GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

INTAKE CODE: APU/APD2F2411MMT, APU/APD2F2411CGD

LECTURER: FARHANA ILLIANI BINTI HASSAN

DATE ASSIGNED: 6th JANUARY 2025

DATE COMPLETED: 19TH FEBRUARY 2025

GROUP NO: 19

Student ID	Student Name
TP078098	AZWA AL ISLAM
TP072345	LARA DOMINIQUE ISIDRO CASTRO
TP079398	NARINA KAUR SIDHU
TP077536	RYAN YEW KHY HERN

Table of Contents

1.0 Introduction

Cyberattacks, particularly website defacement, are a growing threat with financial, reputational, and operational consequences. This study analyses real data to uncover trends and correlations, helping organizations strengthen their defenses. By examining key factors influencing attack severity, we provide data-driven insights for cybersecurity improvements. As threats evolve, understanding past incidents can aid in forecasting and mitigating future risks, ultimately enhancing digital resilience.

1.1 Data Description

The dataset spans 15 years of cyberattacks (2000 to 2015) on websites, including attack dates, notification sources, compromised URLs, and geographical details like IP addresses and countries. It also covers technical aspects such as operating systems, web servers, and encoding, alongside the impact of attacks, including ransom amounts, downtime, and financial losses.

1.2 Hypothesis & Objectives

Hypothesis: Organizations that 1) utilize certain web server versions, 2) operate on specific server operating systems, and 3) use particular web server types are more likely to 1) experience higher financial losses due to defacement incidents, 2) face increased hacking vulnerability, 3) be subjected to higher ransom demands, and 4) suffer greater financial losses based on the amount of ransom paid and downtime experienced.

Ryan Yew Khy Hern, TP077536

Objective 1: To investigate the relationship between web server and financial loss

- Analysis 1-1: Web Server Versions Associated with the Highest Revenue Loss
    - Identify which web server versions are linked to the most significant revenue losses due to defacement.
- Analysis 1-2: Correlation Between Web Server Version and Average Loss

    - Calculate the average revenue loss per incident for each web server version and analyse correlation trends.
- Analysis 1-3: Web Server Version vs. Downtime Impact on Financial Loss
    - Investigate whether specific web server versions contribute to longer downtimes and how that affects financial loss.
- Analysis 1-4: Web Server Vulnerability Trend Over Time
    - Analyse the historical trend of web server vulnerabilities by tracking which versions have been defaced most frequently over time.
- Analysis 1-5: Financial Loss Distribution Across Different Web Server Types
    - Categorize web servers (e.g., Apache, Nginx, IIS, etc.) and analyse the distribution of financial losses to determine which type is the most financially impacted.

Lara Dominique Isidro Castro, TP072345

Objective 2: To investigate the influence of server operating systems and web server types on hacking vulnerability and financial loss

- Analysis 2-1: Which server operating systems are most commonly associated with hacking incidents?
- Analysis 2-2: Is there a significant correlation between specific server operating systems and financial loss?
- Analysis 2-3: How does the distribution of downtime vary across different server operating systems?
- Analysis 2-4: What is the variability in financial loss associated with different web server types?
- Analysis 2-5: Do certain server OS and web server combinations show a higher risk of hacking vulnerability?

Narina Kaur Sidhu, TP079398

Objective 3: To investigate the relationship between ransom demands and encoding methods on financial loss.

- Analysis 3-1: What is the relationship between ransom demands and financial loss?
    - Investigate whether higher ransom demands correlate with higher financial losses.
- Analysis 3-2: Do specific encoding methods result in higher financial losses?
    - Compare financial losses across different encoding methods (e.g., UTF-8, ISO-8859-1, Windows-1252).
- Analysis 3-3: How does the frequency of different encoding methods vary across ransom attacks?
    - Analyse which encoding methods are most frequently used in ransomware attacks and whether certain methods dominate.
- Analysis 3-4: Does ransom amount and encoding method collectively influence the prediction of financial loss?
    - Investigate the combined effect of ransom and encoding methods on loss.

#Azwa Al Islam, TP078098

Objective 4: To investigate if financial loss is influenced by the amount of ransom paid and downtime experienced.

- Analysis 4-1: Does an increased downtime cause a larger financial loss?
    - Analyse total financial loss across different downtime categories.
- Analysis 4-2: How does the distribution of financial loss vary across different ransom amounts categorized as low, medium, and high?
    - Examine loss distribution for low, medium and high categories.
- Analysis 4-3: Which has a greater impact on financial loss—ransom or downtime?
    - Compare correlations and regressions for ransom and downtime with loss.
- Analysis 4-4: Can financial loss be predicted using ransom and downtime?
    - Evaluate whether both factors combined improve loss prediction.

2.0 Data Preparation

2.1 Data Import

```r
# Load necessary libraries
library(dplyr)
library(stringr)
library(lubridate)
library(ggplot2)
library(countrycode)  # For country name standardization
library(tidyverse)  # Data wrangling
library(janitor)  # Cleaning column names
library(readr)
library(zoo) # For rolling mean
library(scales)
library(forcats)              # For handling factors in a cleaner way
library(corrplot)             # For creating correlation points
library(RColorBrewer)         # For color palettes
library(viridis)  # For color scales
```

The necessary libraries are loaded to support data manipulation, visualization, and analysis. These include dplyr and tidyverse for data wrangling, ggplot2 for visualization, and corrplot for correlation analysis, among others.

```r
# Read the CSV file "4.hackingdata.csv" from the current working directory
df <- read.csv("4.hackingdata.csv")

# Function to check for both NA and empty string values
missing_values <- colSums(is.na(df) | df == "")

# Convert to data frame for better readability
missing_values_df <- data.frame(Column = names(missing_values), Missing_Values = missing_values)
```

The dataset "4.hackingdata.csv" is then read into df. To ensure data completeness, a function checks for missing values and empty strings across all columns. The results are stored in missing_values_df for better readability, allowing for a clearer overview of data quality.

2.2 Cleaning / Pre-Processing / Data Validation

```r
# Clean the 'Date' column
# Step 1: Convert the 'Date' column to Date format (Automatically detects format)
df$Date <- parse_date_time(df$Date, orders = c("ymd", "dmy", "mdy", "d-b-Y"))

# Step 2: Determine the latest date in the dataset
latest_date <- max(df$Date, na.rm = TRUE)  # Find the most recent date in the dataset

# Step 3: Calculate the cutoff date (15 years before the latest recorded date)
cutoff_date <- latest_date - years(15)

# Step 4: Filter the dataset to keep only records from the past 15 years
df_filtered <- df %>% filter(Date >= cutoff_date)

# Step 5: Extract Year and Month for trend analysis
# Extract Year and Month as numeric values
df_filtered <- df_filtered %>%
  mutate(Year = year(Date),
         Month = month(Date))  # Ensure Month is numeric (1-12)

# Step 6: Verify the range of dates in the filtered dataset
summary(df_filtered$Date)
range(df_filtered$Date)  # Check if only data from the past 15 years is included
```

