013-10-09	http://www.aquamat.gr	144.76.159.110	Germany	lighttpd/1.4.32	60	NA	2013	Oct	9	Europe	BSD	<NA>
1	2013-10-09	http://store0.mobile88.co.id//indonesia.html	175.41.137.198	Singapore	chamber88 nginx/1.4.1	51	NA	2013	Oct	9	Asia	Linux	<NA>
	...	...	...	...	...	...	...	...	...	...	...	...	...

After

	Date	URL	IP	Country	WebServer	DownTime	Loss	Year	Month	Day	Continent	OS_Family	OS_Version	WebServer_Version
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	2001-11-04	http://www.xiangtan.gov.cn	<NA>	China	Microsoft-IIS	29	1009	2001	Nov	4	Asia	Windows	nt9x	4.0
2	2001-11-07	http://www.ferrari.co.za	<NA>	<NA>	Microsoft-IIS	44	3409	2001	Nov	7	<NA>	Windows	nt9x	4.0
5	2003-10-08	http://lisa.datalab.telge.kth.se	130.237.77.254	Sweden	Microsoft-IIS	47	3146	2003	Oct	8	Europe	Windows	2000	5.0
6	2003-10-08	http://mirteknoloji.com/isko.htm	212.98.197.249	Turkey	Microsoft-IIS	27	1029	2003	Oct	8	Asia	Windows	2000	5.0
4	2013-10-06	http://aemaq.cl	69.174.241.60	United States	lighttpd	26	1345.0	2013	Oct	6	Americas	Linux	<NA>	1.4.28
5	2013-10-09	http://orleans.sc.gov.br/RooteD.html	189.38.91.14	Brazil	lighttpd	54	NA	2013	Oct	9	Americas	Linux	<NA>	1.4.26
6	2013-10-09	http://www.aquamat.gr	144.76.159.110	Germany	lighttpd	60	NA	2013	Oct	9	Europe	BSD	<NA>	1.4.32
1	2013-10-09	http://store0.mobile88.co.id//indonesia.html	175.41.137.198	Singapore	nginx	51	NA	2013	Oct	9	Asia	Linux	<NA>	1.4.1
	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Moving on to the Downtime attribute, we found no empty values. After performing inspections via generating plots, inspecting skewness, and using IQR and *z*-score to identify outliers, all the results looked fine, and no data cleaning was performed.

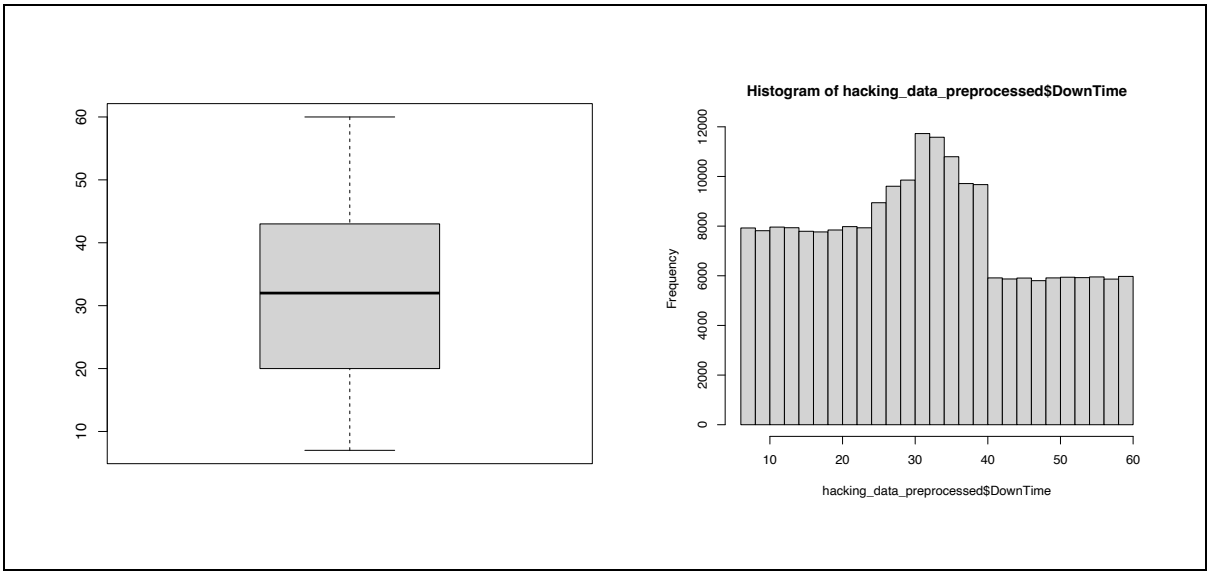
Figure 20

Code Snippet to Inspect Downtime

```
# Step 1: Have a glance at the downtime attribute using simple plots: boxplot and histogram
boxplot(hacking_data_preprocessed$DownTime)
hist(hacking_data_preprocessed$DownTime)

# Step 1a: Data seems normally distributed, calculate the skewness of the data
skewness(hacking_data_preprocessed$DownTime)           # Skewness: 0.1384, which is close to normal distribution

# Step 2: Use IQR to check for outliers
downtime_IQR <- IQR(hacking_data_preprocessed$DownTime, na.rm = TRUE)
cat("Lower bound smaller than minimum:",
    (quantile(hacking_data_preprocessed$DownTime, 0.25) - (1.5 * downtime_IQR)) <= min(hacking_data_preprocessed$DownTime) ,
    "\n",
    "Upper bound larger than maximum:",
    (quantile(hacking_data_preprocessed$DownTime, 0.75) + (1.5 * downtime_IQR)) >= max(hacking_data_preprocessed$DownTime),
    "\n"
  )

# Since the boundaries are -14.5 and 77.5, which exceeds the range of the data, no outliers are detected using IQR.

# Step 3: Check for outliers using z-score
downtime_z_scaled <- scale(hacking_data_preprocessed$DownTime)
cat("Exist z-score lower than -3:",
    min(downtime_z_scaled) < -3,
    "\n",
    "Exist z-score greater than 3:",
    max(downtime_z_scaled) > 3,
    "\n"
)
```

Figure 21

Boxplot and Histogram for Inspecting Downtime Distribution



Note. Both plots display a distribution which shows no skewness, no transformation required.

Figure 22

Text Output for Checking Skewness and Outliers

```
Skewness
[1] 0.138761

Checking outliers using IQR
Lower bound smaller than minimum: TRUE
 Upper bound larger than maximum: TRUE

Checking outliers using z-score
Exist z-score lower than -3: FALSE
 Exist z-score greater than 3: FALSE
```

Lastly, the Loss attribute, which contains missing values, is imputed using the MICE algorithm. A series of tests are applied next to determine the best data transformation method. Ultimately, our team found that the box-cox transformation is the most effective in adjusting the data from a right-skewed distribution to a more normal one. Hence, the transformation is applied before the winsorization technique is applied to remove outliers.

Figure 23

Code Snippet to Demonstrate the Use of MICE Algorithm to Impute Data

```
# Step 1: Investigate the number of empty values in loss
cat(
  "No of empty data:",
  sum(is.na(hacking_data_preprocessed$Loss)),
  "(",
  sum(is.na(hacking_data_preprocessed$Loss)) / length(hacking_data_preprocessed$Loss) * 100,
  "% )\n"
)

# Create predictor matrix
pred_matrix <- make.predictorMatrix(hacking_data_preprocessed)

# Perform imputation
mice_imputation <- mice(hacking_data_preprocessed,
                        m = 3,                          # Create 3 datasets with imputed values
                        maxit = 20,                     # Perform up to 20 times of iteration
                        method = "pmm",                 # Uses predictive mean matching as the imputation method
                        pred = pred_matrix)             # The variables involved

# Save the imputed data to the dataset
hacking_data_preprocessed <- complete(mice_imputation)

# Check if the imputation is successful
cat(
  "Imputation successful:",
  sum(is.na(hacking_data_preprocessed$Loss)) == 0,
  "\n"
)
```

Figure 24

Output after Data Imputation using MICE Algorithm

Before

	Date	Year	Month	Day	URL	IP	Continent	Country	OS_Family	OS_Version	WebServer	WebServer_Version	DownTime	Loss
1	1998-01-02	1998	Jan	2	http://www.janet-jackson.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	18	678.0
2	1998-01-03	1998	Jan	3	http://cariari.ucr.ac.cr	<NA>	Americas	Costa Rica	<NA>	<NA>	<NA>	<NA>	31	1988.0
3	1998-01-04	1998	Jan	4	http://marin.k12.ca.us	<NA>	Americas	United States	<NA>	<NA>	<NA>	<NA>	35	12204.0
4	1998-01-04	1998	Jan	4	http://www.dm.af.mil	<NA>	Asia	Afghanistan	<NA>	<NA>	<NA>	<NA>	58	NA
5	1998-01-04	1998	Jan	4	http://www.bolling.af.mil	<NA>	Asia	Afghanistan	<NA>	<NA>	<NA>	<NA>	59	NA
6	1998-01-04	1998	Jan	4	http://hope.hinet.net	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	34	13896.0
7	1998-01-15	1998	Jan	15	http://www.emergent.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	50	2296.0
8	1998-01-17	1998	Jan	17	http://www.thebuzz.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	34	11388.0
9	1998-01-18	1998	Jan	18	http://www.kvtc.mtcs.tec.me.us	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	55	NA
10	1998-01-19	1998	Jan	19	http://www.njcool.net	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	13	97.0
11	1998-01-19	1998	Jan	19	http://www.kalnet.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	40	10854.0

After

	Date	Year	Month	Day	URL	IP	Continent	Country	OS_Family	OS_Version	WebServer	WebServer_Version	DownTime	Loss
1	1998-01-02	1998	Jan	2	http://www.janet-jackson.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	18	678.0
2	1998-01-03	1998	Jan	3	http://cariari.ucr.ac.cr	<NA>	Americas	Costa Rica	<NA>	<NA>	<NA>	<NA>	31	1988.0
3	1998-01-04	1998	Jan	4	http://marin.k12.ca.us	<NA>	Americas	United States	<NA>	<NA>	<NA>	<NA>	35	12204.0
4	1998-01-04	1998	Jan	4	http://www.dm.af.mil	<NA>	Asia	Afghanistan	<NA>	<NA>	<NA>	<NA>	58	2385.0
5	1998-01-04	1998	Jan	4	http://www.bolling.af.mil	<NA>	Asia	Afghanistan	<NA>	<NA>	<NA>	<NA>	59	3964.0
6	1998-01-04	1998	Jan	4	http://hope.hinet.net	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	34	13896.0
7	1998-01-15	1998	Jan	15	http://www.emergent.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	50	2296.0
8	1998-01-17	1998	Jan	17	http://www.thebuzz.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	34	11388.0
9	1998-01-18	1998	Jan	18	http://www.kvtc.mtcs.tec.me.us	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	55	3236.0
10	1998-01-19	1998	Jan	19	http://www.njcool.net	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	13	97.0
11	1998-01-19	1998	Jan	19	http://www.kalnet.com	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	40	10854.0

Figure 25

Output for Comparison Results of Data Transformation Techniques for Loss Attribute

	Method	Skewness
1	Log	-0.92101946
2	Square root	1.00942934
3	Cube root	0.37283675
4	Box-cox	-0.07271249

Figure 26

Code Snippet Performing Different Data Transformation with Comparison among Techniques

```
# Step 0: Convert loss to numeric to ensure it can be used to count IQR
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(
    Loss = as.numeric(Loss)
  )

# Step 1: Have a glance at the downtime attribute using simple plots: boxplot and histogram
boxplot(hacking_data_preprocessed$Loss)
hist(hacking_data_preprocessed$Loss)

# Step 1a: Data is definitely not normally distributed, calculate the skewness of the data
skewness(hacking_data_preprocessed$Loss, na.rm = TRUE)      # Skewness: 2.3309, extremely right-skewed.

# Step 2: Perform different transformation to determine the best
# Step 2a: Perform log transformation
transformation_test <- hacking_data_preprocessed
transformation_test$Loss <- log1p(transformation_test$Loss)
transformation_log_skewness <- skewness(transformation_test$Loss, na.rm = TRUE)

# Step 2b: Perform square root transformation
transformation_test <- hacking_data_preprocessed
transformation_test$Loss <- sqrt(transformation_test$Loss)
transformation_sqrt_skewness <- skewness(transformation_test$Loss, na.rm = TRUE)

# Step 2c: Perform cube root transformation
transformation_test <- hacking_data_preprocessed
transformation_test$Loss <- (transformation_test$Loss) ^ (1/3)
transformation_cbrt_skewness <- skewness(transformation_test$Loss, na.rm = TRUE)

# Step 2d: Perform box-cox transformation
transformation_test <- hacking_data_preprocessed
boxcox_result <- boxcox(Loss ~ 1, data = transformation_test)
lambda <- boxcox_result$x[which.max(boxcox_result$y)]               # lambda = 2/9 in this case
transformation_test$Loss <- (transformation_test$Loss^lambda - 1) / lambda
transformation_box_cox_skewness <- skewness(transformation_test$Loss, na.rm = TRUE)

# Compare results
data.frame(
  Method = c("Log", "Square root", "Cube root", "Box-cox"),
  Skewness = c(
    transformation_log_skewness,
    transformation_sqrt_skewness,
    transformation_cbrt_skewness,
    transformation_box_cox_skewness)
)
```

Figure 27

Output After Appling Winsorization Technique with IQR

Before

	Date	URL	Loss
1	1998-01-08	http://www.wvlc.wvnet.edu	34.47458
2	1999-06-02	http://www.manateeisland.com	33.82183
3	1999-06-29	http://www.sc.gov.br	34.38953
4	1999-07-21	http://www.lawyerforlife.com	35.05249
5	1999-07-29	http://www.cedom.gov.ar	33.78515

After

	Date	URL	Loss
1	1998-01-08	http://www.wvlc.wvnet.edu	33.47957
2	1999-06-02	http://www.manateeisland.com	33.47957
3	1999-06-29	http://www.sc.gov.br	33.47957
4	1999-07-21	http://www.lawyerforlife.com	33.47957
5	1999-07-29	http://www.cedom.gov.ar	33.47957

Figure 28

Code Snippet to Perform Data Imputation and Handling Outliers using IQR and Winsorization

```
# Since box-cox yields the best result (closest to 0), applying box-cox to the dataset
boxcox_result <- boxcox(Loss ~ 1, data = hacking_data_preprocessed)
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
hacking_data_preprocessed$Loss <- (hacking_data_preprocessed$Loss^lambda - 1) / lambda

# Step 2a: Use IQR to check for outliers
loss_IQR <- IQR(hacking_data_preprocessed$Loss, na.rm = TRUE)
loss_IQR_lower <- quantile(hacking_data_preprocessed$Loss, 0.25, na.rm = TRUE) - (1.5 * loss_IQR)
loss_IQR_upper <- quantile(hacking_data_preprocessed$Loss, 0.75, na.rm = TRUE) + (1.5 * loss_IQR)

# Check if there exist any extreme values
cat("Lower bound smaller than minimum:",
    loss_IQR_lower <= min(hacking_data_preprocessed$Loss, na.rm = TRUE) ,
    "\n",
    "Upper bound larger than maximum:",
    loss_IQR_upper >= max(hacking_data_preprocessed$Loss, na.rm = TRUE),
    "\n"
)

# Step 2a - sub: Inspect the number of values that does not fall between the boundaries
cat("Percentage of data exceeding IQR boundaries:",
    sum(
      hacking_data_preprocessed$Loss > loss_IQR_upper |
        hacking_data_preprocessed$Loss < loss_IQR_lower,
        na.rm = TRUE) /
    sum(hacking_data_preprocessed$Loss, na.rm = TRUE) * 100,
    "%\n")

# Use winsorization technique to modify data
hacking_data_preprocessed <- hacking_data_preprocessed %>%
  mutate(Loss = case_when(
    Loss > loss_IQR_upper ~ loss_IQR_upper,
    Loss < loss_IQR_lower ~ loss_IQR_lower,
    TRUE ~ Loss
  ))
```

To simplify the preprocessing stage and reduce computational time, our team has exported the cleaned dataset into a new CSV file named `cleaned_hacking_data.csv`. Additionally, we have defined a new function called `quick_preprocessed_dataset()` so that the cleaned dataset is readily available and easily retrieved for the later data analysis stages.

Figure 29

Code Snippet for Exporting Cleaned Dataset with Function to Retrieve the Dataset

```
write.csv(hacking_data_preprocessed, "cleaned_hacking_data.csv", row.names = FALSE)      # Export cleaned file

quick_preprocessed_dataset <- function() {                                              # Function to retrieve the cleaned dataset
  setwd("~/Documents/RStudio/PFDA Assignment")                                          # Set working directory
  dataset_cleaned <- read.csv("cleaned_hacking_data.csv")                               # Read the cleaned dataset
  dataset_cleaned <- dataset_cleaned %>%                                                # Change data type
    mutate(Date = as.Date(Date),
           Year = factor(Year),
           Month = factor(Month),
           Day = factor(Day),
           Country = factor(Country),
           WebServer = factor(WebServer),
           Continent = factor(Continent),
           OS_Family = factor(OS_Family))
  return (dataset_cleaned)                                                              # Return the dataset
}
```

Data Analysis

With the cleaned dataset, our team performed exploratory data analytics to investigate the relationships between variables and the Loss attribute.

Objective 1:

I intend to investigate the impact of server downtime of different OS families towards revenue loss across years when facing web defacement attacks.

Prepared by: Chan Chun Ming (TP068983)

Further Data Preprocessing

The code in Figure 30 is used to do some preprocessing to achieve the objective. The number of each OS family is counted to check if there is any extreme small group, and k-Nearest Neighbour is applied to impute the missing value in OS_Family column.

Figure 30

Code Snippet to Impute Missing Values for OS Family

```
member_1_dataset <- quick_preprocessed_dataset()

count_OS_label <- member_1_dataset %>%       # Check number of records for each unique OS_Family
  count(OS_Family)
count_OS_label

member_1_dataset <- kNN(member_1_dataset,    # Use kNN to impute the missing value for OS_Family
                        variable = "OS_Family",
                        k = 5)               # Use the nearest 5 similar neighbours
```

Research Question 1:

How does OS family, downtime and year correlate with revenue loss?