The 'Date' column is standardized, and only records from the past 15 years are retained.

```r
# Cleaning & Validating the WebServer Column
# Step 1: Inspect WebServer Column
summary(df_filtered$WebServer)  # Check summary stats
unique(df_filtered$WebServer)   # Identify inconsistencies

# Step 2: Convert to Lowercase & Trim Spaces
df_filtered$WebServer <- tolower(str_trim(df_filtered$WebServer))

# Step 3: Handle NA and Junk Values WITHOUT Removing Rows
df_filtered <- df_filtered %>%
  mutate(
    WebServer = ifelse(is.na(WebServer) | WebServer %in% c("", "noyb", "***************", "&quot;&quot;", "-", "+"), "unknown", WebServer),
    WebServer = ifelse(WebServer %in% c("webserver", "hosting", "generico web server v 5.46"), "other", WebServer)
  )

# Step 4: Count "Unknown" Web Server Cases Before Imputation
unknown_before <- df_filtered %>% filter(str_to_lower(WebServer) == "unknown") %>% nrow()
loss_unknown_before <- df_filtered %>% filter(str_to_lower(WebServer) == "unknown") %>% summarise(sum(Loss, na.rm = TRUE))

print(paste("Before Imputation - Unknown Web Server Count:", unknown_before))
print(paste("Before Imputation - Financial Loss from Unknowns: $", loss_unknown_before))

# Step 5: Impute "Unknown" WebServer Names Using OS Information
df_filtered <- df_filtered %>%
  mutate(WebServer = case_when(
    str_to_lower(WebServer) == "unknown" & str_detect(str_to_lower(OS), "windows") ~ "microsoft-iis",
    str_to_lower(WebServer) == "unknown" & str_detect(str_to_lower(OS), "linux") ~ "apache",
    str_to_lower(WebServer) == "unknown" & str_detect(str_to_lower(OS), "bsd") ~ "nginx",
    TRUE ~ WebServer  # Keep original values
  ))

# Step 6: Count "Unknown" Web Server Cases After Imputation
unknown_after <- df_filtered %>% filter(str_to_lower(WebServer) == "unknown") %>% nrow()
loss_unknown_after <- df_filtered %>% filter(str_to_lower(WebServer) == "unknown") %>% summarise(sum(Loss, na.rm = TRUE))

print(paste("After Imputation - Unknown Web Server Count:", unknown_after))
print(paste("After Imputation - Financial Loss from Remaining Unknowns: $", loss_unknown_after))
```

```r
# Step 7: Reapply Standardization & Version Extraction AFTER Imputation
df_filtered <- df_filtered %>%
  mutate(
    WebServer_Base = case_when(
      str_detect(WebServer, "apache") ~ "apache",
      str_detect(WebServer, "nginx") ~ "nginx",
      str_detect(WebServer, "iis") ~ "microsoft-iis",
      str_detect(WebServer, "lighttpd") ~ "lighttpd",
      str_detect(WebServer, "gws") ~ "google web server",
      str_detect(WebServer, "litespeed") ~ "litespeed",
      str_detect(WebServer, "tomcat") ~ "apache tomcat",
      str_detect(WebServer, "oracle") ~ "oracle server",
      str_detect(WebServer, "zeus") ~ "zeus",
      str_detect(WebServer, "ats") ~ "ats",
      str_detect(WebServer, "varnish") ~ "varnish",
      str_detect(WebServer, "cherrypy") ~ "cherrypy",
      str_detect(WebServer, "ideawebserver") ~ "ideawebserver",
      str_detect(WebServer, "bigip") ~ "bigip",
      str_detect(WebServer, "sun-java-system-web-server") ~ "sun-java-web-server",
      str_detect(WebServer, "modsecurity") ~ "modsecurity",
      str_detect(WebServer, "cloudflare-nginx") ~ "cloudflare",
      WebServer == "unknown" ~ "unknown",  # Preserve "unknown" entries
      TRUE ~ "other"  # Classify everything else as "other"
    )
  )

# Step 8: Extract the Web Server Version (Ensuring it Reflects Imputed Values)
df_filtered <- df_filtered %>%
  mutate(
    WebServer_Version = str_extract(WebServer, "(?<=\\b)[0-9]+(\\.[0-9]+)*"),
    WebServer_Version = na_if(WebServer_Version, "unknown")  # Convert 'unknown' to NA for proper handling
  )

# Step 9: Recreate the Full Web Server Label AFTER Imputation
df_filtered <- df_filtered %>%
  mutate(
    WebServer_Full = case_when(
      is.na(WebServer_Version) ~ WebServer_Base,  # Keep only base if version is missing
      TRUE ~ paste(WebServer_Base, WebServer_Version)  # Otherwise, use full version
    )
  )

# Step 10: Check for Multi-Version Entries (Remove or Handle Separately)
df_filtered <- df_filtered %>%
  filter(!str_detect(WebServer, "&|,|;"))  # Remove multi-server cases if needed

# Step 11: Verify Cleaned Data
unique(df_filtered$WebServer_Base)  # Check standardized names
unique(df_filtered$WebServer_Version)  # Check extracted versions
summary(df_filtered$WebServer_Base)
```

Web server data is standardized by normalizing text, handling missing or invalid values, and categorizing servers into broad groups with extracted versions.

```r
# Clean 'Country' column
# https://rpubs.com/Teal_Emery/cleaning_intl_data_tips_and_tricks
# Helper function: Convert country name to ISO3C code
country_regex_to_iso3c <- function(country_string) {
  country_string %>%
    countrycode::countrycode(origin = "country.name", destination = "iso3c", origin_regex = TRUE)
}

# Helper function: Convert ISO3C back to standardized country name
iso3c_to_country_name <- function(country_string) {
  country_string %>%
    countrycode::countrycode(origin = "iso3c", destination = "country.name")
}

# Step 1: Standardize country names BEFORE applying countrycode
df_filtered <- df_filtered %>%
  mutate(Country = str_trim(Country)) %>%
  mutate(Country = case_when(
    # Handle completely unknown/missing values
    Country %in% c("", "Unknown", "UNKNOWN") ~ "Unknown",

    # Handle geographic regions and ambiguous names
    Country %in% c("Africa", "America", "Asia", "EUROPE", "SouthAmerica", "WestEuro",
                   "EastEuro", "MiddleEast", "Oseania", "ASIA/PACIFIC REGION",
                   "European Union", "European Uni") ~ "Unknown",

    # Handle invalid proxy/provider values
    Country %in% c("Anonymous Proxy", "ANONYMOUS PROXY", "Satellite Provider", "SATELLITE PROVIDER") ~ "Unknown",

    # Fix incorrect or non-standard country names
    Country == "T,rkiye" ~ "Turkey",
    Country == "U.S.A" | Country == "Usa" ~ "United States",
    Country == "United State" ~ "United States",
    Country == "Uk" | Country == "United Kingd" ~ "United Kingdom",
    Country == "Uae" ~ "United Arab Emirates",
    Country == "MICRONESIA" ~ "Federated States of Micronesia",
    Country == "CALEDONIA" | Country == "New Caledoni" ~ "New Caledonia",
    Country %in% c("ASCENSIONISLAND", "Ascension Island") ~ "Saint Helena, Ascension and Tristan da Cunha",
    Country == "French Polyn" ~ "French Polynesia",
    Country == "Virgin Islan" | Country == "Virgin Islands" ~ "United States Virgin Islands",
    Country == "YUGOSLAVIA" ~ "Serbia",  # Yugoslavia no longer exists
    TRUE ~ Country
  ))

# Step 2: Apply ISO3C conversion and handle missing matches
df_filtered <- df_filtered %>%
  mutate(iso3c = country_regex_to_iso3c(Country)) %>%
  mutate(country_name = iso3c_to_country_name(iso3c)) %>%
  mutate(country_name = ifelse(is.na(country_name) | country_name == "", "Unknown", country_name))
```

Country data is cleaned by standardizing names, correcting errors, and converting to ISO3C codes, with ambiguous or invalid entries labelled as "Unknown" for consistency.

```r
# Clean 'OS' column
# Step 1: Inspect the OS column to identify unique values and check for inconsistencies
# View unique OS entries
unique(df_filtered$OS)

# Check for missing values or empty entries in the OS column
sum(is.na(df_filtered$OS) | df_filtered$OS == "")

# Step 2: Clean the OS column
# Convert the OS column to lowercase and trim any extra spaces
df_filtered$OS <- tolower(str_trim(df_filtered$OS))

# Replace missing or empty values with "unknown"
df_filtered <- df_filtered %>%
  mutate(OS = ifelse(is.na(OS) | OS == "", "unknown", OS))

# Step 3: Standardize OS Names
# Create a new column 'OS_Base' to classify the OS into categories
df_filtered <- df_filtered %>%
  mutate(
    OS_Base = case_when(
      str_detect(OS, "windows") ~ "windows",    # Categorize Windows OS
      str_detect(OS, "linux") ~ "linux",        # Categorize Linux OS
      str_detect(OS, "mac") ~ "macos",          # Categorize MacOS
      str_detect(OS, "unix") ~ "unix",          # Categorize Unix
      str_detect(OS, "android") ~ "android",    # Categorize Android
      str_detect(OS, "ios") ~ "ios",            # Categorize iOS
      str_detect(OS, "bsd") ~ "bsd",            # Categorize BSD
      OS == "unknown" ~ "unknown",              # Preserve "unknown"
      TRUE ~ "other"                            # Classify anything else as "other"
    )
  )

# Step 4: Extract OS version numbers (if available)
# Use regular expressions to extract version numbers
df_filtered$OS_Version <- str_extract(df_filtered$OS, "[0-9]+(\\.[0-9]+)*")

# Step 5: Verify the cleaned data
# Check unique standardized OS categories
unique(df_filtered$OS_Base)

# Check unique extracted OS versions
unique(df_filtered$OS_Version)

# Summary of the OS_Base column
summary(df_filtered$OS_Base)

# Step 6: Final Check for missing or incorrect data
# Verify there are no unexpected missing values in OS or OS_Version
sum(is.na(df_filtered$OS) | df_filtered$OS == "")
sum(is.na(df_filtered$OS_Version))
```

The OS column is cleaned by addressing inconsistencies, converting names to lowercase, removing extra spaces, and replacing missing values with “Unknown”. A new column, ‘OS_Base’, categorises OS into broad groups, labelling unrecognized values as “Other.” The cleaned data is verified by reviewing unique OS categories, checking version numbers, and ensuring no missing values.

```r
# Clean 'Encoding' column
# Replace "NULL" and empty values with NA
df_filtered$Encoding <- ifelse(df_filtered$Encoding %in% c("NULL", "", NA), NA, df_filtered$Encoding)

# Standardize encoding names
df_filtered$Encoding <- tolower(df_filtered$Encoding)  # Convert to lowercase
df_filtered$Encoding <- gsub("utf-8", "UTF-8", df_filtered$Encoding)
df_filtered$Encoding <- gsub("iso-8859-1", "ISO-8859-1", df_filtered$Encoding)
df_filtered$Encoding <- gsub("windows-1252", "Windows-1252", df_filtered$Encoding)
df_filtered$Encoding <- gsub("gb2312", "GB2312", df_filtered$Encoding)
df_filtered$Encoding <- gsub("big5", "BIG5", df_filtered$Encoding)

# Fill missing values with "Unknown"
df_filtered$Encoding[is.na(df_filtered$Encoding)] <- "Unknown"

# Check results
table(df_filtered$Encoding)
```

Encoding column reveals that "NULL" and "N" are present as categories. To improve readability and minimise errors, these categories are grouped and renamed as "NA". Furthermore, all entries are converted to lowercase and remaining missing values are labelled "Unknown."

```r
#Cleaning 'Loss' column

# 1. Checking for missing values or empty strings in the 'Loss' column
missing_loss_before <- sum(is.na(df_filtered$Loss) | df_filtered$Loss == "")
print(paste("Missing Loss values before cleaning:", missing_loss_before))

# 2. Convert 'Loss' column to numeric by removing commas and handling characters
df_filtered <- df_filtered %>%
  mutate(Loss = as.character(Loss),  # Ensure it's character before replacing commas
         Loss = gsub(",", "", Loss),  # Remove commas
         Loss = as.numeric(Loss))     # Convert to numeric

# 3. Impute Missing Values by Median Imputation per Web Server Group
df_filtered <- df_filtered %>%
  group_by(WebServer_Base) %>%
  mutate(Loss = ifelse(is.na(Loss), ifelse(all(is.na(Loss)), 0, median(Loss, na.rm = TRUE)), Loss)) %>%
  ungroup()  # Remove grouping after imputation

# 4. Checking for remaining missing values after imputation
missing_loss_after <- sum(is.na(df_filtered$Loss))
print(paste("Missing Loss values after cleaning:", missing_loss_after))
```

The ‘Loss’ column is cleaned by detecting missing values, converting it to numeric, and applying median imputation per Web Server Base group. Missing values are replaced with the group median, or 0 if all values are missing.

```r
#Cleaning 'Ransom' column

# 1. Checking for missing values or empty strings in the 'Ransom' column
missing_ransom_before <- sum(is.na(df_filtered$Ransom) | df_filtered$Ransom == "")
print(paste("Missing Ransom values before cleaning:", missing_ransom_before))

# 2. Convert 'Ransom' column to numeric by removing commas and handling characters
df_filtered <- df_filtered %>%
  mutate(Ransom = as.character(Ransom),  # Ensure it's character before replacing commas
         Ransom = gsub(",", "", Ransom),  # Remove commas
         Ransom = as.numeric(Ransom))     # Convert to numeric

# 3. Impute Missing Values by Mean
df_filtered <- df_filtered %>%
  mutate(Ransom = ifelse(is.na(Ransom), mean(Ransom, na.rm = TRUE), Ransom))

# 4. Checking for remaining missing values after imputation
missing_ransom_after <- sum(is.na(df_filtered$Ransom))
print(paste("Missing Ransom values after cleaning:", missing_ransom_after))
```

The ‘Ransom’ column is cleaned by identifying missing values, converting it to numeric, and applying mean imputation to replace missing values with the column mean.

3.0 Data Analysis

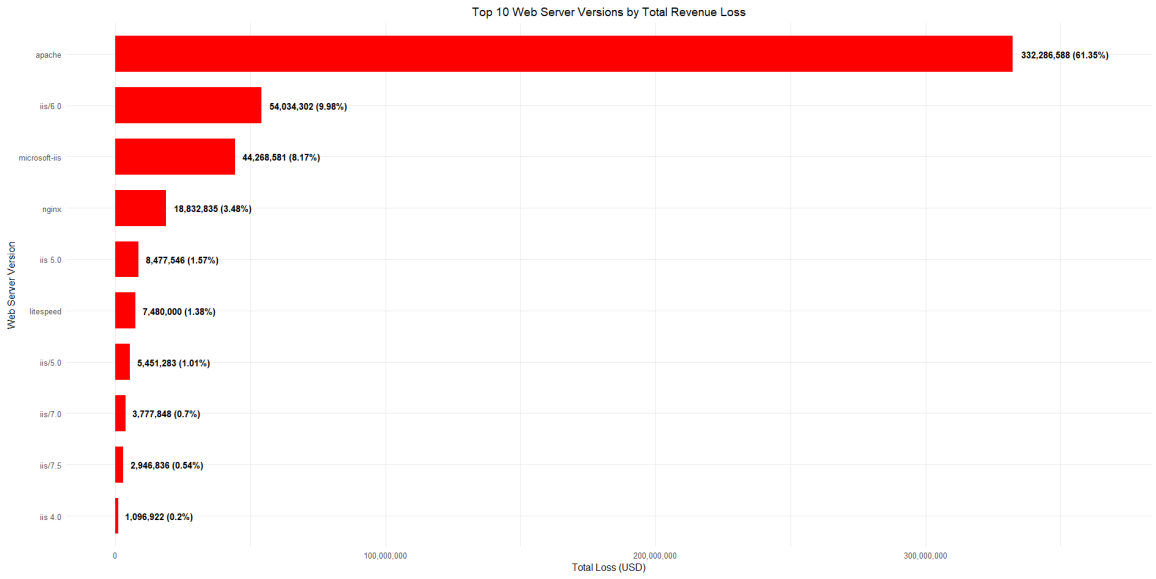
3.1 **Ryan Yew Khy Hern, TP077536** — Objective 1: To investigate the relationship between web server and financial loss

Analysis 1-1: Web Server Versions Associated with the Highest Revenue Loss

```r
# Step 1: Recalculate Aggregation Without "Unknown"
webserver_grouped <- df_filtered %>%
  filter(str_to_lower(webServer) != "unknown") %>%
  group_by(webServer) %>%
  summarise(
    Total_Loss = sum(Loss, na.rm = TRUE),
    Incident_Count = n(),
    Avg_Loss = mean(Loss, na.rm = TRUE),
    Median_Loss = median(Loss, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(desc(Total_Loss))

# Step 2: Display the Top 10 Web Server Versions by Total Loss
top_webservers <- webserver_grouped %>% slice_max(Total_Loss, n = 10)
print(top_webservers)
```

```r
# Step 3: Visualize the Data
ggplot(top_webservers, aes(x = reorder(webServer, Total_Loss), y = Total_Loss)) +
  geom_col(fill = "red", width = 0.7) +  # Improve bar spacing
  coord_flip() +  # Flip for better readability
  labs(title = "Top 10 Web Server Versions by Total Revenue Loss",
       x = "Web Server Version",
       y = "Total Loss (USD)") +
  scale_y_continuous(labels = scales::comma, limits = c(0, 350000000)) +  # Format Y-axis and set limit
  geom_text(aes(label = scales::comma(Total_Loss)),
            hjust = -0.1, size = 4, color = "black", fontface = "bold") +  # Position labels to the right
  theme_minimal(base_size = 12) +  # Adjust font size
  theme(plot.title = element_text(hjust = 0.5))  # Center title
```

The analysis identifies the most vulnerable web server versions and their financial impact by filtering out unknown records and aggregating revenue loss per version. The top 10 web servers with the highest financial losses are visualized in a bar chart, highlighting their total loss and percentage contribution.



Apache incurs the highest financial loss at $332.29 million (61.35% of total losses), suggesting frequent targeting or severe consequences when compromised. Microsoft IIS versions, particularly IIS 6.0 ($54.03 million, 9.98%) and Microsoft IIS ($44.27 million, 8.17%), also face significant losses. Nginx ranks fourth with $18.83 million (3.48%) in losses, likely due to misconfigurations or outdated versions. Other web servers, including LiteSpeed and legacy IIS versions, contribute smaller losses. Overall, Apache, IIS 6.0, Microsoft IIS, and Nginx account for over 80% of financial losses, indicating attackers target widely used or poorly secured platforms for maximum impact.

Analysis 1-2: Assessing the Correlation Between Web Server Versions and Average Financial Loss Per Incident

```r
# 1. Aggregate data: Calculate average loss per incident for each web server version
webserver_avg_loss <- df_filtered %>%
  group_by(WebServer_Full) %>%
  summarise(
    Total_Loss = sum(Loss, na.rm = TRUE),
    Incident_Count = n(),
    Avg_Loss_Per_Incident = mean(Loss, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  filter(Incident_Count >= 5)  # Remove web servers with very few incidents to avoid bias
```

This analysis investigates the relationship between web server versions and average financial loss per incident using descriptive statistics, correlation analysis, and visualization. Data is aggregated to compute total financial loss, incident count, and average loss per incident, with filtering applied to exclude servers with minimal incidents for statistical reliability.

```r
# 2. Compute correlation matrix (ensuring no missing values)
correlation_data <- webserver_avg_loss %>%
  select(Avg_Loss_Per_Incident, Incident_Count, Total_Loss) %>%
  na.omit()  # Remove missing values before correlation analysis

correlation_matrix <- cor(correlation_data, use = "complete.obs", method = "pearson")
print("Correlation Matrix:")
print(correlation_matrix)
```

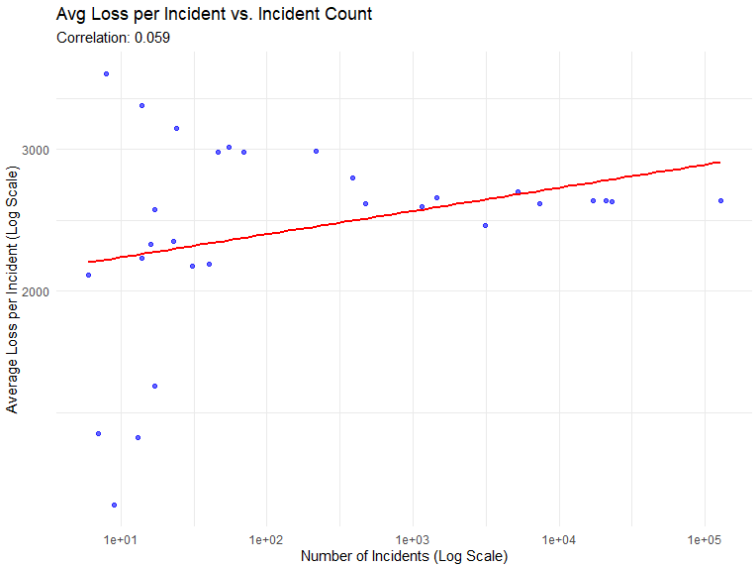
A Pearson correlation matrix assesses whether frequently attacked web servers have higher or lower average losses per incident.