To gain an overview of the combination of independent variables can how affecting loss, a multiple regression model is applied for **relationship analysis**. Since relationship analysis can determine if one changes affect the others, it is suitable in this context when we want to analyse the correlation between OS family, downtime, year and revenue loss.

Figure 31

Code Snippet to Build a Multiple Regression Model for Analysis

```
# Build a multiple regression model
lm_loss <- lm(Loss ~ DownTime + OS_Family + Year, data = member_1_dataset)
summary(lm_loss)
```

Figure 32

Summary of Multiple Linear Model for Downtime, OS family and Year towards Loss

```
Call:
lm(formula = Loss ~ DownTime + OS_Family + Year, data = member_1_dataset)

Residuals:
    Min      1Q  Median      3Q     Max
-9.0608 -2.6323 -0.9654  0.8308 15.1002

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      15.4432343  5.0327788   3.069  0.00215 **
DownTime          0.2847356  0.0007674 371.057  < 2e-16 ***
OS_FamilyLinux   -0.0358763  0.0756810  -0.474  0.63547
OS_FamilyOthers  -0.1832454  0.1744850  -1.050  0.29362
OS_FamilySolaris -0.0990902  0.1180611  -0.839  0.40129
OS_FamilyUnix    -0.1104065  0.1670215  -0.661  0.50859
OS_FamilyWindows -0.0462940  0.0764327  -0.606  0.54473
Year1999         -7.1024074  5.2778677  -1.346  0.17840
Year2000         -5.6050796  5.0327800  -1.114  0.26540
Year2001         -5.5317676  5.0322805  -1.099  0.27166
Year2002         -5.5446977  5.0322482  -1.102  0.27054
Year2003         -5.5082431  5.0324055  -1.095  0.27371
Year2005         -5.2774259  5.0346904  -1.048  0.29454
Year2006         -5.6138597  5.0346814  -1.115  0.26484
Year2007         -5.6885826  5.0346491  -1.130  0.25853
Year2008         -5.2265647  5.0469597  -1.036  0.30040
Year2009         -5.5719749  5.0322178  -1.107  0.26818
Year2010         -5.5456339  5.0339675  -1.102  0.27062
Year2011         -5.2287884  5.0347517  -1.039  0.29902
Year2012         -5.4751865  5.0345991  -1.088  0.27681
Year2013         -5.5530215  5.0322535  -1.103  0.26982
Year2014         -5.5355182  5.0324032  -1.100  0.27134
Year2015         -5.8523267  5.0509429  -1.159  0.24660
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.032 on 202084 degrees of freedom
  (9802 observations deleted due to missingness)
Multiple R-squared:  0.4053,         Adjusted R-squared:  0.4052
F-statistic:  6259 on 22 and 202084 DF,  p-value: < 2.2e-16
```

Downtime vs Loss. Scatter plot with regression line is generated with the code in Figure 33.

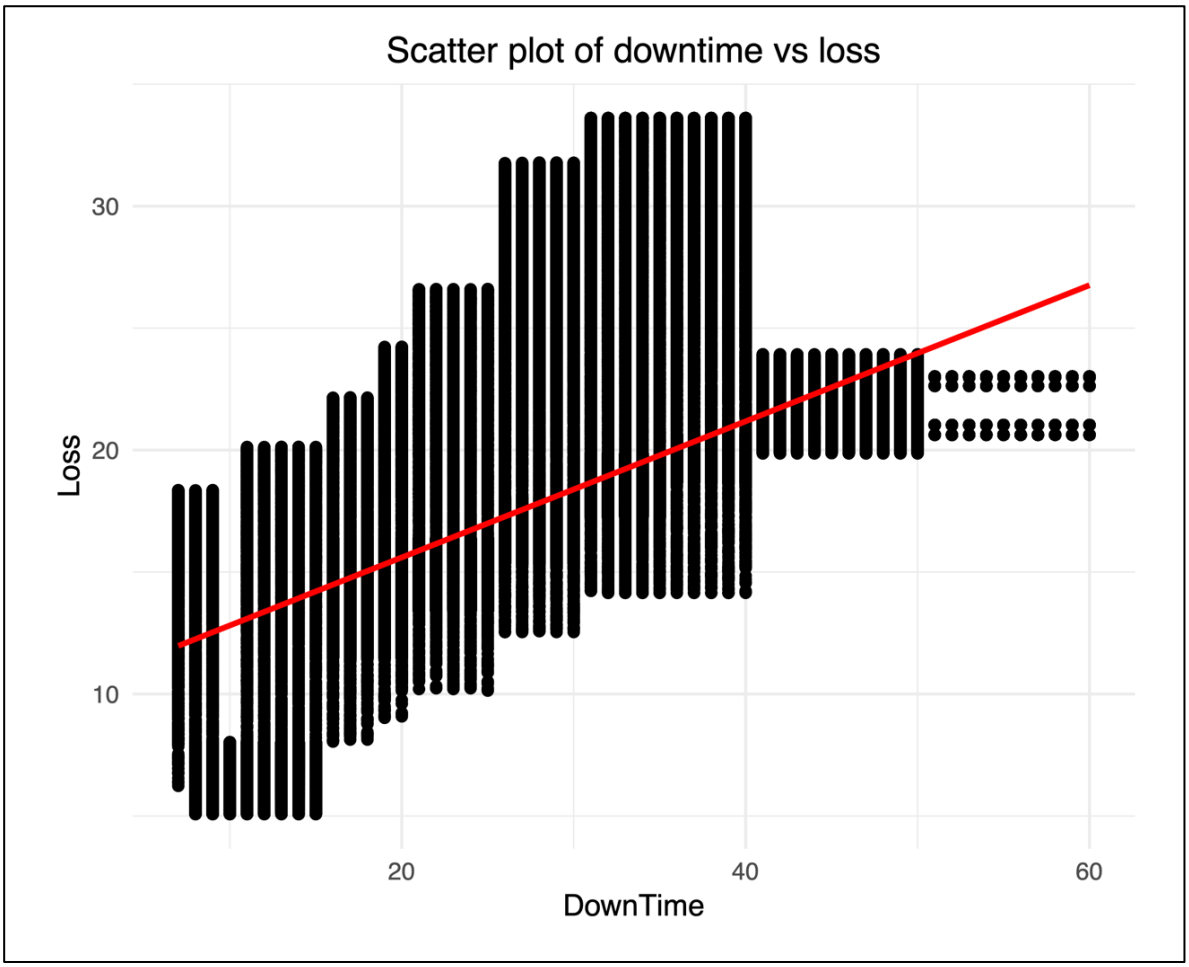
Figure 33

Code Snippet to Plot Scatter Graph between Downtime and Loss

```
ggplot(member_1_dataset, aes(x = DownTime, y = Loss)) +      # Scatter plot for downtime vs loss
  geom_point() +
  labs(
    title = "Scatter plot of downtime vs loss",
    xlab = "Downtime",
    ylab = "Loss"
  ) +
  geom_smooth(method = "lm", col = "red") +                  # Draw a regression line
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

# Check the correlation between downtime and loss
cor(member_1_dataset$DownTime, member_1_dataset$Loss, method = "pearson")
```

Figure 34

Scatter Plot for Downtime vs Loss



This scatter plot shows a positive linear regression line, indicates that a higher downtime leads to a higher revenue loss. To study the actual correlation between downtime and loss, Pearson's correlation is applied to check the strength of correlation between downtime and loss. As shown in the result below, 0.6366 represents a moderate to strong positive relationship between downtime and loss, which means when downtime increase by 1 day, loss will also increase by 0.6366 unit.

Figure 35

Pearson Correlation to Calculate Correlation between Downtime and Loss

```
> # Check the correlation between downtime and loss
> cor(member_1_dataset$DownTime, member_1_dataset$Loss, method = "pearson")
[1] 0.6365526
```

OS Family vs Loss. The distribution of average loss across each OS families will be plotted with box plot.

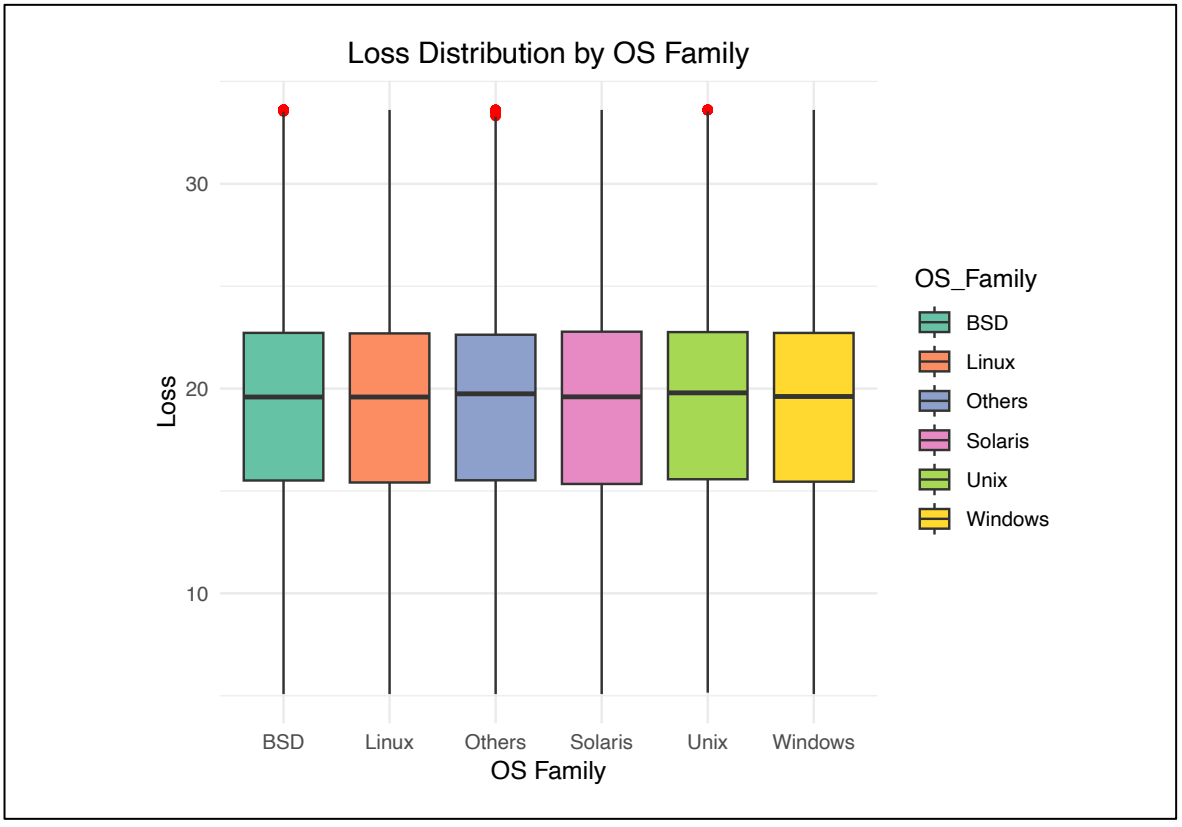
Figure 36

Code to Draw Box Plot to View Distribution of Loss for Each OS Family

```
#Visualize Loss distribution for each OS Family
ggplot(member_1_dataset, aes(x = OS_Family, y = Loss, fill = OS_Family)) +
  geom_boxplot(outlier.color = "red") +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal() +
  labs(title = "Loss Distribution by OS Family",
       x = "OS Family",
       y = "Loss") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 37

Distribution of Loss across all OS Family



From the box plot above, we can easily notice that the median and interquartile range for each OS families appear almost the same. Since the distribution for loss is quite consistent across each OS family, this indicates that family might not be a significant factor that can affect loss by itself as there should be differences in data spreads if OS Family is a significant variable.

OS Families towards Loss across Years. Heatmap is used to visualize combinations of OS families and year can how affect loss with the code below. The data is aggregated by grouping Year and OS Family to show the trend towards Loss.

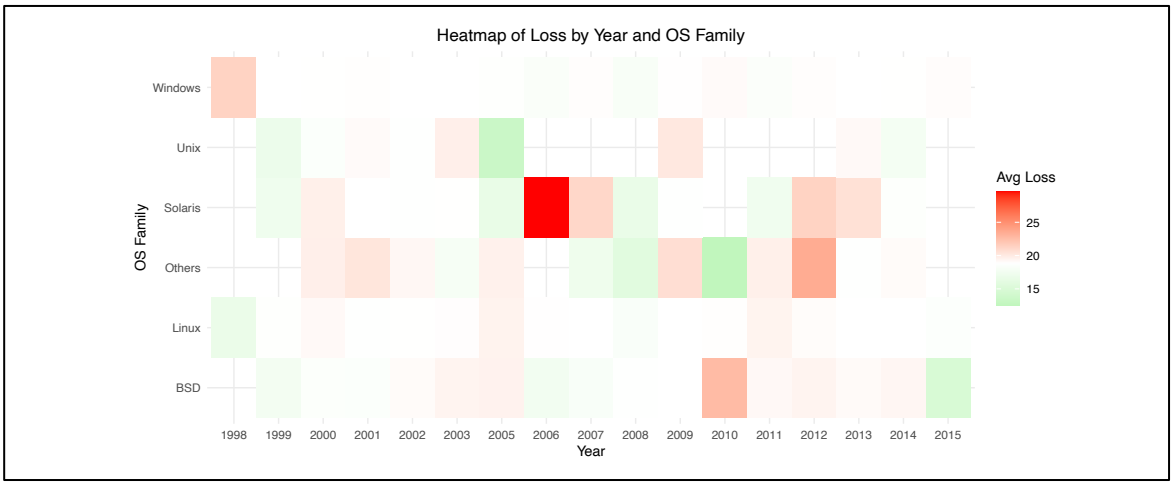
Figure 38

Code Snippet to Plot Heatmap for Loss in Each Year and OS Family

```
heatmap_data <- aggregate(
  Loss ~ Year + OS_Family, data = member_1_dataset, FUN = "mean"
)

# Heatmap Plot (Combination for year and os towards loss)
ggplot(heatmap_data, aes(x = Year, y = OS_Family, fill = Loss)) +
  geom_tile() +
  scale_fill_gradient2(low = "lightgreen", high = "red", mid = "white",
                       midpoint = median(heatmap_data$Loss)) +
  labs(
    title = "Heatmap of Loss by Year and OS Family",
    x = "Year", y = "OS Family",
    fill = "Avg Loss"
    ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 39

Heatmap of Loss by Year and OS Family



The heatmap above indicates that Solaris has the highest average loss on year 2006 with a red cell, but there is not much pattern observed for the else OS families. There is not any single OS family consistently shows a high average loss across years, only “Other” OS family is more frequently to have red cells, indicating higher average loss. The lack of overall trend shows there is no clear correlation between OS Family and Loss across different years.

Research Question 2:

What is the proportion of each OS family that contributed to revenue loss due to web defacement attacks over the years?

To validate the proportion for how each OS family contributes to loss across years, a stacked-bar chart is perfect in this scenario to carry out **distribution analysis**. The transition of each OS family can be observed easily when there are any significant changes on their distribution. The code below is used to generate the chart in Figure 41. The data is grouped by Year and OS Family then summarise with total Loss to view the proportion.