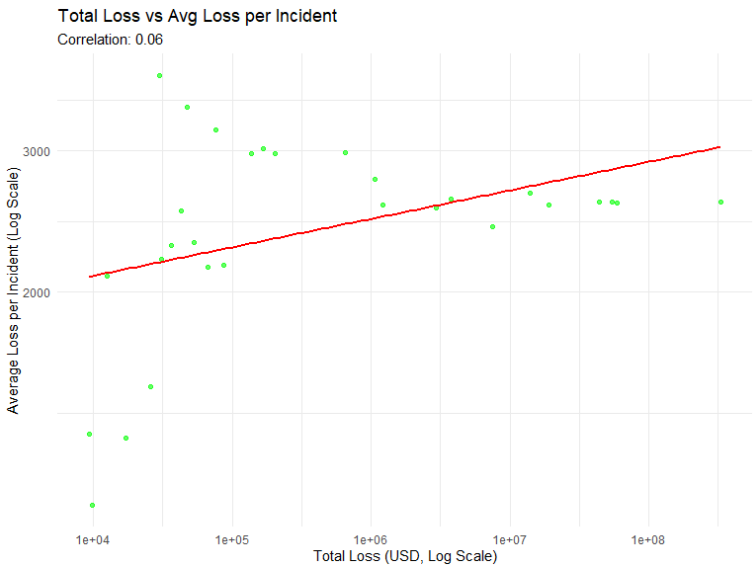
```r
# 3. Scatter plot: Avg Loss per Incident vs Incident Count
ggplot(webserver_avg_loss, aes(x = Incident_Count, y = Avg_Loss_Per_Incident)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  scale_x_log10() +  # Log scale for better visibility
  scale_y_log10() +  # Log scale to prevent extreme values dominating the plot
  labs(title = "Avg Loss per Incident vs. Incident Count",
       subtitle = paste("Correlation:", round(correlation_matrix["Incident_Count", "Avg_Loss_Per_Incident"], 3)),
       x = "Number of Incidents (Log Scale)",
       y = "Average Loss per Incident (Log Scale)") +
  theme_minimal()

# 4. Scatter plot: Total Loss vs Avg Loss per Incident
ggplot(webserver_avg_loss, aes(x = Total_Loss, y = Avg_Loss_Per_Incident)) +
  geom_point(color = "green", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  scale_x_log10() +  # Log scale for better visualization
  scale_y_log10() +
  labs(title = "Total Loss vs Avg Loss per Incident",
       subtitle = paste("Correlation:", round(correlation_matrix["Total_Loss", "Avg_Loss_Per_Incident"], 3)),
       x = "Total Loss (USD, Log Scale)",
       y = "Average Loss per Incident (Log Scale)") +
  theme_minimal()
```

Scatter plots visualize these relationships, comparing incident count with average loss per incident and total financial loss.

```
> print(correlation_matrix)
                      Avg_Loss_Per_Incident Incident_Count Total_Loss
Avg_Loss_Per_Incident            1.00000000     0.05924752 0.05952257
Incident_Count                   0.05924752     1.00000000 0.99999787
Total_Loss                       0.05952257     0.99999787 1.00000000
```

Findings show a near-perfect correlation (0.9999) between incident count and total financial loss, indicating that more frequent attacks lead to higher cumulative losses. However, the correlation between incident count and average loss per incident is weak (0.0592), suggesting that financial damage per attack varies independently of attack frequency. Similarly, total loss and average loss per incident show a weak correlation (0.0595), reinforcing that overall losses are driven by attack frequency rather than individual incident severity.





Scatter plots confirm these findings, showing no clear pattern between incident count and average loss per incident, with only a slight upward trend between total loss and average loss per incident. This suggests that some servers experience frequent low-cost attacks, while others suffer fewer but more financially significant incidents.

Analysis 1-3: Web Server Version vs. Downtime Impact on Financial Loss

```
# Analysis 1-3: Web Server Version vs. Downtime Impact on Financial Loss
# Investigate whether specific web server versions contribute to longer downtimes and how that affects financial loss.

  # 1. Aggregate downtime impact per Web Server Version
  webserver_downtime <- df_filtered %>%
    group_by(WebServer_Full) %>%
    summarise(
      Total_Loss = sum(Loss, na.rm = TRUE),
      Total_Downtime = sum(DownTime, na.rm = TRUE),
      Avg_Downtime = mean(DownTime, na.rm = TRUE),
      Incident_Count = n(),
      Avg_Loss_Per_Incident = mean(Loss, na.rm = TRUE),
      .groups = "drop"
    ) %>%
    filter(Incident_Count >= 5)  # Remove web servers with very few incidents to avoid bias
```

This analysis examines the relationship between web server versions, downtime duration, and financial losses from defacement incidents. It evaluates whether prolonged downtimes lead to higher financial losses and if frequently attacked servers experience extended recovery times. Descriptive statistics compute total and average downtime, financial loss, and incident count, filtering out unreliable data.

```
# 2. Compute correlation matrix (ensuring proper NA handling)
correlation_data <- webserver_downtime %>%
  select(Avg_Downtime, Total_Downtime, Total_Loss, Avg_Loss_Per_Incident, Incident_Count) %>%
  na.omit()  # Remove missing values before correlation analysis

correlation_matrix <- cor(correlation_data, use = "pairwise.complete.obs", method = "pearson")
print("Correlation Matrix between DownTime and Financial Loss:")
print(correlation_matrix)
```

A Pearson correlation analysis assesses whether longer downtimes correlate with greater financial losses.

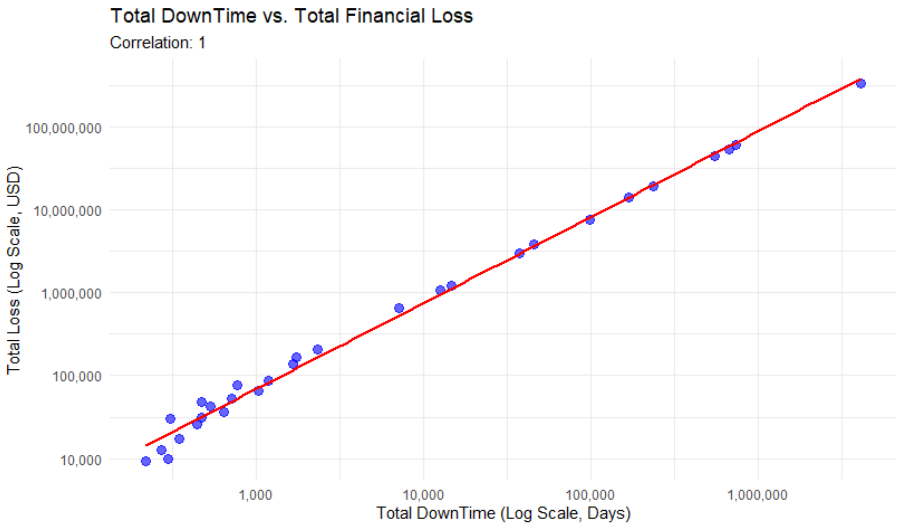
```
# 3. Scatter plot: Total DownTime vs. Financial Loss
ggplot(webserver_downtime, aes(x = Total_Downtime, y = Total_Loss)) +
  geom_point(color = "blue", alpha = 0.6, size = 3) +  # Adjust transparency & size
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  scale_x_log10(labels = scales::comma) +  # Log scale for better visualization
  scale_y_log10(labels = scales::comma) +
  labs(title = "Total DownTime vs. Total Financial Loss",
       subtitle = paste("Correlation:", round(correlation_matrix["Total_Downtime", "Total_Loss"], 3)),
       x = "Total DownTime (Log Scale, Days)",
       y = "Total Loss (Log Scale, USD)") +
  theme_minimal()

# 4. Scatter plot: Average Downtime vs. Average Loss per Incident
ggplot(webserver_downtime, aes(x = Avg_Downtime, y = Avg_Loss_Per_Incident)) +
  geom_point(color = "green", alpha = 0.6, size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  scale_x_log10(labels = scales::comma) +  # Log scale to prevent extreme values dominating the plot
  scale_y_log10(labels = scales::comma) +
  labs(title = "Avg Downtime vs. Avg Loss per Incident",
       subtitle = paste("Correlation:", round(correlation_matrix["Avg_Downtime", "Avg_Loss_Per_Incident"], 3)),
       x = "Average Downtime (Log Scale, Days)",
       y = "Average Loss per Incident (Log Scale, USD)") +
  theme_minimal()
```

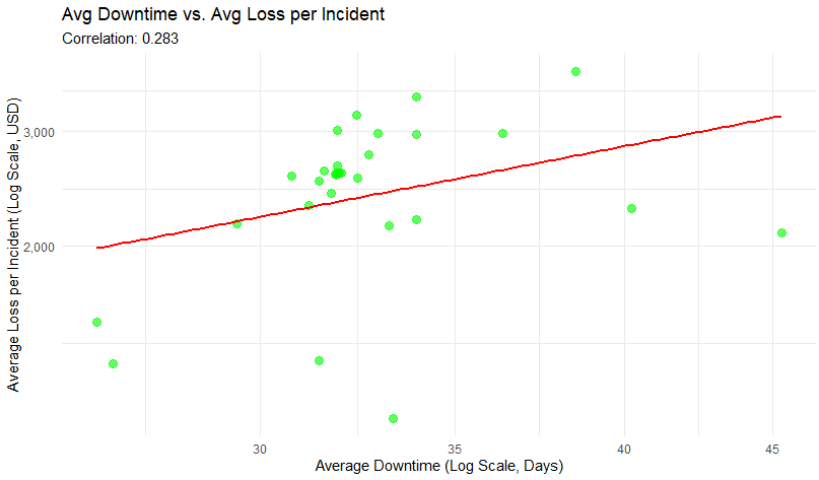
```r
# 5. Identify Top 10 Web Server Versions with the Longest Downtimes
top_downtime_webservers <- webserver_downtime %>%
  arrange(desc(Total_Downtime)) %>%
  slice_max(Total_Downtime, n = 10)

print("Top 10 Web Server Versions with the Longest Downtimes:")
print(top_downtime_webservers)

# 6. Bar Plot for Top 10 Web Servers with the Longest Downtime (Improved)
ggplot(top_downtime_webservers, aes(x = reorder(WebServer_Full, Total_Downtime), y = Total_Downtime)) +
  geom_col(fill = "purple", width = 0.7) +
  coord_flip() +  # Flip for better label visibility
  geom_text(aes(label = scales::comma(Total_Downtime)),
            hjust = -0.1,  # Move text to the right of bars
            color = "black", fontface = "bold", size = 4) +  # Make text clear
  scale_y_continuous(labels = scales::comma, limits = c(0, 4500000)) +  # Set y-axis max to 4,500,000
  labs(title = "Top 10 Web Server Versions with the Longest Downtime",
       x = "Web Server Version",
       y = "Total Downtime (Days)") +
  theme_minimal()
```

Exploratory data analysis (EDA) visualizes Total Downtime vs. Total Financial Loss and Average Downtime vs. Average Loss per Incident using scatter plots, while a bar chart presents the top 10 web servers with the longest downtimes.

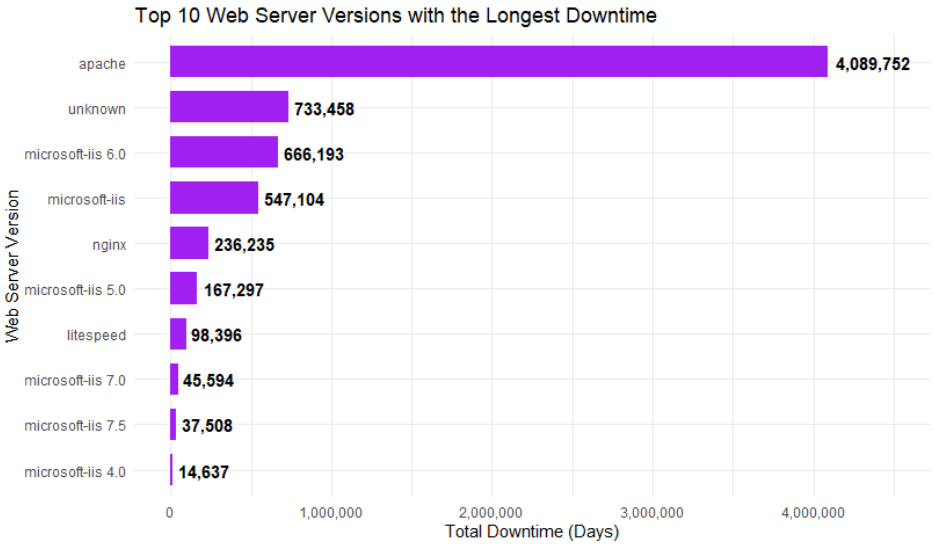
```
> print(correlation_matrix)
                     Avg_Downtime Total_Downtime  Total_Loss Avg_Loss_Per_Incident Incident_Count
Avg_Downtime           1.00000000    -0.08281365 -0.08275735            0.28267526    -0.08288202
Total_Downtime        -0.08281365     1.00000000  0.99999800            0.05930432     0.99999979
Total_Loss            -0.08275735     0.99999800  1.00000000            0.05952257     0.99999787
Avg_Loss_Per_Incident  0.28267526     0.05930432  0.05952257            1.00000000     0.05924752
Incident_Count        -0.08288202     0.99999979  0.99999787            0.05924752     1.00000000
```

Findings show a near-perfect correlation (0.999998) between total downtime and financial loss, confirming that prolonged disruptions increase cumulative losses. A strong correlation (0.999999) between incident count and total downtime suggests frequently attacked web servers also suffer extended downtimes. However, a moderate correlation (0.283) between average downtime and average loss per incident suggests longer downtimes do not always lead to higher losses per attack. Additionally, the weak correlation (-0.0827) between average downtime and total financial loss indicates that frequent incidents drive overall losses more than downtime duration.





Scatter plots reinforce these findings, with Total Downtime vs. Total Financial Loss showing a strong linear trend, while Average Downtime vs. Average Loss per Incident exhibits more variability.



Apache accumulates over 4 million days of downtime, far exceeding any other server. Several Microsoft IIS versions (6.0, 5.0, 7.0, 7.5, and 4.0) also rank among the top 10, highlighting their susceptibility to extended disruptions. Nginx and LiteSpeed also show notable downtimes, emphasizing that downtime issues are not limited to a single provider. These findings suggest that security vulnerabilities, inefficient recovery processes, and frequent targeting contribute to prolonged downtimes, underscoring the need for stronger mitigation strategies.

Extra Feature 1-3

```r
# Extra Feature 1-3: Categorizing Web Servers by Downtime Severity
# ----------------------------------------------------------------
# This extra feature enhances the analysis by grouping web servers into three downtime severity categories:
# - Low Downtime: < 100,000 total downtime
# - Moderate Downtime: 100,000 - 499,999 total downtime
# - High Downtime: ≥ 500,000 total downtime
# By categorizing the data, we can compare the financial impact of different downtime levels more effectively.

webserver_downtime <- webserver_downtime %>%
  mutate(Downtime_Category = factor(case_when(
    Total_Downtime < 100000 ~ "Low Downtime",
    Total_Downtime >= 100000 & Total_Downtime < 500000 ~ "Moderate Downtime",
    Total_Downtime >= 500000 ~ "High Downtime"
  ), levels = c("Low Downtime", "Moderate Downtime", "High Downtime")))  # Set order for better visualization
```

Categorizing web servers into Low, Moderate, and High Downtime groups enables clearer comparisons of financial losses across downtime levels. This classification enhances data interpretation, revealing whether prolonged downtime correlates with higher financial losses and providing actionable insights for risk assessment and proactive security strategies.

Findings confirm that High Downtime web servers suffer the greatest financial losses, with a narrow interquartile range (IQR), indicating prolonged outages almost always lead to severe financial consequences due to operational disruptions, ransom demands, or reputational damage. A few extreme outliers suggest exceptionally high losses, reinforcing the risks of extended service disruptions. The Moderate Downtime category shows mid-range financial losses with low variability, suggesting more predictable financial consequences. In contrast, the Low Downtime category exhibits a wider spread in financial losses, with some servers experiencing significant damage despite minimal downtime. High-loss outliers in this category suggest that factors beyond downtime, such as business criticality, attack severity, or industry risks, also contribute to financial

impact. This categorization improves understanding of downtime's role in financial losses and helps stakeholders refine risk assessment and mitigation strategies.

Analysis 1-4: Web Server Vulnerability Trend Over Time

```
# 1. Aggregate incidents per Web Server Version over time (grouped by year-month)
webserver_trend <- df_filtered %>%
  group_by(Year, Month, WebServer_Full) %>%
  summarise(
    Incident_Count = n(),
    .groups = "drop"
  ) %>%
  mutate(Date = ymd(paste(Year, Month, "01")))  # Convert Year-Month to Date format

# 2. Identify the Top 5 Most Frequently Attacked Web Server Versions
top_vulnerable_webservers <- webserver_trend %>%
  group_by(WebServer_Full) %>%
  summarise(Total_Incidents = sum(Incident_Count)) %>%
  arrange(desc(Total_Incidents)) %>%
  slice_head(n = 5) %>%
  pull(WebServer_Full)

# 3. Filter dataset to include only the top 5 vulnerable web servers
webserver_trend_top <- webserver_trend %>%
  filter(WebServer_Full %in% top_vulnerable_webservers) %>%
  arrange(Date)
```