Figure 40

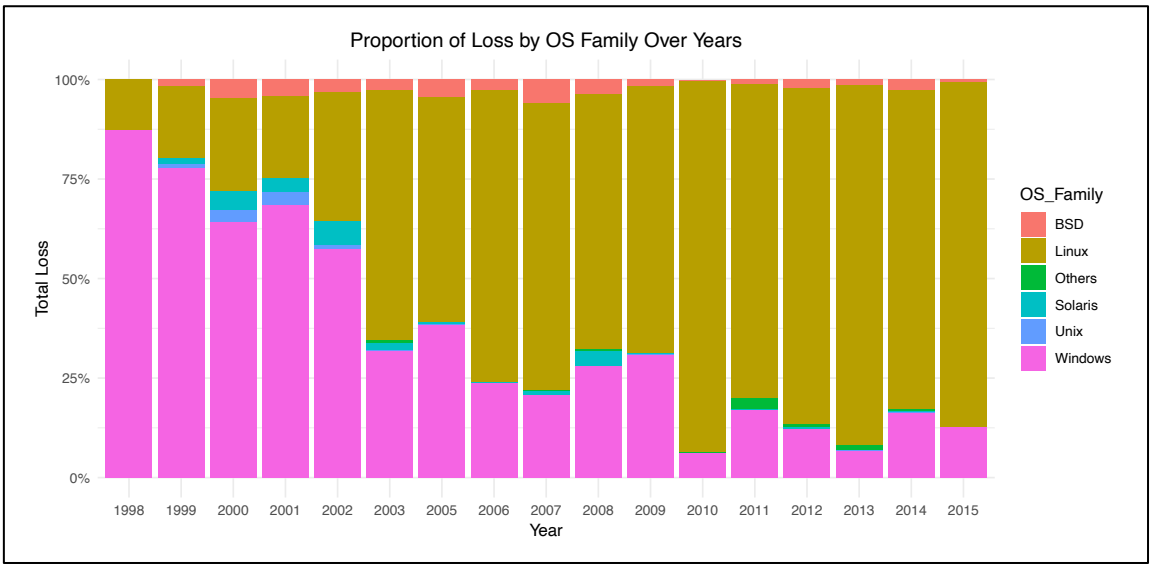
Code Snippet to Display Stacked Bar Chart of Proportion of Total Loss for Each OS Family

```
# Aggregate total loss for each OS Family per Year
loss_proportion <- member_1_dataset %>%
  group_by(Year, OS_Family) %>%
  summarise(
    Total_Loss = sum(Loss)
    ) %>%
  mutate(
    Proportion = Total_Loss / sum(Total_Loss)
    )  # Calculate proportion for each OS Family

#Stacked Bar to check the proportion for each OS Family
ggplot(loss_proportion, aes(x = Year, y = Proportion, fill = OS_Family)) +
  geom_bar(stat = "identity", position = "fill") +              #Stacked-bar
  scale_y_continuous(labels = scales::percent) +                   #Convert the scale into
proportion for frequency
  labs(
    title = "Proportion of Loss by OS Family Over Years",
    x = "Year",
    y = "Total Loss"
    ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

In early 1990s since 1993 when Windows NT 3.1 is founded, Windows servers were widely used for web hosting. This reflects in Figure 41 with the **highest proportion** for **Windows** (pink) from **1998 until 2002**, where the OS, having numerous vulnerabilities, has led to hackers targeting them more likely during the dawn of modern hacking in 2000s (Finnerty, 2023). For instance, the Code Red worm in 2001 mostly targeted Microsoft IIS as victims, causing thousands of servers, including most government or corporate websites, to facing web defacement (Industry Perspective, 2024).

Figure 41

Stacked Bar Chart Representing the Proportion of Total Loss for Each OS Family Across Years



Starting from 2003, due to its open-source nature, **Linux**-based servers like Apache or Nginx were highly adaptable as a hosting OS, making Linux started dominating the markets with the price of replacing Windows, which improved its security, to become the most dominant target for web defacement attacks, as reflected by the significant **increase** of **gold** portion (Williams, 2021). Web attackers can easily find exposed Linux servers due to misconfigured servers, enabling them to gain remote access via vulnerable PHP scripts like Web Shell Attacks in 2008 and Shellshock in 2014 (MrXcrypt, 2025).

On the other hand, the remaining OS had a minor presence until 2004, especially when Linux went public in 2003, their markets rapidly **turned down** (Ocelic, 2023). Looking into **Solaris OS** for internal enterprise applications, making them had a lower chance to get attacked by hackers, their role was replaced by Oracle's acquisition in 2010, resulting in **no cyan** portion after 2010. Also, **BSD**, which is known for security, has less exposure to web defacement attacks as attackers find easier to exploit Linux or Windows OS, resulting in a **small** but **consistent red** proportion across the years.

In short, **2003 – 2005** was a significant **transition** where Linux OS took over Windows OS roles to become the primary target, with the remaining OS like Solaris and BSD OS fading out since then.

Research Question 3:

What is the trend of loss for each OS Family across years and which OS Family improves the most?

Time-series analysis focuses on how a variable changes over time to identify trend which fits the analysis question. Since we want to analyze the loss trend by applying **time-series analysis**, line plot is the most suitable in this situation to visualize the loss trend over years. The R code below is used to generate faceted line plot in Figure 43.

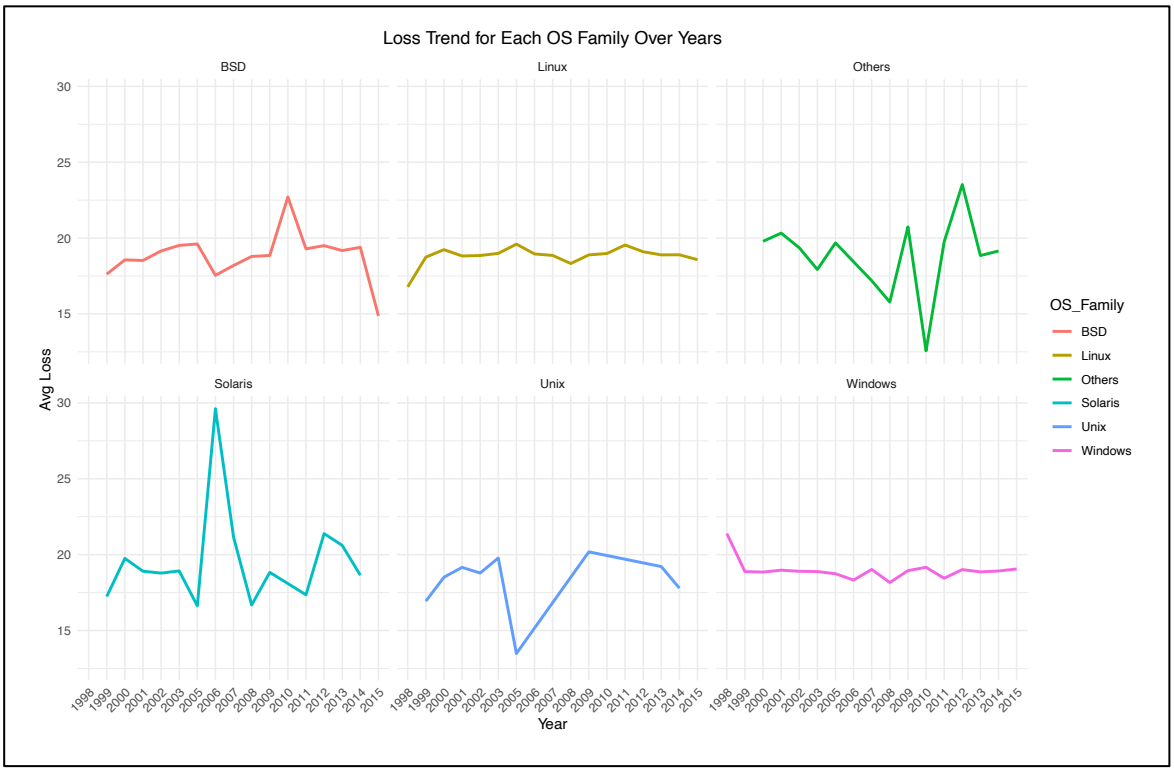
Figure 42

Code Snippet to Visualize Loss Trend of OS Family across Years

```
ggplot(heatmap_data, aes(x = Year, y = Loss, color = OS_Family, group = 1)) +
  geom_line(stat = "summary", fun = "mean", size = 1) +  #plot line based on average loss
  facet_wrap(~OS_Family) +                               #Split each OS Family
  labs(
    title = "Loss Trend for Each OS Family Over Years",
    x = "Year",
    y = "Avg Loss"
    ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) + #Slightly rotate the axis label
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 43

Faceted Line Plot for OS Family over Years against Average Loss



As **Windows** and **Linux** servers are dedicated to tackle security threats, they maintained a **consistent** loss trend, indicating they did not experience a large shift in loss. Meanwhile, **Unix** experienced sudden **drops** around year 2005, indicating that Unix has introduced a new secure configuration, leading to **Solaris** servers becoming the victim of large web defacement attacks with its sudden **spike** in graph (Jones, 2007). Besides, **BSD** servers have a **decline** of average loss starting from 2011, suggested that there might either be an enhancement or BSD is more rarely to be targeted by web defacement attacks. While "Other" OS servers show an inconsistent pattern on loss rate, this might be due to the low sample size and different variation of OS Family type grouped into this class.

Since there is no identifiable patterns or trends to be observed in any OS families, comparison between early years and later years of each OS family is performed. Years are separated into **two timeframes**, as shown in Figure 44, to plot the bar plot in Figure 45.

Figure 44

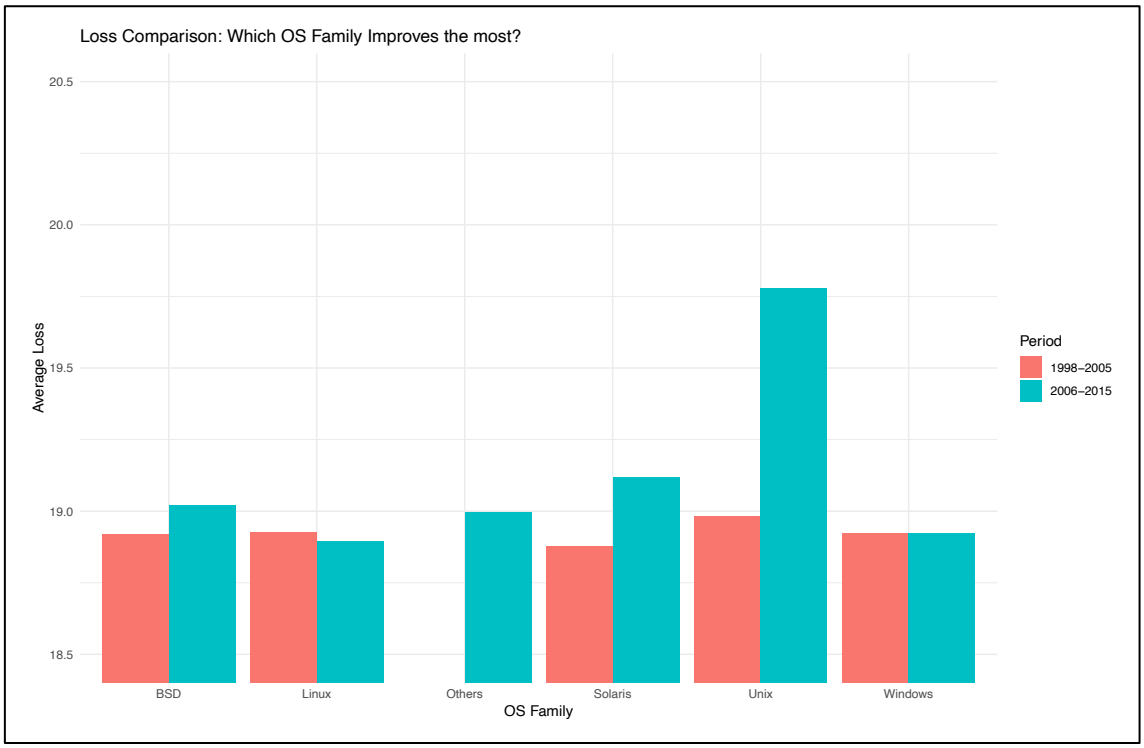
Code Snippet to Aggregate Timeframe and Plot Bar Chart

```
RQ3_m1 <- member_1_dataset %>%
  mutate(Year = as.numeric(as.character(Year)),                          # Convert Year from factor to numeric
         Period = case_when(Year >= 1998 & Year <= 2005 ~ "1998-2005",
                            Year >= 2006 & Year <= 2015 ~ "2006-2015",
                            TRUE ~ NA_character)) %>%_                   # Catch unexpected values
  group_by(OS_Family, Period) %>%
  summarize(Avg_Loss = mean(Loss), .groups = "drop")

#Bar plot to compare early years and late years for each OS family
ggplot(RQ3_m1, aes(x = OS_Family, y = Avg_Loss, fill = Period)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Loss Comparison: Which OS Family Improves the most?",
       x = "OS Family",
       y = "Average Loss",
       fill = "Period") +
  coord_cartesian(ylim = c(18.5, 20.5)) +    #Zoom the data within this 18.5 - 20.5 range for comparison
  theme_minimal()
```

From the bar plot, we can notice that all OS family shows an increase in the later years except for Linux servers. **Unix** and **Solaris** both shows a strong increase in loss in later years, especially for Unix. However, Unix servers had almost not existed in later years based on the proportion stacked bar plot in Figure 41, which indicates that the increased average loss may be due to a few high-loss incidents but not consistent trend. **BSD** OS servers also reflect in a small increase of average loss in later years, but did not indicate any significant trend. Moreover, **Windows** and **Linux** have similar average loss in both timeframes, reflecting that they did not experience major fluctuations over years but since Linux servers has a tiny decrease in average revenue loss, suggesting that **Linux** had **better security improvements** when tackling web defacement attacks.

Figure 45

Bar Plot to Compare Late and Early Years



Research Question 4:

How OS families react differently to loss when correlated with downtime?

Comparative analysis focuses on comparing different groups to check significant differences, which is applied in this analysis question differentiate the effect of different OS families and downtime leads to loss. To compare their performance, a scatter plot faceted by OS family is plotted in Figure 47 using the R code below:

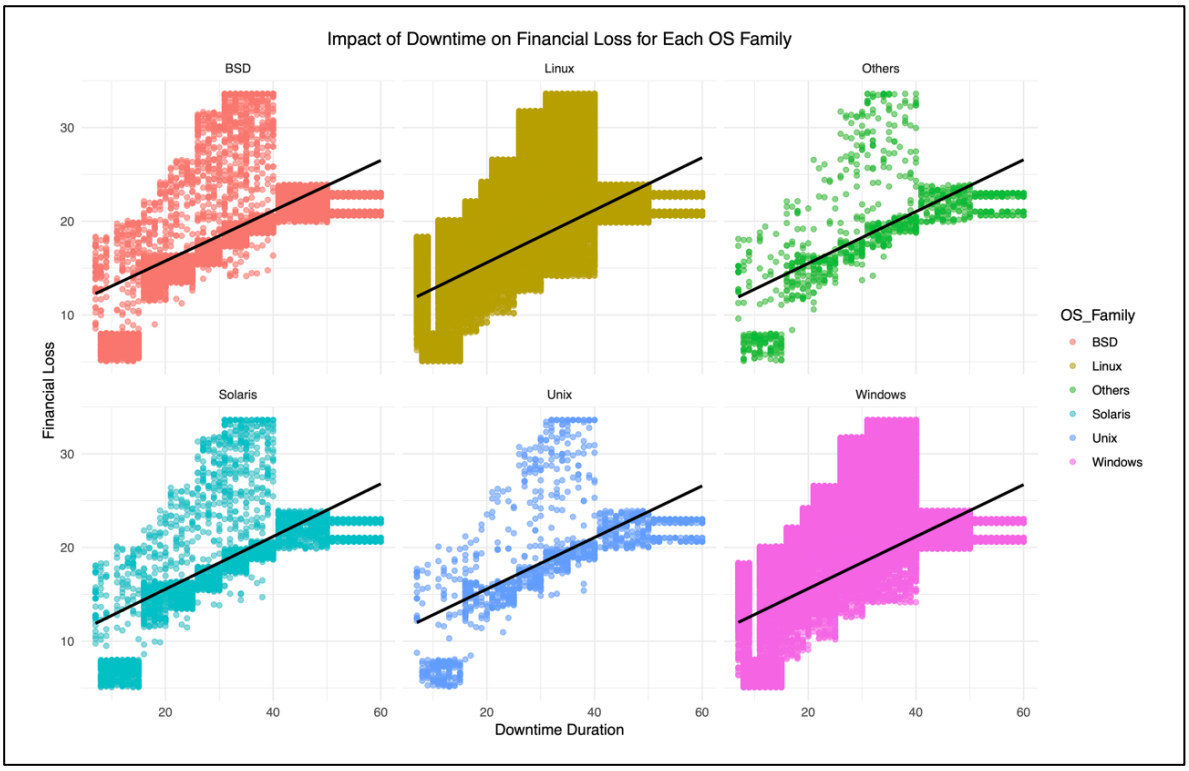
Figure 46

Code Snippet to Plot Faceted Scatter Plot

```
ggplot(member_1_dataset, aes(x = DownTime, y = Loss)) +
  geom_point(aes(color = OS_Family), alpha = 0.6) +
  geom_smooth(method = "lm",
              formula = y ~ x,
              se = FALSE,                              #Ignore shaded CI area
              color = "black") +
  facet_wrap(~ OS_Family) +                    #Group by OS family
  labs(title = "Impact of Downtime on Financial Loss for Each OS Family",
       x = "Downtime Duration",
       y = "Financial Loss") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 47

Scatter Plot to Show Impact of Downtime towards Loss across all OS Family



All OS families **show a positive relationship** between **downtime and loss**, indicated by the **upward slope** of the regression line. **Linux, Windows and Solaris OS** servers have a **steeper slope** compared to others, indicating a possibility of higher loss when there is an increase in downtime. This might be due to the unbalance variances for each OS family. Other than that, **Windows, Linux and BSD** servers had a bunch of **data points at high loss value** which shows that there were many high-cost incidents on these OS families. Overall, downtime has a strong relationship with loss across each OS family. Organizations should try to reduce the downtime for any OS they used as server host because there will be economic risks if their server has a frequent or long downtime.

Objective 2:

I wish to analyse if the combination of specific OS and web server leads to higher loss.

Prepared by: Eu Jun Hong (TP068580)

Research Question 1:

What is the relationship between OS family and loss?

Figure 48

Code Snippet to Retrieve OS Family and Loss

```
## Create ONLY "OS" and "Loss" dataset
Q1_data <- member_2_dataset %>%                  # Create a data frame that is called "Q1_data"
  filter(!is.na(OS_Family)) %>%                  # Get "OS_Family" data that is not a missing value
  mutate(OS_Family = droplevels(OS_Family)) %>%  # Removes all unused levels in OS_Family
  select(os_family = OS_Family, loss = Loss)     # Select OS_Family and Loss to display ONLY

## Summary Statistics
Q1_summary <- Q1_data %>%                        # Create a data frame called "Q1_summary"
  group_by(os_family) %>%                        # Group the data based on OS_Family
  summarise(                                     # Used to create summary statistics in data frame
    mean = mean(loss),                           # Calculate the mean / average of Loss
    variance = var(loss),                        # Calculate the variance of Loss
    standard_deviation = sd(loss),               # Calculate the standard deviation of Loss
    minimum = min(loss),                         # Search for the smallest number of loss
    number_of_OS = n()                           # Find the unique names of OS rows in OS_Family -> group by(os_family)
  )
```

Firstly, I filter out “OS_Family” to exclude all missing values in “OS_Family”. After that, mutate function allows me to use “droplevels” function to remove missing values. After a simple clean, I created a dataset which only selects “OS_Family” and “Loss” column and named it as “Q1_data”. After that, I have done a summary using “Q1_data” dataset. The summary contains some statistics results including mean, variance, standard deviation, the minimum loss and an additional column called “Number of Operating System” to count how many Operating System does “Q1_data” dataset have for each OS_Family on record. This summary will be used to display graphical output for visualization purposes.

Figure 49

Summary Statistics of OS Family

	os_family	mean	variance	standard_deviation	minimum	number_of_OS
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	BSD	19.0	40.7	6.38	5.08	4675
2	Linux	18.9	42.6	6.52	5.08	129670
3	Others	18.9	40.5	6.37	5.08	1025
4	Solaris	18.9	42.7	6.53	5.08	3043
5	Unix	19.0	42.2	6.49	5.15	1145
6	Windows	18.9	42.8	6.54	5.08	62549

Figure 50

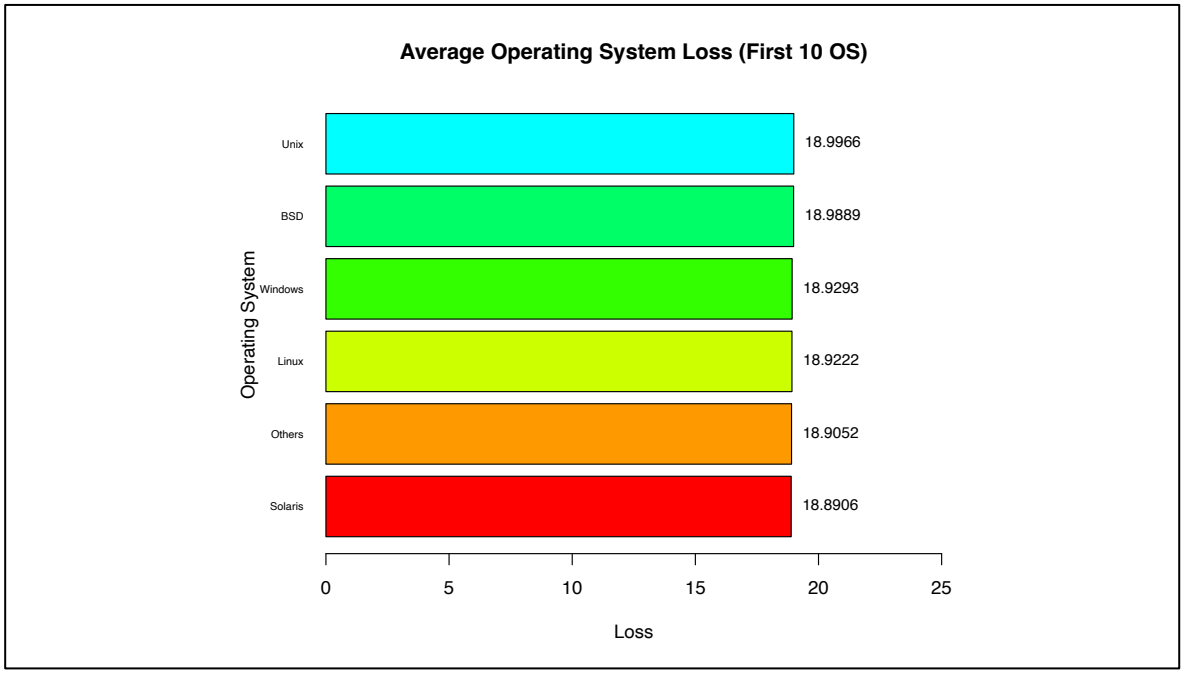
Code Snippet to Generate Bar Chart for the Top 10 OS with the Highest Average Loss