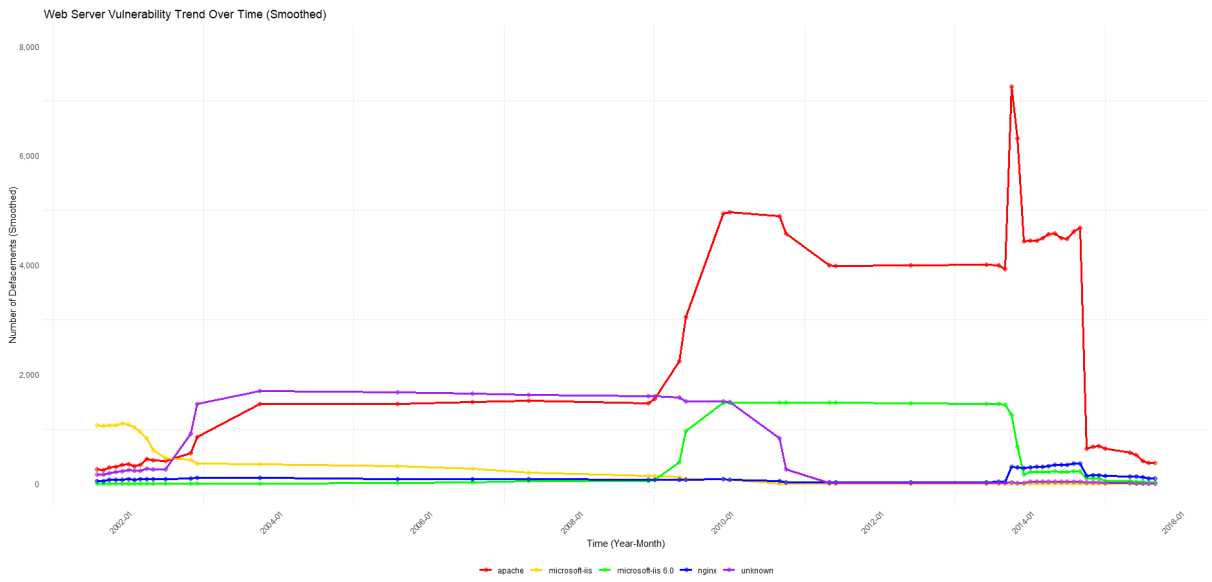
This analysis uses descriptive statistics, time-series analysis, and data visualization to examine trends in web server defacement incidents over time. It aims to identify the most frequently attacked web server versions and analyze their evolving vulnerabilities. Incident counts are aggregated by web server version and grouped by year and month to track attack patterns, with the top five most targeted web servers selected for in-depth analysis.

```
# 4. Apply a Rolling Mean (Moving Average) to Smooth Trends (12-Month Window)
webserver_trend_top <- webserver_trend_top %>%
  group_by(WebServer_Full) %>%
  mutate(Smoothed_Incidents = zoo::rollmean(Incident_Count, k = 12, fill = NA, align = "right"))

# 5. Remove NA values from the smoothed dataset before plotting
webserver_trend_top <- webserver_trend_top %>%
  filter(!is.na(Smoothed_Incidents))
```

A 12-month rolling mean is applied to smooth short-term fluctuations and highlight long-term trends.

```
# 8. Adjust Y-Axis Scaling to Avoid Large Empty Spaces
y_max <- max(webserver_trend_top$Smoothed_Incidents, na.rm = TRUE) * 1.1

# 9. Plot the Smoothed Time Series with Enhanced Formatting
ggplot(webserver_trend_top, aes(x = Date, y = Smoothed_Incidents, color = WebServer_Full)) +
  geom_line(size = 1.2) +  # Slightly thicker lines
  geom_point(alpha = 0.5, size = 2) +  # Points for context
  scale_x_date(date_labels = "%Y-%m", date_breaks = "2 years") +  # Format x-axis with 2-year breaks
  scale_y_continuous(limits = c(0, y_max), labels = scales::comma) +  # Adjust y-axis scaling
  scale_color_manual(values = custom_colors) +  # Use predefined colors
  labs(title = "Web Server Vulnerability Trend Over Time (Smoothed)",
       x = "Time (Year-Month)",
       y = "Number of Defacements (Smoothed)",
       color = "Web Server Version") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
        legend.position = "bottom",  # Move legend below chart
        legend.title = element_blank(),  # Remove legend title for clarity
        panel.grid.major = element_blank())  # Remove major grid lines
```

A color-coded time-series line plot visualizes trends for high-risk web servers.



Findings indicate that Apache was the most targeted between 2009 and 2014, IIS remained a consistent long-term target, and Nginx saw a gradual rise in defacements before stabilizing. The post-2014 decline in attacks across all web servers suggests improvements in cybersecurity practices, including automated updates, stronger security frameworks, and increased awareness of cyber threats.

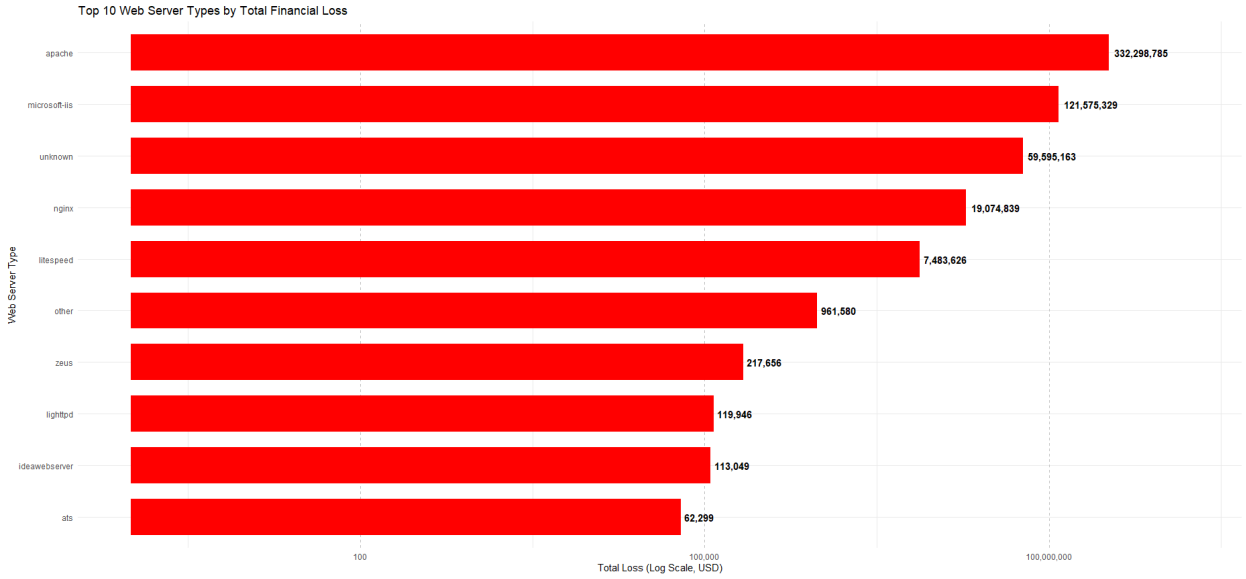
Analysis 1-5: Financial Loss Distribution Across Different Web Server Types

```r
# Aggregate financial loss per Web Server Type (WebServer_Base)
webserver_loss <- df_filtered %>%
  group_by(WebServer_Base) %>%
  summarise(
    Total_Loss = sum(Loss, na.rm = TRUE),  # Sum should now work correctly
    Incident_Count = n(),
    Avg_Loss = mean(Loss, na.rm = TRUE),
    Median_Loss = median(Loss, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(desc(Total_Loss))  # Sort by highest financial impact

# Select top 10 web servers by financial impact
top_webserver_loss <- head(webserver_loss, 10)
```