```
Q1_graph <- Q1_summary %>% # Create a data frame and call it "Q1_graph"
  arrange(mean) %>%  # Sort by average loss
  slice(1:10)  # Display ONLY first 10 OS

### Barplot Code
Q1_barplot <- barplot(Q1_graph$mean,                 # Create a barplot for mean
                      names.arg = Q1_graph$os_family, # Plot "OS_Family" specifically in barplot
                      main = "Average Operating System Loss (First 10 OS)", # Title of the barplot
                      xlab = "Loss",                 # Label at x-axis
                      ylab = "Operating System",     # Label at y-axis
                      xlim = c(0, 25),               # X-axis Grid with scale from 0 to 25
                      col = rainbow(10),             # Set the bars display in rainbow color
                      horiz = TRUE,                  # Display barplot horizontally
                      las = 1,                       # Make labels shows horizontally
                      cex.names = 0.6)               # "Operating System" word size

# Adjust text placement
text(x = Q1_graph$mean,                   # Set x-axis as mean of loss
     y = Q1_barplot,                      # Set y-axis as barplot
     labels = round(Q1_graph$mean, 4),    # Round up average loss to 4 decimal places
     pos = 4,                             # Display average loss to the right
     offset = 0.5,                        # Distance between bars and text
     cex = 0.8)                           # Reduce text size
```

By using this code, I created a barplot that holds "Loss" as average loss on *x*-axis and "OS_Family" as Operating System on y-axis. Before creating the barplot, I have sort "mean" ascending that refers to average loss of OS and use slice function to display only first 10 Operating System because I only want to know the shortest average loss for Operating System that have been recorded.

Figure 51

Bar Chart Representing the Top 10 Operating Systems with the Highest Average Loss



Based on the barplot, we can see there is only 10 results and "OS/2" is shown as having the shortest average loss among all Operating Systems. The result of "OS/2" displayed 16.6576 as average loss which is at least two unit better than other Operating System including macOS which is 18.7349, Linux which is 18.908 and Windows which is 18.9206.

Regarding the barplot, I have set the *x*-axis's grid starting from zero and ends on twenty-five units because there is no average loss exceed this number and scale every five units to best fit this graph. I have set this barplot display horizontally because the long name of Operating Systems causing overlay problems. I have also set the numbers beside bars to four decimal places because some of the average loss bars are too close to each other, especially from Unix (third bar) to macOS (last second bar).

Research Question 2:

What is the relationship between web server and loss?

Figure 52

Code Snippet to Retrieve WebServer and Loss Attributes with Summary Statistics

```
## Create ONLY "WebServer" and "Loss" dataset
Q2_data <- member_2_dataset %>%                      # Create "Q2_data" data frame
  filter(!is.na(WebServer)) %>%                     # Display "WebServer" that have values only
  mutate(WebServer = droplevels(WebServer)) %>%     # Remove all unused level in WebServer
  select(ws = WebServer, loss = Loss)               # Display only "WebServer" and "Loss" column

## Summary statistics
Q2_summary <- Q2_data %>%                 # Create "Q2_summary" data frame
  group_by(ws) %>%                        # Group by "WebServer"
  summarise(                              # Create summary statistics
    mean = mean(loss),                    # Calculate mean / average of loss
    variance = var(loss),                 # Calculate variance of loss
    standard_deviation = sd(loss),        # Calculate the standard deviation of loss
    minimum = min(loss),                  # Search for the least loss
    number_of_WebServer = n()             # Display unique names for WebServer
  )
```

I have created a dataset and named it "Q2_data" that has "WebServer" and "Loss" column only. I have redefined "WebServer" as "ws" and "Loss" as "loss" to help me focus on finding out the relationship between WebServer and Loss only.

Then, I use summarize function to find the result for mean, variance, standard deviation, minimum loss and number of WebServer loss for Q2 graph output. This summary is named "Q2_summary" dataset and will be used on the barplot later. By creating "Q2_summary" dataset, we can find out the statistics for "WebServer" and "Loss" such as average loss and number of WebServer that provide useful data for barplot.

Figure 53

Summary Statistics of the WebServer Attribute

	ws	mean	variance	standard_deviation	minimum	number_of_WebServer
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	AkamaiGHost	21.7	57.7	7.59	14.8	3
2	Apache	18.9	42.5	6.52	5.08	114237
3	Google-Web-Server	19.1	29.7	5.45	6.92	20
4	lighttpd	17.7	38.8	6.23	5.15	61
5	LiteSpeed	18.6	41.9	6.47	5.08	3073
6	Lotus-Domino	15.7	25.1	5.01	6.84	7
7	Microsoft-IIS	18.9	43.1	6.56	5.08	29723
8	nginx	18.9	42.7	6.53	5.08	5431
9	Other	19.3	45.7	6.76	5.35	358
10	Sun-Oracle-Web-Server	16.8	38.3	6.19	6.88	28
11	thttpd	19.5	14.6	3.82	15.1	7
12	Varnish	15.6	NA	NA	15.6	1
13	Zeus	19.8	41.9	6.48	5.42	75

Lastly, I have implemented “Q2_summary” dataset’s statistics into a barplot. By referring to this barplot, we can see that the shortest WebServer Loss is “Varnish” which average loss is 15.588 units. It means that “Varnish” is the first WebServer option to use because it makes lesser loss comparing with WebServer like “Google-Web-Server”, “Apache” and “Microsoft-IIS”.

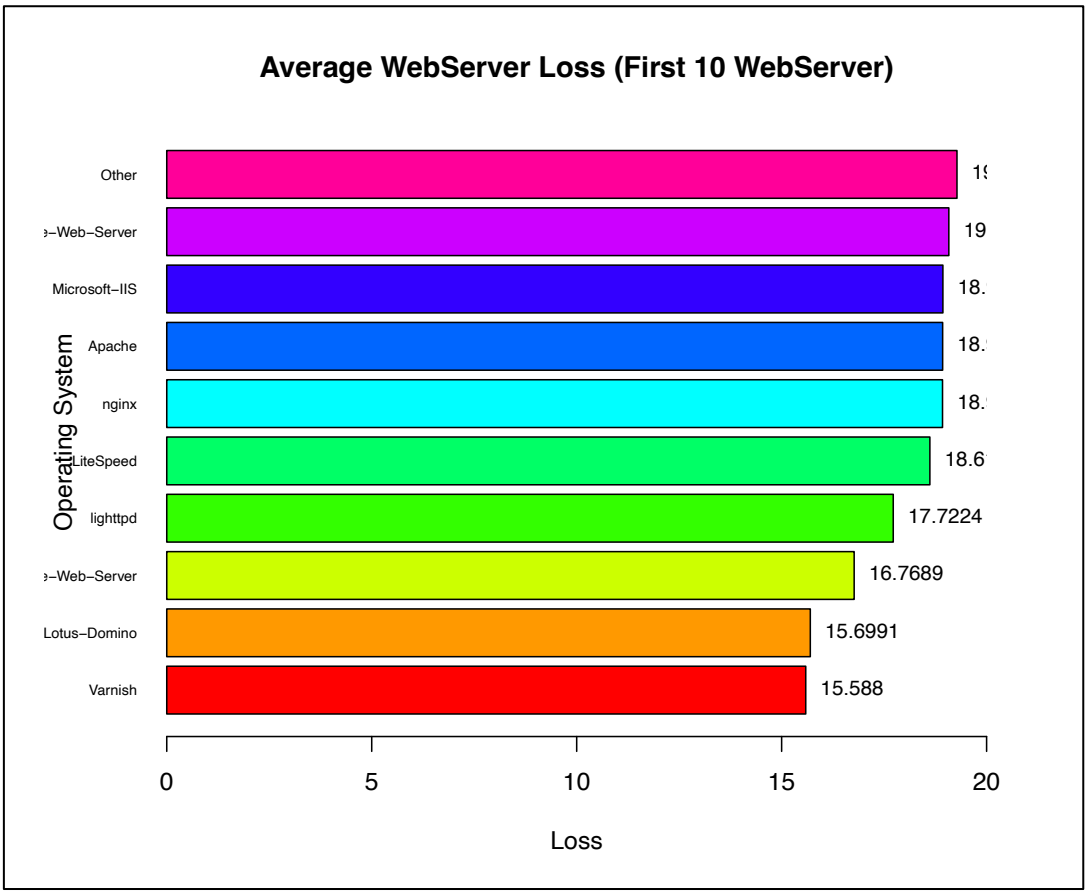
Figure 54

Code Snippet to Generate Barplot for Top 10 Web Servers with the Highest Average Loss

```
## Barplot
Q2_data <- Q2_summary %>%     # Create "Q2_data" data frame
  arrange(mean) %>%           # Arrange "Q2_data" based on mean of loss ascendingly
  slice(1:10)                 # Display only the first 10 of the data frame

barplot <- barplot(Q2_data$mean,                # Create "barplot" data frame <- use mean of loss from "Q2_data"
                   names.arg = Q2_data$ws,      # Plot WebServer as x-axis in barplot
                   main = "Average WebServer Loss (First 10 WebServer)", # Title of barplot
                   xlab = "Loss",               # Label at x-axis
                   ylab = "Operating System",   # Label at y-axis
                   xlim = c(0, 20),             # Display a-axis grid from 0 to 20
                   col = rainbow(10),           # Set bars to rainbow color
                   horiz = TRUE,                # Set barplot to display horizontally
                   las = 1,                     # Set labels shows horizontally
                   cex.names = 0.6)             # Adjust the size of y-axis label

text(x = Q2_data$mean,                  # Use text function <- plot mean of loss from "Q2_data" as x-axis
     y = barplot,                       # Plot y-axis as barplot
     labels = round(Q2_data$mean, 4),   # Round up the decimal place of mean to 4 decimal places
     pos = 4,                           # Display average loss on the right of the bar
     offset = 0.5,                      # Distance between bar and the text
     cex = 0.8)                         # Slightly reduce the size of text
```

Figure 55

Barplot of the Top 10 Web Servers with the Highest Average Loss



Research Question 3:

What is the relationship between OS family and web server?

Figure 56

Code Snippet to Retrieve OS Family, WebServer and Loss Attributes

```
## Create "OS", "WebServer" and "Loss" dataset
Q3_data <- member_2_dataset %>%                          # Create "Q3_data" data frame
  filter(!is.na(OS_Family)) %>%                          # Include "OS_Family" that have values only
  mutate(OS_Family = droplevels(OS_Family)) %>%          # Removes all unused levels in OS_Family
  filter(!is.na(WebServer)) %>%                          # Filter missing values out of "WebServer"
  mutate(WebServer = droplevels(WebServer)) %>%          # Removes all unused levels in WebServer
  select(os = OS_Family, ws = WebServer, loss = Loss)    # Only display specific attributes
```

This block of code has created a dataset called “Q3_data” that contains “OS_Family”, “WebServer” and “Loss” only. Before I further my analysis, I have dropped all missing value in “OS_Family” and “WebServer” then redefine their name for further calling.

Figure 57

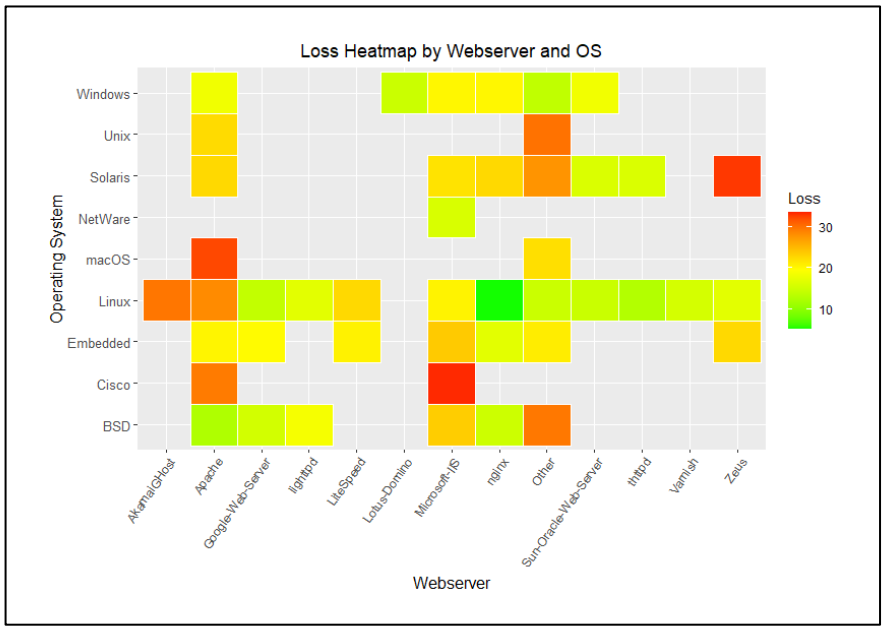
Code Snippet to Generate a Heatmap to Display OS Family, Web Server and Loss

```
# Heatmap
ggplot(Q3_data, aes(x = ws, y = os, fill = loss)) +
  geom_tile(color = "white", linewidth = 0.5) +     # Create a heatmap <- Add white borders between loss
  scale_fill_gradient2(low = "green",
                       mid = "yellow",
                       high = "red",
                       midpoint = median(Q3_data$loss)) +   # fill loss with different colors
  labs(title = "Loss Heatmap by Webserver and OS",          # Title of the heatmap
       x = "Webserver",                                     # Label of x-axis
       y = "Operating System",                              # Label of y-axis
       fill = "Loss") +                                     # Values within the heatmap
  theme(
    axis.text.x = element_text(angle = 55, hjust = 1),      # Rotate x-axis labels to readability angle
    axis.text.y = element_text(size = 10),                  # Adjust y-axis text size
    axis.title = element_text(size = 12),                   # Adjust titles text size
    plot.title = element_text(hjust = 0.5)                  # Center the plot title
  )
```

Next on, I have created a heatmap to visualize two independent variables (“OS” and “WebServer”) and one dependent variable (“Loss”). Based on this code, we can view the “Loss” for different combinations of “OS_Family” and “WebServer”. The colour refers to the number of losses and the detail of colour is shown on the right of the heatmap, we can know the loss is more than thirty when the loss is showing red, the loss will show green when it is less than 10. The least loss for combination of “OS_Famliy” and “WebServer” that is shown in this heatmap is “Linux” as “OS_Family” and “nginx” as WebServer.

Figure 58

Heatmap of Loss Across Different OS and WebServer



Objective 3:

I aim to study how the loss incurred due to web defacement attacks evolves across different timeframes based on website infrastructure characteristics derived from URLs and IP addresses.

Prepared by: Lim Beng Rhui (TP068495)

Further Data Preprocessing

Before conducting exploratory data analysis (EDA), additional preprocessing steps are performed towards the dataset retrieved using the `quick_preprocessed_dataset()` method. Additional attributes, including Protocol_Type, Sector, URL_Linked_File, and IP_Class, are extracted to provide more variations for dataset visualization.

Figure 59

Code Snippet for Extracting Protocol Type using Regex

```
member_3_dataset <- quick_preprocessed_dataset()      # Retrieve dataset

member_3_dataset <- member_3_dataset %>%              # Retrieve the involved attributes
  select(Date, URL, IP, Loss)

member_3_dataset <- member_3_dataset %>%              # New Attribute: Protocol Type (HTTP and HTTPS)
  mutate(Protocol_Type = if_else(str_detect(URL, "(?i)^https://"),
                                 "HTTPS",
                                 if_else(str_detect(URL, "(?i)^http://"),
                                         "HTTP",
                                         NA_character_)))

sum(is.na(member_3_dataset$Protocol_Type))            # Check if there is any NA invoked
filter(member_3_dataset, is.na(Protocol_Type))        # View the empty records
```

Figure 60

Code Snippet for Extracting the Sector Attribute

```
member_3_dataset <- member_3_dataset %>%
  mutate(                                                 # First remove the http and https from the URL
    Sector = case_when(str_detect(URL, "^(?i)https?://\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}") ~ NA_character_,
                       str_detect(URL, "(?i)https://") ~ str_remove(URL, "(?i)https://"),
                       str_detect(URL, "(?i)http://") ~ str_remove(URL, "(?i)http://"),
                       TRUE ~ NA_character_)) %>%

  mutate(Sector = sub("^([^/]+).*", "\\1", Sector)) %>%   # Then remove the words after the slash ("/") character, if available

  mutate(Sector = case_when(str_detect(Sector, "(?i)\\.(com|co|mobi|biz|shop|store|enterprise|ltd|inc)") ~ "Commercial",
                            str_detect(Sector, "(?i)\\.(edu|ac|k12|sch|lib)") ~ "Education",
                            str_detect(Sector, "(?i)\\.(net|wanadoo)") ~ "Network",
                            str_detect(Sector, "(?i)\\.org") ~ "Organization",
                            str_detect(Sector, "(?i)\\.(gov|gob|go|govt)") ~ "Government",
                            str_detect(Sector, "(?i)\\.mil") ~ "Military",
                            str_detect(Sector, "(?i)\\.info") ~ "Information",
                            str_detect(Sector, "(?i)\\.name") ~ "Individual",
                            str_detect(Sector, "(?i)(tv|media|fm|radio)") ~ "Media",
                            str_detect(Sector, "(?i)(museum|art|gallery)") ~ "Cultural",
                            str_detect(Sector, "(?i)(blog|wiki|press)") ~ "Publishing",
                            str_detect(Sector, "(?i)(tech|dev|app)") ~ "Technology",
                            TRUE ~ NA_character_))

count(member_3_dataset, Sector) # Retrieve the number of data for each sector
```

Figure 61

Code Snippet for Extracting IP Classes

```
member_3_dataset <- member_3_dataset %>%
  mutate(first_octet = as.numeric(sub("\\..*", "", IP)),         # Retrieve the first octet
         IP_Class = case_when(is.na(IP) ~ NA_character_,         # Calculate the IP class accordingly
                              first_octet < 128 ~ "Class A",
                              first_octet < 192 ~ "Class B",
                              first_octet < 224 ~ "Class C",
                              first_octet < 240 ~ "Class D",
                              TRUE ~ "Class E")) %>%
  select(-first_octet)                                           # Remove the column that stores the first octet
```

Figure 62

Code Snippet for Extracting the URL File Type Attribute

```
member_3_dataset <- member_3_dataset %>%

  # We first remove the http and https
  mutate(URL_File_Type = case_when(str_detect(URL, "(?i)https://") ~ str_remove(URL, "(?i)https://"),
                                   str_detect(URL, "(?i)http://") ~ str_remove(URL, "(?i)http://"),
                                   TRUE ~ URL)) %>%

  # Then we remove the characters in front of the last slash (/) and the text after the question mark (?)
  mutate(URL_File_Type = case_when(str_detect(URL_File_Type, "/") ~ sub(".*/(.*?)(\\?.*)?$", "\\1", URL_File_Type),
                                   TRUE ~ NA_character_)) %>%     # For URLs without a path, return empty string

  # Lastly, we remove the name of the file
  mutate(URL_File_Type = case_when(str_detect(URL_File_Type, "\\.") ~ str_extract(URL_File_Type, "[^.]+$"),
                                   TRUE ~ NA_character_)) %>%

  # Exist some repetitive names, try to modify them by first converting to lower case
  mutate(URL_File_Type = sapply(URL_File_Type, tolower)) %>%

  # Then categorize them based on inspection
  mutate(URL_File_Type = case_when(str_detect(URL_File_Type, "htm|thml|html|php|phtml|asp|jsp|css|js") ~ "Website",
                                   str_detect(URL_File_Type , "jpg|jpeg|gif|png|mp4") ~ "Media",
                                   str_detect(URL_File_Type, "txt|pdf|xml|xsl|rdf") ~ "Document",
                                   str_detect(URL_File_Type, "cgi|pl|cfm") ~ "Script",
                                   str_detect(URL_File_Type, "sql|db|dat") ~ "Database",
                                   str_detect(URL_File_Type, "bak|back|old|tmp|sav|log|ds_store|index|bk|prev|bad|ini|inc") ~ "Log",
                                   TRUE ~ NA_character_))
```

Figure 63

Dataset after Adding Protocol Type, Sector, URL Linked File and IP Class

	Date	Year	Month	Day	URL	Protocol_Type	Sector	URL_File_Type	IP	IP_Class	Loss
1	2002-03-15	2002	Mar	15	http://www.iguatemi.com.br/default.htm	HTTP	Commercial	Website	200.215.187.181	Class C	17.867911
2	2002-03-19	2002	Mar	19	http://www.iguatemi.com.br/main.htm	HTTP	Commercial	Website	200.215.187.181	Class C	18.065699
3	2002-03-25	2002	Mar	25	http://pa83.jastrzebie.sdi.tpnet.pl/ie.htm	HTTP	<NA>	Website	217.96.207.83	Class C	33.481995
4	2002-04-08	2002	Apr	8	http://www.lechefsa.com.br/index.htm	HTTP	Commercial	Website	200.179.148.2	Class C	21.190124
5	2002-04-10	2002	Apr	10	http://www.fazendajuisa.com.br/index0.htm	HTTP	Commercial	Website	200.198.179.251	Class C	19.418150
6	2002-04-10	2002	Apr	10	http://www.programadomala.com.br/index0.htm	HTTP	Commercial	Website	200.198.179.251	Class C	15.134127
7	2002-04-15	2002	Apr	15	http://ut.gamer.hr/index.php	HTTP	<NA>	Website	212.91.96.132	Class C	23.865978
8	2002-04-15	2002	Apr	15	http://www.zvans.llu.lv/index.php	HTTP	<NA>	Website	159.148.237.5	Class B	20.382003
9	2002-04-16	2002	Apr	16	http://www.philscan.com/index.html	HTTP	Commercial	Website	64.49.222.172	Class A	11.507561
10	2002-04-16	2002	Apr	16	http://www.cc-jensen.com/index.php	HTTP	Commercial	Website	64.39.17.229	Class A	17.561152

Research Question 1:

What is the correlation between temporal factors (year, month, and day), website information (protocol type, sector, file type, IP class), and web defacement loss?

Correlation analysis is often used to evaluate if there exists any direct relationship between two variables (James, 2019). Hence, to gain an initial insight into how different variables affect web defacement loss, correlation analysis is performed towards the dataset by implementing a correlation matrix using R, as shown in Figure 64, to obtain the correlation coefficient for each pair of variables and identify if there is any individual factor that strongly influences the amount of loss.

Figure 64

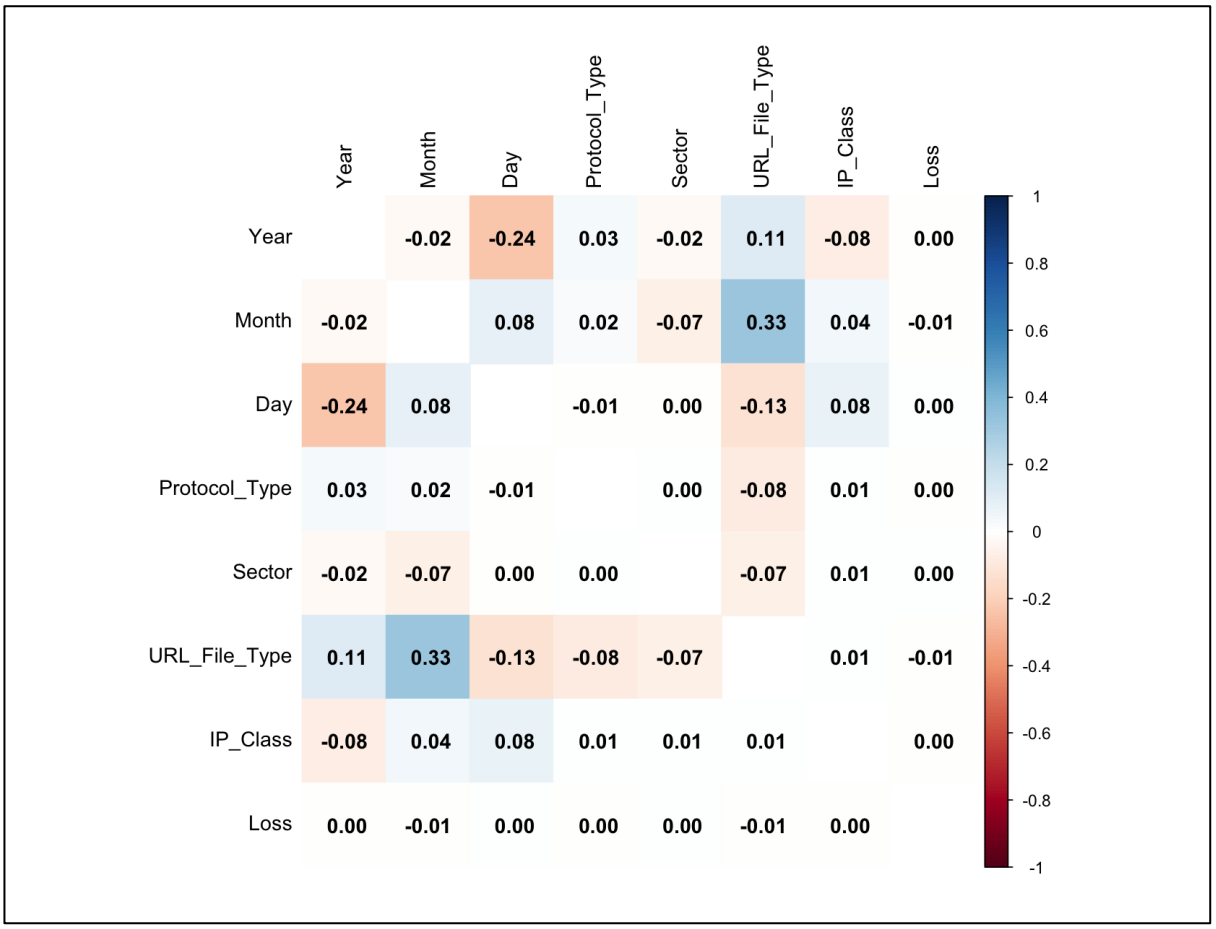
R Code for Generating Correlation Matrix for Temporal Attributes, Website

```
# Convert the variables into numeric type and ignore records with NA
member_3_correlation_data <- member_3_dataset %>%
  select(Year, Month, Day, Protocol_Type, Sector, URL_File_Type, IP_Class, Loss) %>%
  mutate(
    Year = as.numeric(Year),
    Month = match(Month, month.abb),
    Day = as.numeric(Day),
    Loss = as.numeric(Loss),
    Sector = as.numeric(Sector),
    URL_File_Type = as.numeric(URL_File_Type),
    IP_Class = as.numeric(IP_Class),
    Protocol_Type = as.numeric(Protocol_Type)
  ) %>%
  na.omit()

# Calculate correlation matrix
member_3_cor_matrix <- cor(member_3_correlation_data)

# Visualize correlation matrix
corrplot(member_3_cor_matrix,
         method = "color",           # Fill the entire box
         diag = FALSE,               # Remove diagonal (not useful for analysis)
         addCoef.col = "black",      # Add coefficients to each box except diagonal
         tl.col = "black")           # Change label to black colour instead of red
```

Based on the correlation matrix in Figure 65, it is inferred that there is no strong relationship between the loss attribute and any individual variables, as the correlation coefficients for the loss attribute are very close to 0. Interestingly, there is a moderately positive correlation between the URL_File_Type and the Month attributes. This suggests that the two variables potentially have a relationship that can be further investigated in exploratory data analysis later, along with other variables to identify any potential patterns that can depict the variables' relationship with loss.

Figure 65

Correlation Matrix for Temporal Attributes, Website Information Attributes and Loss



Research Question 2:

How does the average web defacement losses vary for different protocol types across different years?

Time series analysis, which analyses the changes of data over consistent intervals of time, is implemented to interpret the change of the average loss value for different protocol types (HTTP and HTTPS) across year 1998 till 2015 (Pandian, 2025). With the R code in Figure 66 that demonstrates the analysis, not only we can gain an understanding towards the pattern between protocol types and losses but also having the ability to forecast future trends and perform predictive analytics if needed.

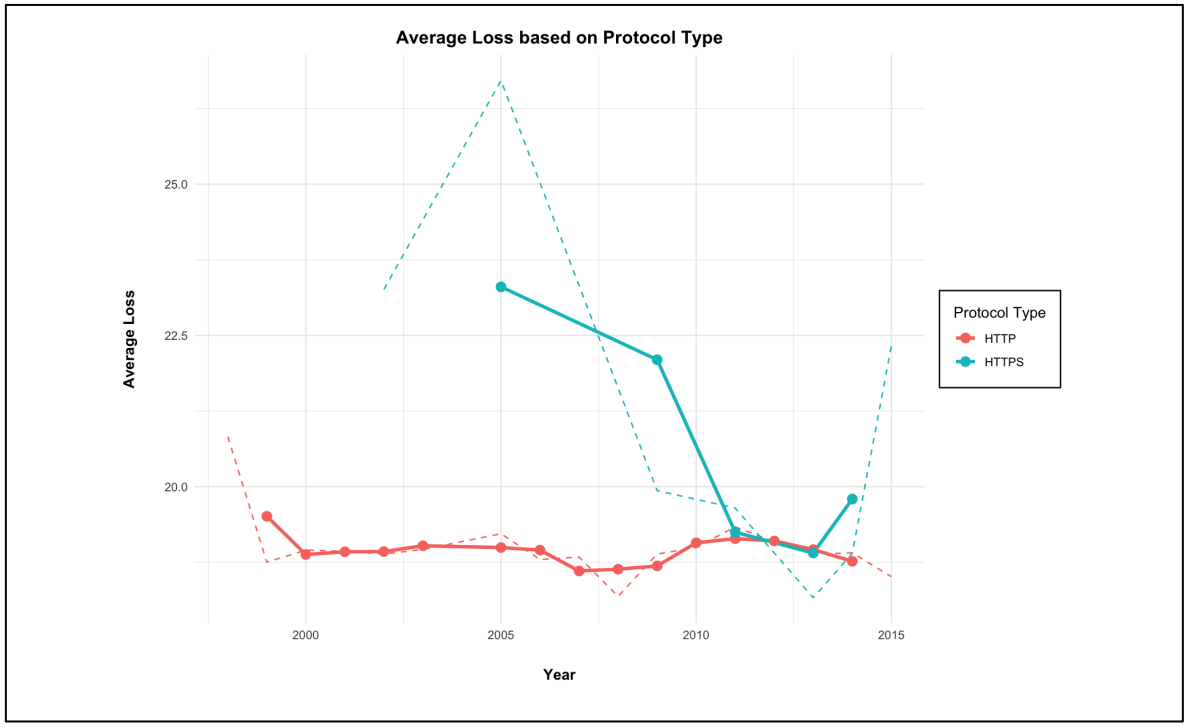
Figure 66

R Code for Generating Average Yearly Loss based on Protocol Type

```r
# Calculate the aggregated yearly loss for each protocol type
member_3_average_yearly_loss_based_on_protocol <- member_3_dataset %>%
  filter(!is.na(Protocol_Type)) %>%                                    # Remove NA values
  group_by(Year, Protocol_Type) %>%                                    # Group data by year and protocol type
  summarise(mean_loss = mean(Loss, na.rm = TRUE), .groups = "drop") %>%  # Calculate mean and remove the grouping
  group_by(Protocol_Type) %>%                                          # Regroup data for smoothing purpose
  arrange(Year) %>%                                                    # Arrange the data by year
  mutate(                                                              # Perform smoothing to better display trend
    smoothed_loss = rollmean(mean_loss, k = 3, fill = NA, align = "center")
  ) %>%
  ungroup()                                                            # Ungroup to avoid graph not working

# Generate plot with the data
ggplot(member_3_average_yearly_loss_based_on_protocol, aes(x = Year)) +

  geom_line(aes(y = mean_loss, color = Protocol_Type), linetype = "dashed") +          # Dashed line (before smoothing)
  geom_line(aes(y = smoothed_loss, color = Protocol_Type), size = 1.2, na.rm = TRUE) +  # Solid line (smoothed)
  geom_point(aes(y = smoothed_loss, color = Protocol_Type), size = 3, na.rm = TRUE) +   # Data points (smoothed)

  labs(
    title = "Average Loss based on Protocol Type",    # Title
    x = "Year",                                       # x-axis label
    y = "Average Loss",                               # y-axis label
    color = "Protocol Type"                           # Legend
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),                  # Center title
    axis.title.x = element_text(margin = margin(t = 20), face = "bold"),    # Add space for x-axis
    axis.title.y = element_text(margin = margin(r = 20), face = "bold"),    # Add space for y-axis
    legend.box.background = element_rect(color = "black", size = 0.5),      # Box the legend
    legend.box.margin = margin(t = 5, r = 5, b = 5, l = 5)                  # Add margin to legend box
  )
```

Figure 67

Line Graph Representing the Average Loss based on Protocol Type Across Years



Looking at the red line representing the HTTP protocol type, its average loss remains stable across different years, except in the early 2000's where it experienced a slight decline that might be due to the birth of HTTPS in 1999, where websites are slowly switching to HTTPS due to the enforced network security, eventually leading to a reduction in loss incurred in HTTP websites as HTTPS became the new focus for hackers (Smitter, 2023). On the other hand, the HTTPS protocol type represented by the blue line shows a significantly high average loss in the year 2005, which might occur due to the vulnerabilities for the new HTTPS technology by that time. However, the average loss drops drastically with HTTPS becoming more matured, till to the point where it has a lower average loss than HTTP around year 2013. Things changed when there is a sudden spike of HTTPS in year 2014, where it might be caused by the Heartbleed bug that allows intruders to read the memory of servers, impacting websites and lead to higher loss (Lee, 2015).

Research Question 3:

What is the relative contribution of each URL sector to the total web defacement loss across 5-year periods?

Distribution analysis is a way to help data analysts to gain an idea on how the data is located and arranged across different values on a graph (Devansh, 2023). As shown in Figure 69, using a stacked bar chart allows us to display the proportional distribution of loss contributed by each URL sector across a 5-year interval so that we will be able to observe any increase or decrease trends associated with the data.

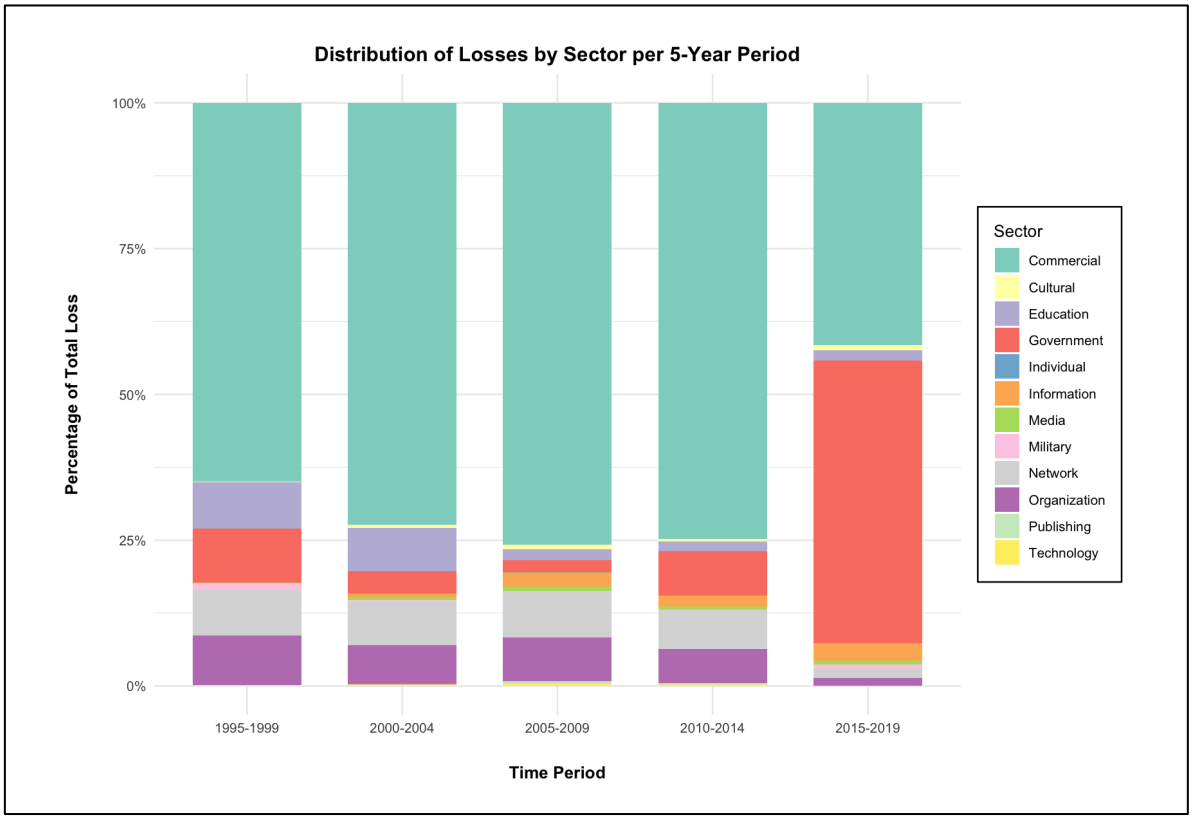
Figure 68

R Code to Generate Stacked Bar Chart for Proportion of Loss across URL Sectors