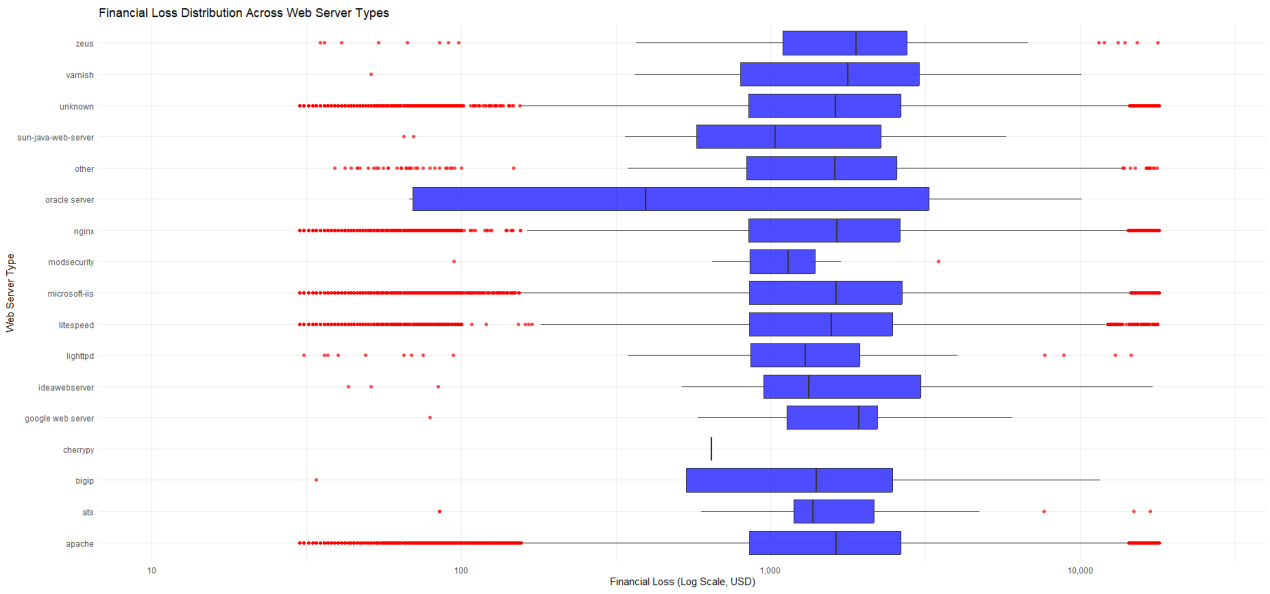
```r
# 1. Define Y-Axis Limit (Extend by 10% for better spacing)
y_max <- max(top_webserver_loss$Total_Loss, na.rm = TRUE) * 5

# 2. Log Scale Bar Chart - Total Loss per Web Server Type
ggplot(top_webserver_loss, aes(x = reorder(WebServer_Base, Total_Loss), y = Total_Loss)) +
  geom_col(fill = "red", width = 0.7) +  # Restore original bar width
  geom_text(aes(label = scales::comma(Total_Loss)),
            hjust = -0.1, size = 4, color = "black", fontface = "bold") +  # Restore standard label positioning
  coord_flip() +  # Flip for better readability
  scale_y_log10(labels = scales::comma, breaks = scales::log_breaks(n = 5), limits = c(1, y_max)) +  # Log scale on Y-axis
  labs(title = "Top 10 Web Server Types by Total Financial Loss",
       x = "Web Server Type",
       y = "Total Loss (Log Scale, USD)") +
  theme_minimal() +
  theme(panel.grid.major.x = element_line(color = "gray", linetype = "dashed"))  # Keep grid lines for readability
```

```r
# 3. Box Plot - Financial Loss Distribution Across Web Server Types (Log Scale)
ggplot(df_filtered, aes(x = WebServer_Base, y = Loss)) +
  geom_boxplot(fill = "blue", alpha = 0.7, outlier.color = "red", outlier.shape = 16, outlier.size = 1.5) +
  stat_summary(fun = median, geom = "text", aes(label = round(..y.., 2)),
               size = 3, vjust = -0.5, color = "black") +  # Improve median label placement
  coord_flip() +  # Flip for better readability
  scale_y_log10(labels = scales::comma, limits = c(10, max(df_filtered$Loss, na.rm = TRUE) * 1.5)) +  # Adjust log scale range
  labs(title = "Financial Loss Distribution Across Web Server Types",
       x = "Web Server Type",
       y = "Financial Loss (Log Scale, USD)") +
  theme_minimal()
```

This analysis explores the distribution of financial losses across different web server types to determine which are most financially impacted. Using descriptive statistics, data visualization, and log-scale transformations, it provides a clearer understanding of financial loss distribution. A log-scale bar chart visualizes total financial loss per web server type, while a box plot examines the variability and distribution of financial losses.



Apache experiences the highest financial loss (~$332 million), followed by Microsoft IIS ($121 million), Unknown server types ($59.6 million), and Nginx ($19 million. The high losses among Unknown server types suggest inadequate security tracking. Nginx's lower loss may indicate stronger security features or reduced exposure to costly attacks. Other servers like LiteSpeed, Zeus, and Lighttpd contribute to financial losses but on a smaller scale.



The box plot illustrates the financial loss distribution across various web server types on a logarithmic scale, highlighting significant variations. Web servers such as Apache, IIS, and Oracle Server exhibit the highest variability in financial losses, with wide interquartile ranges and multiple outliers. The presence of numerous extreme values suggests that certain defacement incidents result in exceptionally high financial damage. Notably, Oracle Server stands out with a particularly broad distribution, indicating potential security vulnerabilities or frequent targeting.

3.2 **Lara Dominique Isidro Castro, TP072345 —** Objective 2: To investigate the influence of server operating systems and web server types on hacking vulnerability and financial loss

Analysis 2-1: Which server operating systems are most commonly associated with hacking incidents?

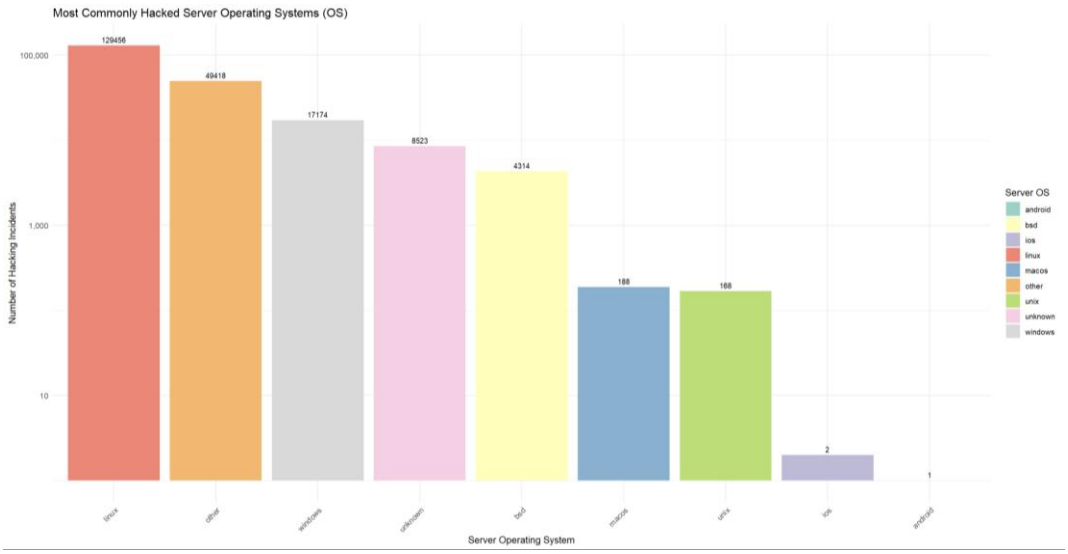
A descriptive analysis was conducted to count the number of hacking incidents for each OS. The results were sorted in descending order and visualized using a bar chart with a logarithmic scale to the y-axis for better differentiation of lower-frequency OS types.

```
# Analysis 2-1: Which server operating systems are most commonly associated with hacking incidents?

# Counts the number of hacking incidents for each server OS
os_counts <- df_filtered %>%
  count(OS_Base, sort = TRUE)

# Prints the OS counts in descending order
print(os_counts)

# Plots the most commonly hacked server OS using a bar chart
ggplot(os_counts, aes(x = reorder(OS_Base, -n), y = n, fill = OS_Base)) +
  geom_bar(stat = "identity") +
  labs(title = "Most Commonly Hacked Server Operating Systems (OS)", fill = "Server OS",
       x = "Server Operating System", y = "Number of Hacking Incidents") +
  scale_y_log10(labels = scales::comma) +                    # Uses a logarithmic scale for the y-axis for better visualization of results
  scale_fill_brewer(palette = "Set3") +
  theme_minimal() +                                          # Changes the plot theme for a cleaner and more modern look
  theme(axis.text.x = element_text(angle = 45, hjust = 1))   # Rotates x-axis labels for better readability
```

Linux is the most frequently hacked OS, with 126,456 incidents. Its wide adoption and potential vulnerabilities in third-party applications make it a prime target (Kidman, 2004). The “Other” category follows with 49,418 incidents, indicating that many attacks also occur on niche operating systems. Windows has 17,174 recorded incidents, reflecting its widespread use in many businesses. The “Unknown” category accounts for 8,523 incidents, suggesting that OS types often remain unidentified due to data classification gaps. BSD and macOS are targeted at much lower rates, with 4,314 and 188 incidents, respectively. Unix follows closely behind with 168 incidents.

Mobile platforms like iOS and Android have the fewest recorded hacking incidents, with only 2 and 1 attacks, respectively. This suggests they are either not commonly used as server OS or have stronger security implementations that make them less susceptible to large-scale attacks in a server environment.

Overall, the data highlights Linux and Windows as the primary targets for cyberattacks, with significant occurrences also in the “Other” and “Unknown” categories, suggesting the need for stronger security in widely used and poorly categorized OS types.



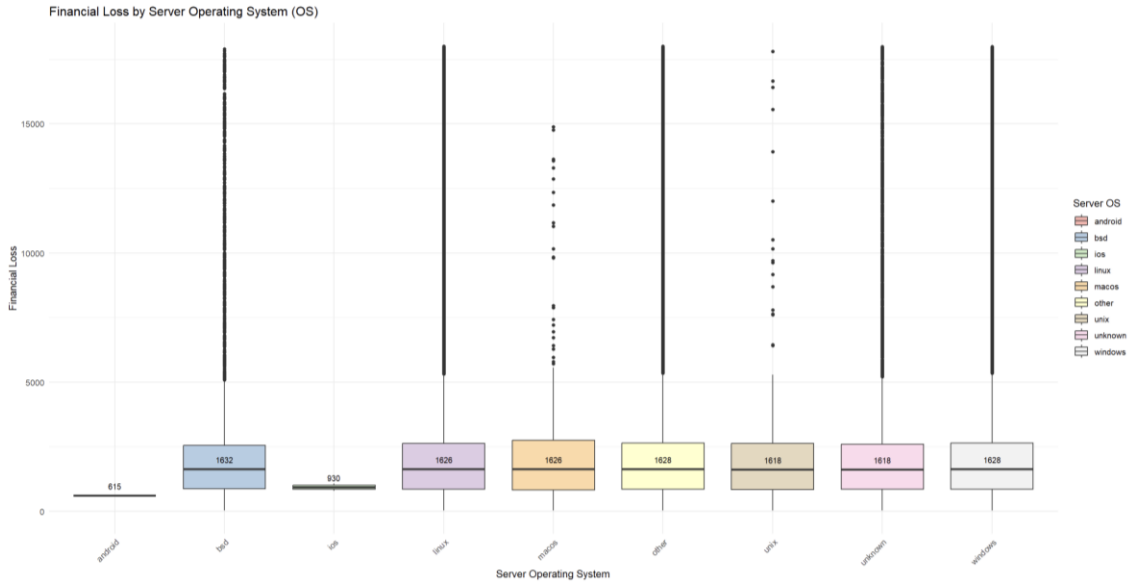
Analysis 2-2: Is there a significant correlation between specific server operating systems and financial loss?