```
member_3_aggregated_5_year_loss_by_sector <- member_3_dataset %>%
  filter(!is.na(Sector)) %>%                                          # Remove unlabelled sectors
  mutate(Period = paste0(floor(Year/5)*5, "-", floor(Year/5)*5 + 4)) %>%     # Separate period into 5 years
  group_by(Period, Sector) %>%                                        # Group data
  summarise(Total_Loss = sum(Loss, na.rm = TRUE), .groups = "drop") %>%      # Calculate loss of combination
  arrange(Period)                                                     # Arrange the period ascendingly

ggplot(member_3_aggregated_5_year_loss_by_sector, aes(x = Period, y = Total_Loss, fill = Sector)) +
  geom_bar(position = "fill", stat = "identity", width = 0.7) +       # Create stacked bar plots
  scale_y_continuous(labels = scales::percent) +                      # Convert into percentage
  scale_fill_brewer(palette = "Set3") +                               # Set colour
  labs(title = "Distribution of Losses by Sector per 5-Year Period",  # Title
       x = "Time Period",                                             # x-axis label
       y = "Percentage of Total Loss",                                # y-axis label
       fill = "Sector") +                                             # Legend
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),        # Center title
        axis.title.x = element_text(margin = margin(t = 20), face = "bold"), # Add space for x-axis
        axis.title.y = element_text(margin = margin(r = 20), face = "bold"), # Add space for y-axis
        legend.box.background = element_rect(color = "black", size = 0.5),   # Box the legend
        legend.box.margin = margin(t = 5, r = 5, b = 5, l = 5))       # Add margin to legend box
```

Figure 69

Stacked Bar Chart displaying Distribution of Percentage of Total Loss by Sector using 5-Year Period



As shown in Figure 69, it is observed that the **commercial** sector (represented as .com in URL) occupies the most proportion of loss from 1995 till 2014, indicating that hackers target businesses due to their lucrative revenues. However, after 2015, there is a significant decrease in the proportion of commercial sector, replaced by the **government** sector (represented as .gov in URL) that occupies roughly half of the proportion despite its small proportion in previous years. As the digital government concept was introduced in 2014 (OECD, 2014), the transition of government services towards digitalized platforms had increased the chance for intruders to launch cybersecurity attacks, like the WannaCry ransomware attack in 2017 that involved users from more than 150 countries (Bergal, 2017). Other sectors like **education** (represented by .edu in URL) and **organization** (represented by .org in URL) on the other hand experience a percentage decrease due to the increasing security awareness along with the shift of focus by cyberattackers towards corporate and government-related websites (Cyber Threat Alliance, 2024).

Research Question 4:

What monthly trends exist in the loss distribution among different IP classes?

Trend analysis, aiming to identify the recurring trends or behaviours that take place over a specific period, is also applied to analyse repetitive patterns that occurs among different IP classes for each month (Infomineo, 2024). To perform trend analysis, the loss is summed up based on the month and IP class of the record, then passed into the `ggplot()` function to generate a heatmap.

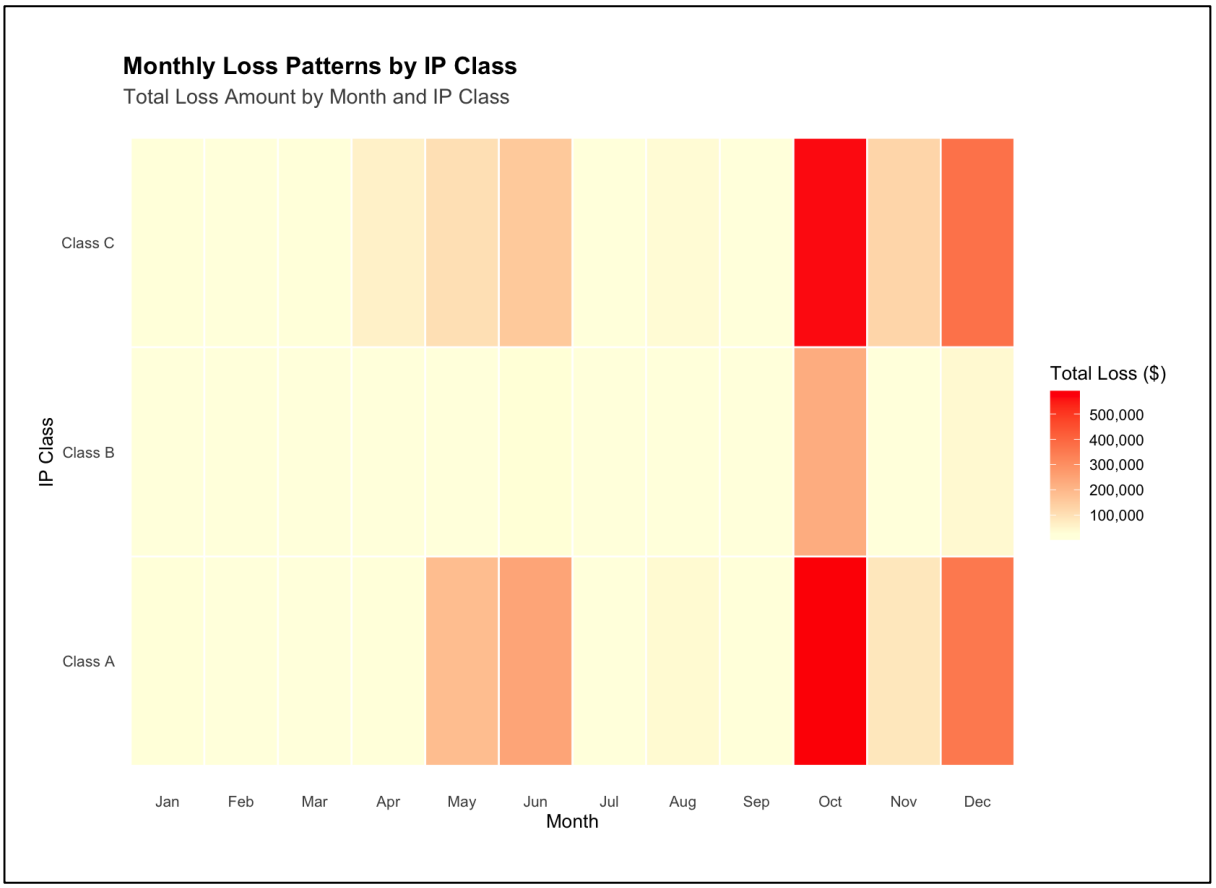
Figure 70

R Code to Generate Monthly Loss Patterns by IP Class

```
# Calculate median for midpoint before plotting
median_loss <- member_3_dataset %>%
  filter(!is.na(IP_Class), IP_Class != "Class E") %>%
  group_by(Month, IP_Class) %>%
  summarize(total_loss = sum(Loss)) %>%
  pull(total_loss) %>%
  median()

# Create heatmap
member_3_dataset %>%
  filter(!is.na(IP_Class), IP_Class != "Class E") %>%
  group_by(Month, IP_Class) %>%
  summarize(total_loss = sum(Loss), .groups = "drop") %>%
  mutate(
    Month = factor(month.abb[Month],
                   levels = month.abb),
    loss_formatted = scales::comma(total_loss)
  ) %>%
  ggplot(aes(x = Month, y = IP_Class, fill = total_loss)) +
  geom_tile(color = "white", linewidth = 0.5) +
  scale_fill_gradient2(
    low = "white",
    mid = "lightyellow",
    high = "red",
    midpoint = median_loss,
    labels = scales::comma
  ) +
  labs(
    title = "Monthly Loss Patterns by IP Class",
    subtitle = "Total Loss Amount by Month and IP Class",
    x = "Month",
    y = "IP Class",
    fill = "Total Loss ($)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 12, color = "gray30"),
    axis.text.x = element_text(angle = 0),
    panel.grid = element_blank(),
    legend.position = "right",
    plot.margin = margin(t = 20, r = 20, b = 20, l = 20)
  )
```

Figure 71

Heatmap Representing the Sum of Amount of Loss in IP Class



As shown by the heatmap in Figure 71, the total loss for a few months, particularly for October, is significantly higher than the total loss in the remaining months, as represented by the red blocks. As October marks the start of holiday seasons like Black Friday, Thanksgiving and Cyber Monday, the increased online activity by users potentially leads to the increase opportunity for hackers to launch their attacks as the high volume of financial transactions allows hackers to maximize their impacts and exploit available vulnerabilities to raid profits (Pereira, 2021). However, the situation is alleviated for servers located in the Class B segment, as we can observe with the slightly lighter colour in the heatmap. Since class B is usually catered for organizations like educational institutions and medium-sized corporate networks, due to the lower public exposure if compared to class A (usually used in large-scaled institutions) and class C (usually used in home networks), this makes class B not the focus of hackers for their attacks (NRS, 2024).

Objective 4:

I have in view on how year and country affect downtime due to web defacement attacks.

Prepared by: Ng Xuan Jack (TP067678)

Research Question 1:

What patterns or trends can be observed via average loss caused by web defacement attacks over the years?

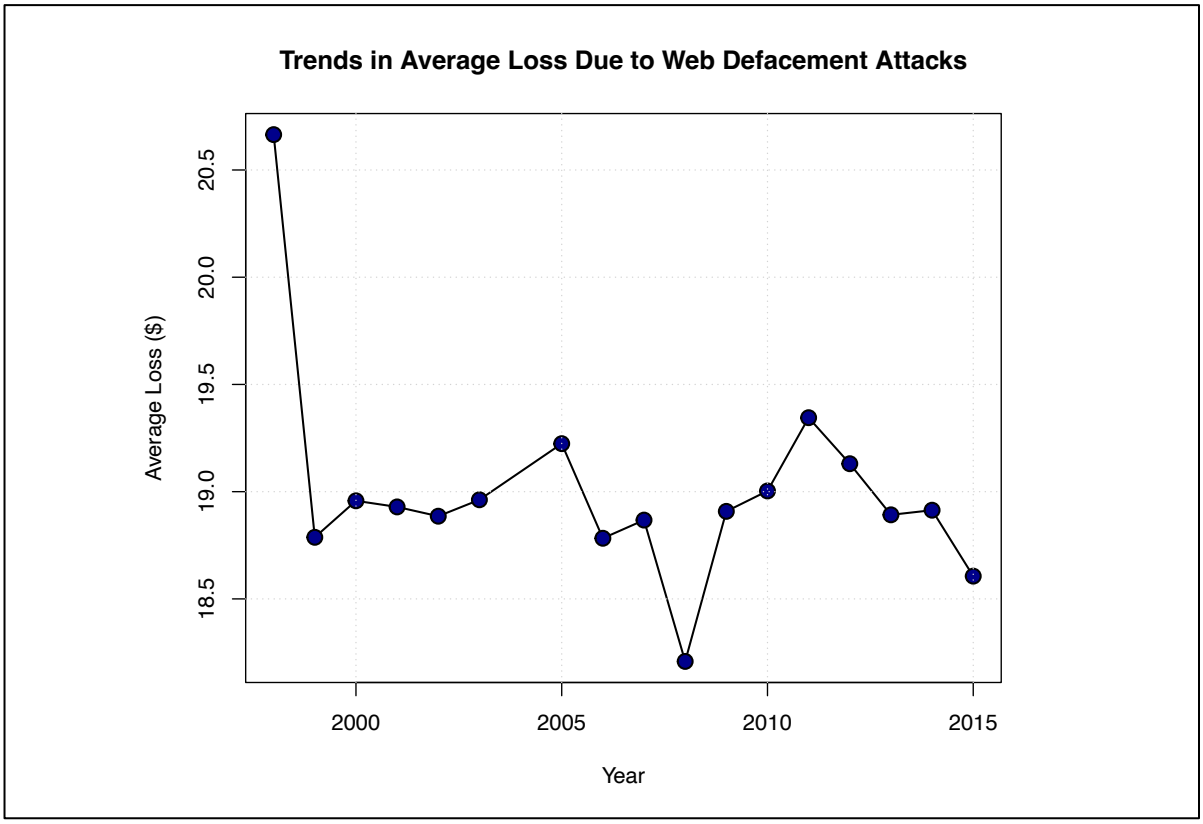
Figure 72

Code Snippet to Generate Line Chart of Average Loss across Different Years

```
data <- quick_preprocessed_dataset()                          # Retrieve dataset

yearly_trends = data %>%
    group_by(Year) %>%                                        # Group data based on Year
    summarise(Average_Loss = mean(Loss, na.rm = TRUE))        # Calculates average loss & ignore missing values

yearly_trends$Year = as.numeric(yearly_trends$Year)           # Converts Year into numerical-based column for chronological order

plot(yearly_trends$Year, yearly_trends$Average_Loss, type = "o",         # Plotting the line plot
     pch = 21, bg = "darkblue", cex = 1.5, lwd = 1.5,
     xlab = "Year", ylab = "Average Loss ($)",
     main = "Trends in Average Loss Due to Web Defacement Attacks")
grid()
```

Figure 73

Line Chart of Average Loss across Different Years



Line plot was used due to its effectiveness to visualizes web defacement attacks has fluctuations while being ideal in identifying time-series data because it clearly shows how values change over time, while it can also reveal periods when average ***Loss*** spiked, timeframe of increased cybercriminal activity, major and minor hacking events occurred.

The line plot analysed the trend of average ***Loss*** due to web defacement attacks from the late 1990s to 2015. Upon inspection, there was a sharp decline in ***Loss*** after ***Year*** 2000, likely due to cybersecurity enhancements. From ***Year*** 2000 to 2010, average ***Loss*** fluctuated with occasional peaks and dips, with sudden downpour burst out around ***Year*** 2008 due to absence of ***Year*** 2008's data in our csv file. A noticeable peak occurred around ***Year*** 2011, showing a surge in high-impact attacks. However, the trend quickly shows a gradual decline after that, possibly indicating better security measures and mitigation actions carried out.

Overall, the pattern suggests that while web defacement attacks remained a persistent issue, there were always web-security counterattack being developed and implemented to the market to hold the line and maintain security.

Research Question 2:
What were the differences existing in average loss caused by web defacement attacks across various countries?

The bar plot in Figure 75 illustrates top 20 countries of average ***Loss*** with Benin as the highest, followed by Netherlands Antilles and Azerbaijan. While these countries experience notable economic impacts, the overall losses remain relatively lower in comparison to global cybersecurity threats, as shown by the yellow bar. While the analysis allows us to identify regions with heightened exposure to web defacement risks, the information extracted is unclear and insufficient, hence a world map is included for visual enhancements.

The world map in Figure 77 illustrates global distribution of average ***Loss*** due to web defacement across countries. The color gradient ranged from yellow (low loss) to dark red (high loss). From the visualization, Africa, South America, parts of Europe and Asia show higher average ***Loss*** due to web defacement attacks, suggesting these regions might be more frequent targets of cyberattacks. Concurrently, some regions in Africa and smaller nations have lighter shades, potentially due to missing data. The visualization helps in understanding regional cybersecurity vulnerabilities and assist policymakers and organizations in prioritizing cybersecurity measures in high-risk areas.

Figure 74

Code Snippet to Generate Bar Chart representing Average Loss of Top 20 Countries

```
library(rworldmap)
library(RColorBrewer)
library(ggplot2)
library(dplyr)

# Aggregate average Loss per country
avg_loss = data %>%
  group_by(Country) %>%
  summarise(Average_Loss = mean(Loss, na.rm = TRUE))

# List the top 20 countries by average Loss
top_20_countries = avg_loss %>%
  arrange(desc(Average_Loss)) %>%
  head(20)

# Generate bar plot, list x, y and title, fill in color
ggplot(top_20_countries, aes(x = reorder(Country, Average_Loss), y = Average_Loss, fill = Average_Loss)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(title = "Top 20 Countries by Average Loss Due to Web Defacement",
       x = "Country", y = "Average Loss", fill = "Avg_Loss") + theme_minimal()
```

Figure 75

Bar Chart Representing the Average Loss of the Top 20 Countries

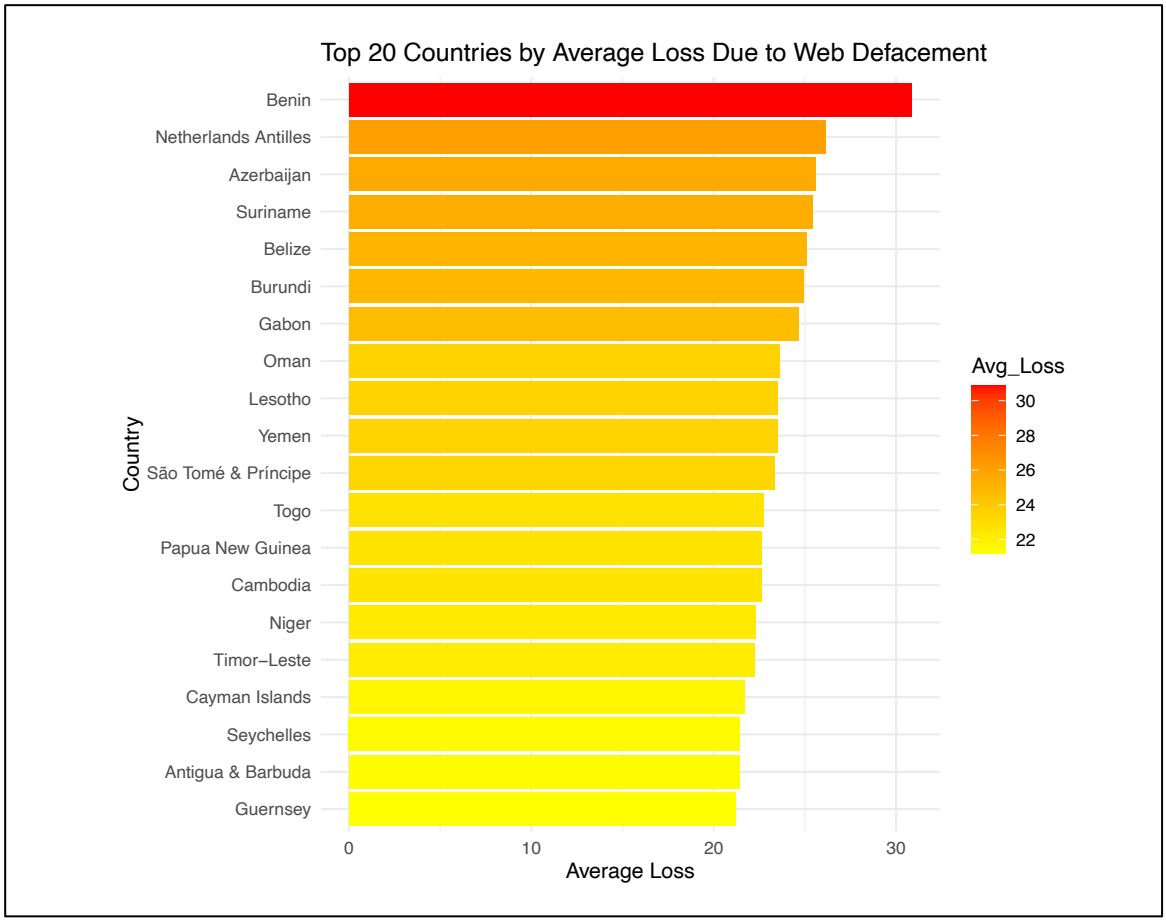


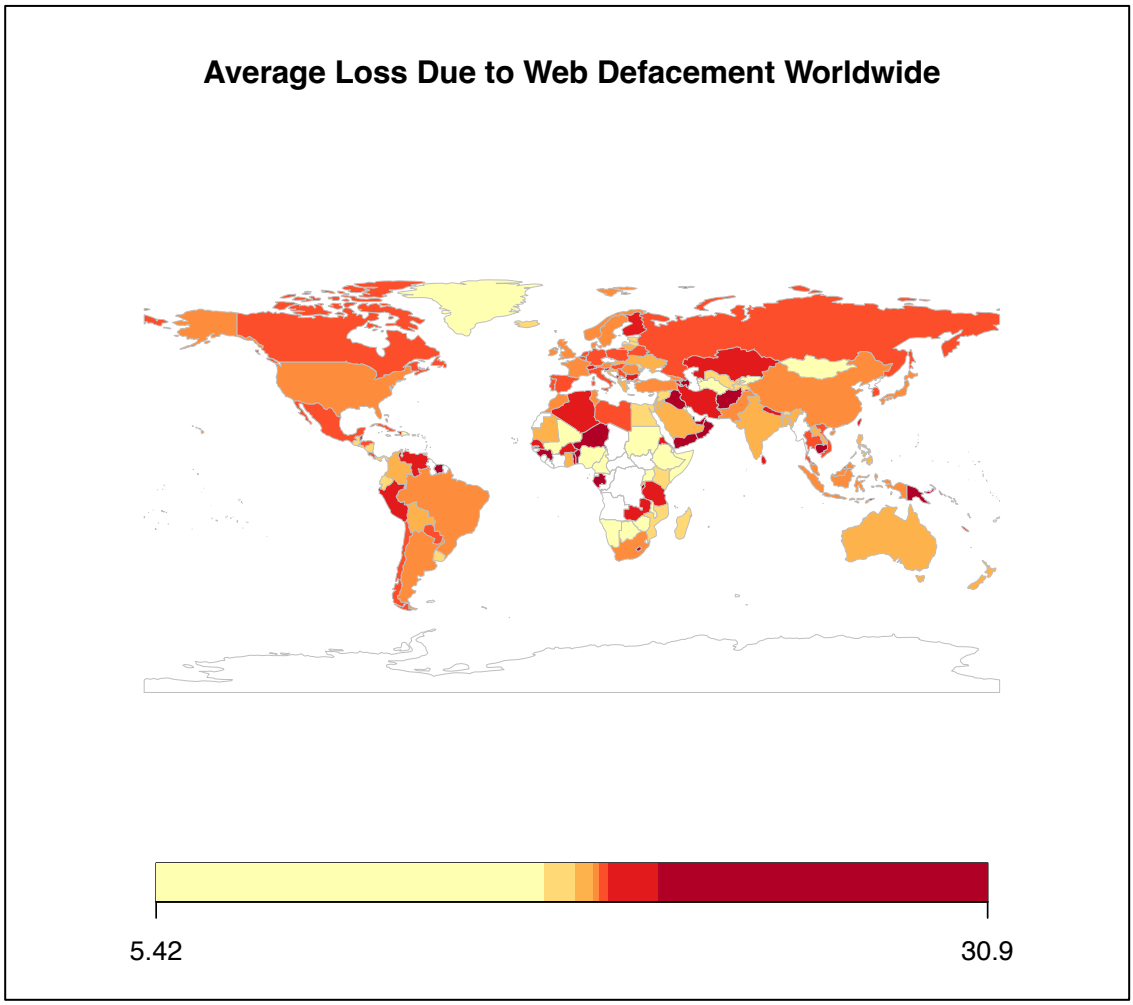
Figure 76

Code Snippet to Generate Map for Worldwide Average Loss

```
# Group countrydata as worldMap
worldMap = joinCountryData2Map(avg_loss, joinCode = "NAME", nameJoinColumn = "Country")

# Assign color to regions (Yellow - Orange - Red)
num_colors = 7
breaks = classIntervals(worldMap$Average_Loss, n = num_colors, style = "quantile")$brks
color_palette = brewer.pal(num_colors, "YlOrRd")

# Plotting world map
mapParams = mapCountryData(worldMap,
                           nameColumnToPlot = "Average_Loss",
                           mapTitle = "Average Loss Due to Web Defacement Worldwide",
                           catMethod = breaks,
                           colourPalette = color_palette, addLegend = TRUE)
```

Figure 77

Map Representing Worldwide Average Loss



Research Question 3:

Does relationship exist or associated across year, country, and the average loss due to web defacement attacks?

Figure 78

Code Snippet to Generate Heatmap for Average Loss of Countries

```
# Aggregate the average Loss of Countries per Year
yearly_country_trends = data %>%
  group_by(Year, Country) %>%
  summarise(Average_Loss = mean(Loss, na.rm = TRUE)) %>%
  ungroup()

# Calculate the top 10 Countries with the highest average Loss
top_10_countries = yearly_country_trends %>%
  group_by(Country) %>%
  summarise(Total_Avg_Loss = mean(Average_Loss, na.rm = TRUE)) %>%
  top_n(10, Total_Avg_Loss) %>%
  pull(Country)

# Limit to only include top 10 Countries
filtered_data = yearly_country_trends %>%
  filter(Country %in% top_10_countries)

# Convert Year to numerical
filtered_data$Year = as.numeric(filtered_data$Year)

# Plotting the heatmap
ggplot(filtered_data, aes(x = Year, y = reorder(Country, -Average_Loss), fill =
Average_Loss)) +
  geom_tile() +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(title = "Heatmap of Countries' Average Loss by Year (Top 10)",
       x = "Year", y = "Country", fill = "Avg Loss ($)") +
  theme_minimal()
```

Aiming to highlight prominent ***Countries*** with higher frequent average ***Loss***, the heatmap uses red colour to display average ***Loss*** variations per year. This makes Benin not dominating the heatmap as it only experienced extreme ***Loss*** in specific ***Years***.

In a nutshell, there is an association between ***Year***, ***Country***, and average ***Loss***, but it appears to be episodic than a steady trend. The visualization identifies patterns and fluctuations in cyberattack severity over the ***Years***. Some ***Countries*** like Benin, experience sporadic yet intense ***Loss***, indicating targeted cyberattacks in specific ***Years*** rather than a consistent threat. Additionally, a stacked bar plot is generated to further address the issue.

Figure 79

Heatmap Representing the Top 10 Countries with the Highest Average Loss

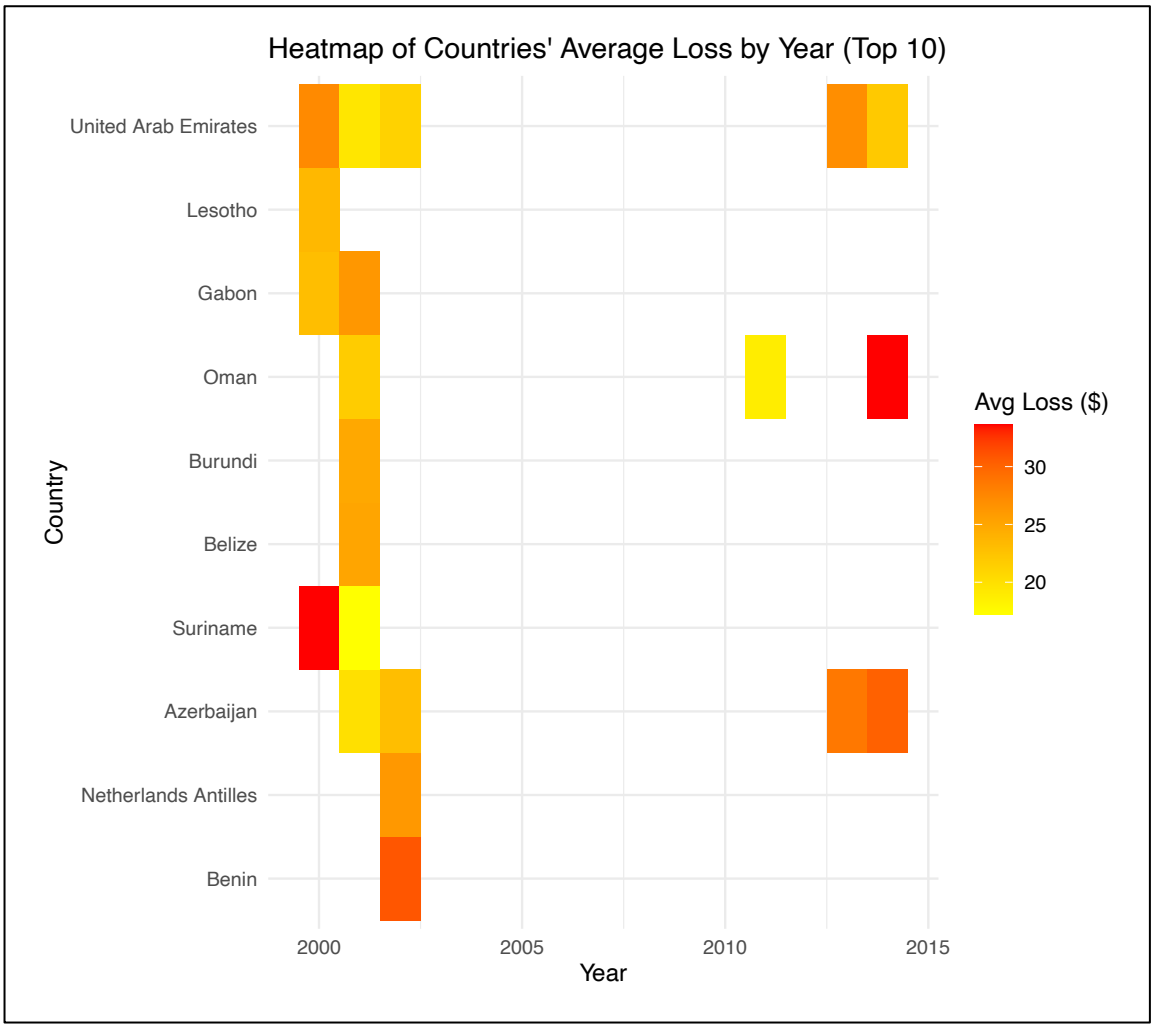


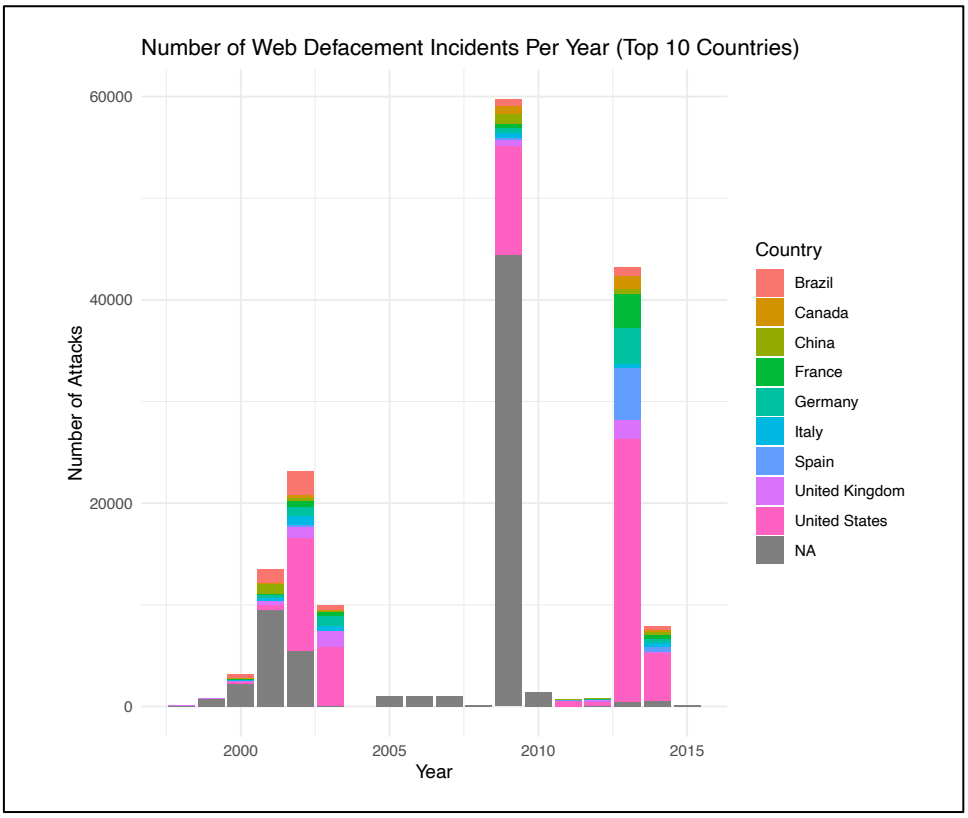
Figure 80

Code Snippet to Generate Heatmap for Average Loss of Countries

```
# Reuse the data compiled in heatmap, group as countries, analyse the top 10
top_10_countries = data %>%
  group_by(Country) %>%
  summarise(Total_Attacks = n()) %>%
  top_n(10, Total_Attacks) %>%
  pull(Country)

# Clean data and extract only the top 10
filtered_data = data %>% filter(Country %in% top_10_countries)

# Plotting the stacked bar plot
ggplot(filtered_data, aes(x = Year, fill = Country)) +
  geom_bar(position = "stack") +
  labs(title = "Number of Web Defacement Incidents Per Year (Top 10 Countries)",
       x = "Year", y = "Number of Attacks", fill = "Country") +
  theme_minimal()
```

Figure 81

Stacked Bar Chart displaying Number of Web Defacement Incidents Per Year



The stacked bar plot illustrates web defacement attacks peaking in ***Year*** 2002, 2009 and 2013, while the previous generated heatmap highlights economic losses in different ***Years*** and countries. The purpose of this plot is to emphasize that cyberattacks' frequency doesn't directly affects average financial ***Loss***; in other words, their relationship aren't correlated.

Countries like United States, United Kingdom, and China experience higher frequency of attacks but show low average financial ***Loss***, potentially indicating low-cost defacements or hinting that the countries have implemented advance cybersecurity defences. On contrary, ***Countries*** like Benin, Suriname and Netherlands Antilles appeared in previous heatmap but not in stacked bar plot, suggesting they experienced lethal losses despite the low web defacement attack frequency.

In a nutshell, cybercriminals operate in phases with diverse manners between regions. Based on the data and several pots, we could clearly see each regions' vulnerabilities, economic, cybersecurity defences level and attacking frequency, allowing professionals to better predict infiltration patterns, allocate resources effectively, and develop ideal defence mechanisms to better enhance cybersecurity.

Hypothesis Testing

After performing EDA, our team has outlined the key attributes that could potentially influence the Loss attribute. With careful consideration, we have selected WebServer, DownTime, and a derived attribute, OS_Family, to craft the advanced complex hypothesis:

Null hypothesis (H_0)

Websites running Apache web servers on a Linux operating system will experience the same amount of loss as websites with baseline configurations.

Alternative hypothesis (H_1)

Websites running Apache web servers on a Linux operating system will experience a reduction in loss by at least 10% if compared to baseline configuration, with the reduction effect amplifying as the downtime increases.

Our team aims to investigate whether using Apache web servers with a Linux-based operating system will impact the overall loss during web defacement activities. The null hypothesis assumes no change will occur towards loss via this configuration. In contrast, the alternative hypothesis suggests that there will be a minimum 10% reduction in the overall loss amount when Apache web servers and Linux are used.

To perform hypothesis testing, our team retrieved the dataset from the third member, which consisted of all additional attributes. After filtering the dataset with the four variables: OS_Family, WebServer, DownTime, and Loss, our team performed an ANCOVA (Analysis of Covariance) to examine the overall relationships between these variables. We then created a linear model with interaction effects to evaluate how the OS family and web server combination, with downtime, will impact the Loss attribute.

Figure 82

Summary of Dataset for Hypothesis Testing

```
 OS_Family           WebServer             DownTime            Loss
Length:211909      Length:211909      Min.   : 7.00      Min.   : 5.082
Class :character   Class :character   1st Qu.:20.00      1st Qu.:15.470
Mode  :character   Mode  :character   Median :32.00      Median :19.598
                                      Mean   :31.87      Mean   :18.924
                                      3rd Qu.:43.00      3rd Qu.:22.675
                                      Max.   :60.00      Max.   :33.482
```

Figure 83

Code Snippet for Hypothesis Testing

```
# Import dataset from member 3
hypothesis_dataset <- read.csv("member_3_dataset.csv")

# Select relevant data
hypothesis_dataset <- hypothesis_dataset %>%
  select(
    OS_Family,
    WebServer,
    DownTime,
    Loss
  )

# ------------------------------------------------------------
# ANCOVA TESTING
# ------------------------------------------------------------

# Modify the number of lines to be printed (to show all results)
options(max.print = 10000)

# Perform ANCOVA testing (for categorical variables and downtime)
ancova_model <- aov(Loss ~ DownTime * WebServer * OS_Family, data = hypothesis_dataset)
summary(ancova_model)

# ------------------------------------------------------------
# LINEAR MODEL FOR TESTING
# ------------------------------------------------------------

# Linear model combined with different variables
linear_model <- lm(Loss ~ DownTime * WebServer * OS_Family, data = hypothesis_dataset)
summary(linear_model)
```

Figure 84

Results of ANCOVA Testing

```
                                Df    Sum Sq   Mean Sq     F value   Pr(>F)
DownTime                         1   2586809   2586809   1.020e+05   <2e-16 ***
WebServer                       12       321        27   1.055e+00   0.3944
OS_Family                        5        48        10   3.780e-01   0.8642
DownTime:WebServer              11       203        18   7.260e-01   0.7141
DownTime:OS_Family               5       159        32   1.255e+00   0.2805
WebServer:OS_Family             22       831        38   1.489e+00   0.0655 .
DownTime:WebServer:OS_Family    18       554        31   1.214e+00   0.2388
Residuals                   149858   3800858        25
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

From the results of the ANCOVA, it is observed that DownTime significantly influences the Loss attribute, as indicated by its high F-value of 102,000 supported by a p-value below 0.05, indicating that DownTime can explain the variance in the data. On the other hand, other variables generally show minimal impact due to their low F-values and high p-values. The only exception is the interaction between WebServer and OS_Family ($p = 0.0655$), which indicates a potential effect if a more lenient significance level is chosen. Since only DownTime significantly impacts loss, a linear model is implemented to examine the effect of specific combinations of OS families and web servers on the Loss attribute.

Figure 85

Results of Linear Model with Interaction Effects

```
Call:
lm(formula = Loss ~ DownTime * WebServer * OS_Family, data = testing_complete)

Residuals:
     Min       1Q   Median       3Q      Max
-11.0530  -2.6341  -0.9677   0.8291  15.0811

Coefficients:
                                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                                        1.346e+02  4.928e+01   2.732  0.00630 **
DownTime                                          -2.894e+00  1.468e+00  -1.971  0.04868 *
WebServerApache                                   -1.242e+02  4.928e+01  -2.521  0.01172 *
OS_FamilyLinux                                    -1.137e+02  4.882e+01  -2.329  0.01988 *
DownTime:WebServerApache                           3.164e+00  1.468e+00   2.155  0.03116 *
DownTime:OS_FamilyLinux                            2.916e+00  1.457e+00   2.002  0.04525 *
WebServerApache:OS_FamilyLinux                     1.131e+02  4.882e+01   2.317  0.02049 *
DownTime:WebServerApache:OS_FamilyLinux           -2.902e+00  1.457e+00  -1.992  0.04636 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.036 on 149858 degrees of freedom
Multiple R-squared:  0.4052,         Adjusted R-squared:  0.4049
F-statistic:  1379 on 74 and 149858 DF,  p-value: < 2.2e-16
```

Looking at the filtered summary of the linear model for Apache web server and Linux, it is observed that each attribute's individual effect is statistically significant ($p < 0.05$), where:

- DownTime has a negative coefficient (-2.894),
- Apache web server has a negative coefficient (-124.2), and
- Linux operating system has a negative coefficient (-113.7).

This indicates that the Apache web server and the Linux operating system are independently associated with a loss reduction. While the DownTime attribute has a negative coefficient independently, its two-way interaction with the Apache web server and Linux OS has positive coefficients, indicating that the negative coefficient in the individual DownTime attribute is used to offset the positive coefficients in the interactions.

Looking at the three-way interaction, the result has a negative coefficient of -2.902, indicating that when comparing with baseline configurations, using the Apache web server and the Linux operating system can reduce losses incurred by 2.902 units for each unit increase in downtime. To translate these effects into percentages, we can plot a graph to show the overall percentage decrease for all available loss values associated with the combination of Apache web server and Linux operating system.

Figure 86

Derivation of Formula of Percentage Change after De-Transformation of Box-Cox

The box-cox transformation uses the formula: $f(x) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{for } \lambda \neq 0 \\ \log x & \text{for } \lambda = 0 \end{cases}$, where λ is automatically determined by the algorithm in the preprocessing stage. In our case, λ is found to be $0.2222 \approx \frac{2}{9}$.

We shall only consider the records with WebServer as Apache and OS_Family as Linux, since we would like to investigate the percentage change for these records.

Let the loss in the dataset (the reduced loss **after** box-cox transformation) be k.
The loss before reduction is then $k + 2.902$, as indicated by the linear model.

De-transforming the box-cox transformation will lead to the following calculation:

$$\text{Percentage Decrease} = \frac{\left[\frac{2}{9}(k+2.902)+1\right]^{\frac{9}{2}} - \left(\frac{2}{9}k+1\right)^{\frac{9}{2}}}{\left[\frac{2}{9}(k+2.902)+1\right]^{\frac{9}{2}}}$$

$$= 1 - \left(\frac{\frac{2}{9}k+1}{\frac{2}{9}(k+2.902)+1}\right)^{\frac{9}{2}}$$

$$= 1 - \left(\frac{2k+9}{2k+9+5.804}\right)^{\frac{9}{2}}$$

$$\therefore \text{Percentage Decrease} = 1 - \left(1 - \frac{5.804}{2k+14.804}\right)^{4.5}$$

Figure 87

Code Snippet to Generate Plot for Percentage Decrease

```
# Formula (as derived in report)
percentage_change <- function(initial_point_in_transformed) {
  return (1 - (1 - 5.804 / (2 * initial_point_in_transformed + 5.804)) ^ 4.5)
}

# Plot a graph for all possible values in dataset
x_values <- hypothesis_dataset %>%
  filter(OS_Family == "Linux", WebServer == "Apache") %>%
  select("Loss")

# Generate a data frame to store the data
data_for_graph <- data.frame(
  Loss = x_values,
  Percentage = sapply(x_values, percentage_change)
)

# Set correct column names if necessary
colnames(data_for_graph) <- c("Loss", "Percentage")

# Draw the graph
plot(x = data_for_graph$Loss,
     y = data_for_graph$Percentage * 100,
     main = "Percentage Decrease of Loss",
     xlab = "Loss (after transformation and after reduction)",
     ylab = "Percentage (%)")

# Print results
cat("Minimum percentage:",
    min(data_for_graph$Percentage) * 100, "%\n",
    "Maximum percentage:",
    max(data_for_graph$Percentage) * 100, "%\n")
```

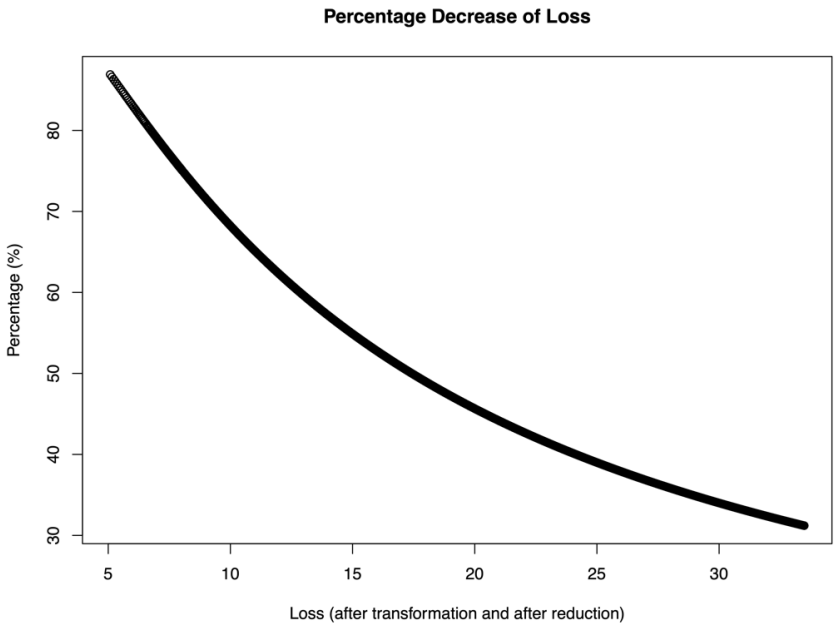
Figure 88

Output for Plot, Minimum and Maximum Percentage

Minimum and Maximum Percentage

Minimum percentage:
31.20542 %

Maximum percentage:
86.90274 %



As we observe, the percentage change ranges from a minimum of 31.2% up to 86.9%, which exceeds the 10% specified in the alternative hypothesis. Supported with a *p*-value of 0.04636, we can hereby conclude a few points:

- Websites running Apache web servers on a Linux operating system will have their loss reduced by a minimum of 31.2% if compared to baseline configurations.
- The increase in downtime leads to a decrease in the reduction of loss, which amplifies the reduction effect.

In conclusion, the null hypothesis is rejected. It is statistically supported that websites running Apache web servers on a Linux operating system will have at least a reduction of 10% in loss if compared to baseline configurations, and the reduction effect amplifies with increasing downtime.

Additional Features – Group

Jaro-Winkler Distance

The Jaro-Winkler distance method is implemented while standardizing countries to obtain the best matches automatically, saving us from writing functions for this purpose.

Figure 89

Implementation of Jaro-Winkler Distance in R Code

```
# Function to standardize country name
standardize_country <- function(country) {

  # Return NA if input is missing
  if (is.na(country)) return(NA)

  # Use Jaro-Winkler distance to calculate match index
  distances <- stringdist::stringdist(
    tolower(country),                    # String to be compared with
    tolower(country_name_list),          # The list to be compared from
    method = "jw"                        # Method used is Jaro-Winkler string distance
  )

  # Retrieve the country name with smallest distance (highest similarity)
  best_index <- which.min(distances)
  best_match <- country_name_list[best_index]

  # Return the most matching country name
  return(best_match)
}
```

MICE Algorithm

The MICE algorithm is used to impute data as it produces more robust estimates compared to methods like imputing with mean, which lacks accuracy, and kNN, which requires significant computational capabilities.

Figure 90

Implementation of MICE Algorithm in R Code

```
# Create predictor matrix
pred_matrix <- make.predictorMatrix(hacking_data_preprocessed)

# Perform imputation
mice_imputation <- mice(hacking_data_preprocessed,
                        m = 3,                          # Create 3 datasets with imputed values
                        maxit = 20,                     # Perform up to 20 times of iteration
                        method = "pmm",                 # Uses predictive mean matching as the imputation method
                        pred = pred_matrix)             # The variables involved

# Save the imputed data to the dataset
hacking_data_preprocessed <- complete(mice_imputation)
```

Box-Cox Transformation

The Box-Cox transformation is applied to the Loss attribute as it can stabilize variance, a key requirement for many statistical models, and it can also handle different types of skewed data.

Figure 91

Implementation of Box-Cox Transformation in R Code

```
# Since box-cox yields the best result (closest to 0), applying box-cox to the dataset
boxcox_result <- boxcox(Loss ~ 1, data = hacking_data_preprocessed)
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
hacking_data_preprocessed$Loss <- (hacking_data_preprocessed$Loss^lambda - 1) / lambda
```

Conclusion

Through an analysis of web defacement incidents, our team has gained valuable insights into how specific configurations for operating systems and web servers can help reduce losses. While significant associations were not found in other factors, they should not be ignored entirely, as interesting patterns found during exploratory data analysis might also hint at their influences toward loss.

Recommendations

To reduce financial losses due to cyberattacks, steps like optimizing web server security by adhering to good practices and conducting audits prior to any special seasons should be implemented by organizations to prevent hackers from having a chance to maximize their profits from the websites. Also, organizations can further implement steps like risk mitigation strategies, server recovery systems, and advanced threat detection systems to enhance their defenses against web defacement attacks so that organizations can protect their data while minimizing losses.

Limitations and Future Direction

As our team primarily conducts exploratory data analysis, no predictive or prescriptive analysis was performed, making our analysis unable to provide insights into future events. Biasness might also exist in the dataset as our team is unsure how the data is collected. Hence, aside from incorporating more advanced techniques like machine learning models, future researchers might also perform additional methods to detect and address any biases in the dataset so that they can provide insights for organizations to mitigate potential risks caused by web defacement incidents.

(5491 words)

References

Bergal, J. (2017, May 19). *'WannaCry' Ransomware Attack Raises Alarm Bells for Cities, States*. Retrieved from Stateline: https://stateline.org/2017/05/19/wannacry-ransomware-attack-raises-alarm-bells-for-cities-states/

Cyber Threat Alliance. (2024, April). *Cyber Threats to NGOs.* Retrieved from Cyber Threat Alliance: https://www.cyberthreatalliance.org/wp-content/uploads/2024/04/Cyber-Threats-to-NGOs_FINAL.pdf

Devansh. (2023, March 21). *Why you should analyze the distribution of your Data*. Retrieved from Medium: https://machine-learning-made-simple.medium.com/why-you-should-analyze-the-distribution-of-your-data-695fd9f0f1be

Finnerty, K. (2023, November 16). *Navigating Through Time: The History of Windows Server Operating Systems*. Retrieved from Medium: https://medium.com/@Kevin_Finnerty_Gabagool/navigating-through-time-the-history-of-windows-server-operating-systems-a4b3cab42b5e

Industry Perspectives. (2024, September 17). *History of Cybersecurity: Key Changes Since the 1990s and Lessons for Today*. Retrieved from ITProToday: https://www.itprotoday.com/vulnerabilities-threats/history-of-cybersecurity-key-changes-since-the-1990s-and-lessons-for-today

Infomineo. (2024, May 24). *Trend Analysis Essentials: Understanding Components, Applications, and Practical Steps*. Retrieved from Infomineo: https://infomineo.com/business-research/trend-analysis-essentials-components-applications-steps/

James, E. (2019, June 18). *What is Correlation Analysis? A Definition and Explanation*. Retrieved from FlexMR: https://blog.flexmr.net/correlation-analysis-definition-exploration

Jones, J. (2007, June 13). *Days-of-risk in 2006 : Linux, Mac OS X, Solaris and Windows*. Retrieved from CSO: https://www.csoonline.com/article/544283/data-protection-days-of-risk-in-2006-linux-mac-os-x-solaris-and-windows.html

Lee, T. B. (2015, May 15). *The Heartbleed Bug, explained*. Retrieved from Vox: https://www.vox.com/2014/6/19/18076318/heartbleed

MrXcrypt. (2025, February 8). *Shellshock — A deep dive into CVE-2014–6271*. Retrieved from Medium: https://infosecwriteups.com/shellshock-a-deep-dive-into-cve-2014-6271-3eb5b33e5de6

NRS. (2024, December 24). *What is a Class B IP Address?* Retrieved from NRS: https://www.nrs.help/post/what-is-a-class-b-ip-address

Ocelic, D. (2023, July 23). *History of Unix, BSD, GNU, and Linux*. Retrieved from CrystalLabs: https://crystallabs.io/unix-history/

OECD. (2014, July 15). *Recommendation of the Council on Digital Government Strategies*. Retrieved from OECD: https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0406

Pandian, S. (2025, February 4). *Time Series Analysis: Definition, Components, Methods, and Applications*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/

Pereira, M. (2021, November 10). *How Cyber Monday Is A Cyber Security Nightmare*. Retrieved from DarkTrace: https://darktrace.com/blog/hacking-season-why-cyber-monday-presents-a-cyber-security-nightmare

Smitter. (2023, May 2). *The history of HTTPS*. Retrieved from DEV Community: https://dev.to/smitterhane/how-tls-was-born-to-secure-modern-age-internet-45jb

Williams, M. (2021, July 9). *Linux vs. Windows: How to Pick the Best Server OS for Your Website*. Retrieved from PCMag: https://www.pcmag.com/how-to/linux-vs-windows-how-to-pick-the-best-server-os-for-your-website

Appendix

Workload Matrix – Group Component

Task	Group Members			
	Chan Chun Ming (TP068983)	**Eu Jun Hong (TP068580)**	**Lim Beng Rhui (TP068495)**	**Ng Xuan Jack (TP067678)**
Introduction	-	-	100%	-
Data Description	-	50%	-	50%
Assumptions	50%	-	50%	-
Hypothesis and Objectives	37.5%	18.75%	25%	18.75%
Data Preprocessing and Validation	25%	20%	35%	20%
Hypothesis Testing	30%	20%	30%	20%
Conclusion	50%	25%	-	25%
Overall Contribution	30%	20%	30%	20%