A descriptive statistical analysis was performed to compute the average financial loss for each OS type. Missing and non-finite values in the Loss column were checked and removed to ensure data integrity. The cleaned data was used to generate a boxplot for comparing financial loss distributions among different OS types, showing the median, interquartile range (IQR), and outliers.

```
# Analysis 2-2: Is there a significant correlation between specific server operating systems and financial loss?

# Computes the average financial loss for each server OS
os_loss <- df_filtered %>%
  group_by(OS_Base) %>%
  summarise(Average_Loss = mean(Loss, na.rm = TRUE)) %>%
  arrange(desc(Average_Loss))

# Prints the average financial loss for each server OS
print(os_loss)

# Checks for missing or non-finite values in the "Loss" column
sum(is.na(df_filtered$Loss))
sum(!is.finite(df_filtered$Loss)) # Keeps only rows with finite Loss values

# Filters data to exclude rows with non-finite financial losses
df_filtered_clean <- df_filtered %>%
  filter(is.finite(Loss))  # Keeps only rows with finite Loss values

# Shows the distribution of financial losses for each OS using boxplots
ggplot(df_filtered_clean, aes(x = OS_Base, y = Loss, fill = OS_Base)) +
  geom_boxplot() +
  geom_text(stat = "summary", fun = median, aes(label = round(..y.., 1)), color = "black", size = 3, vjust = -1) +
  labs(title = "Financial Loss by Server Operating System (OS)", fill = "Server OS",
       x = "Server Operating System", y = "Financial Loss") +
  scale_fill_brewer(palette = "Pastel1") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

All server OS types, except Android and iOS, have similar median financial losses, ranging from 1618 to 1632, with BSD having the highest median at 1632. The presence of outliers and long whiskers across most OS types indicates large variability in financial loss. Android and iOS exhibit much lower financial losses, with median values of 615 and 930, respectively, suggesting smaller financial impacts from breaches on mobile-based servers.

Overall, the data suggests that financial losses from cyberattacks are generally consistent across major server OS types, with many high-loss outliers. BSD experiences slightly higher median losses, while mobile-server OS Android and iOS are far less financially impacted than traditional server operating systems.



Analysis 2-3: How does the distribution of downtime vary across different server operating systems?

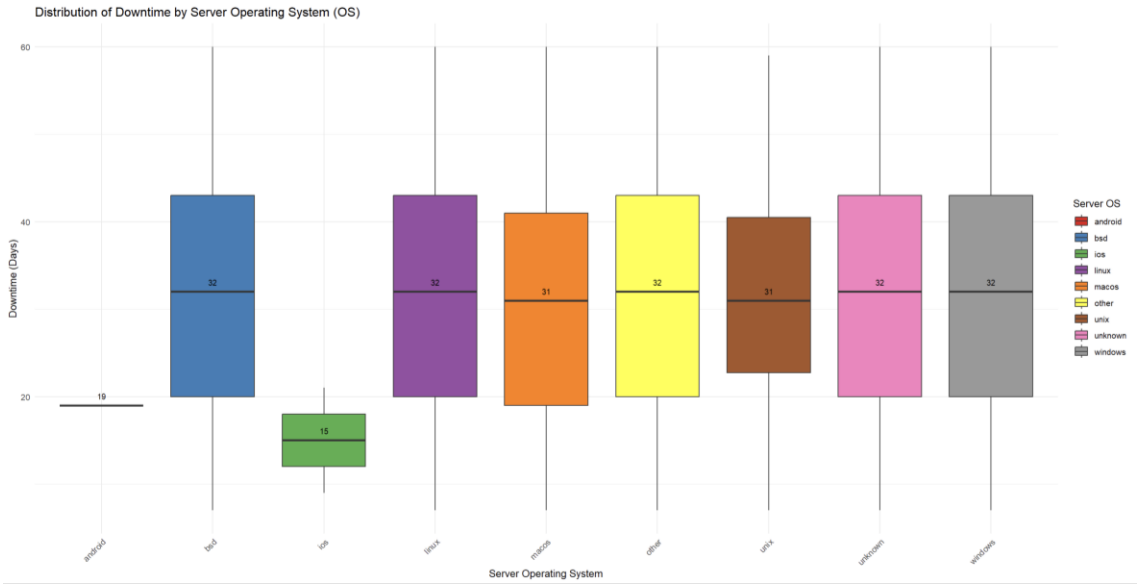
A descriptive statistical analysis was conducted to calculate the average, median, and maximum downtime for each OS type, arranged in descending order. A boxplot was created to visualize the range of downtime for each OS, showing the median, interquartile range (IQR), and whiskers for variability.

```
# Analysis 2-3: How does the distribution of downtime vary across different server operating systems?

# Computes the average downtime for each server operating system
os_downtime <- df_filtered %>%
  group_by(OS_Base) %>%
  summarise(Average_Downtime = mean(DownTime, na.rm = TRUE),
            Median_Downtime = median(DownTime, na.rm = TRUE),
            Max_Downtime = max(DownTime, na.rm = TRUE)) %>%
  arrange(desc(Average_Downtime))

# Prints the average, median, and max downtime for each server OS
print(os_downtime)

# Plots the distribution of downtime for each server OS using boxplots
ggplot(df_filtered, aes(x = OS_Base, y = DownTime, fill = OS_Base)) +
  geom_boxplot() +
  geom_text(stat = "summary", fun = median, aes(label = round(..y.., 1)), color = "black", size = 3, vjust = -1) +
  labs(title = "Distribution of Downtime by Server Operating System (OS)", fill = "Server OS",
       x = "Server Operating System", y = "Downtime (Days)") +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

All server OS types, except Android and iOS, exhibit a median downtime of around 31 to 32 days, indicating approximately a month of server unavailability after hacking incidents regardless of the OS. BSD, Linux, and Windows have a large spread of downtime, with some incidents leading to up to 60 days.

Android has a median downtime of 19 days, while iOS has the lowest median downtime at 15 days. This suggests that mobile platforms tend to recover more quickly from hacking incidents than traditional server OS.

Overall, most traditional server OS types experience similar downtime (about 30 days), while mobile platforms have shorter downtimes, likely due to different use cases and server architectures. The high variation in downtime for BSD, Linux, and Windows implies more complex recovery processes for these systems, leading to prolonged disruptions.



Analysis 2-4: What is the variability in financial loss associated with different web server types?

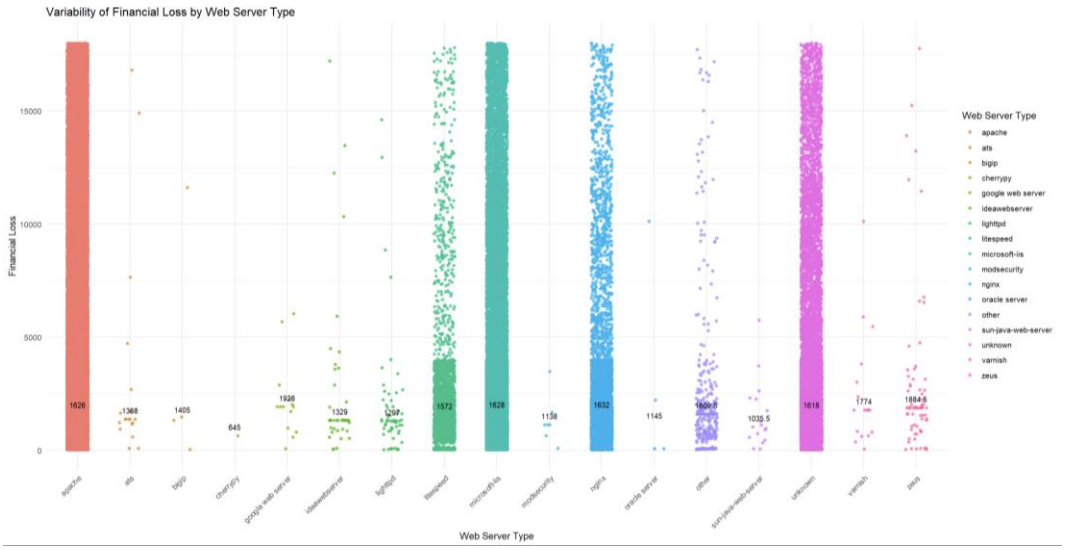
Summary statistics, including the average, median, maximum, and standard deviation of financial were computed and visualized using a scatter plot with jittering to prevent data points from overlapping.

```r
# Analysis 2-4: What is the variability in financial loss associated with different web server types?

# Computes summary statistics (average, median, max, and standard deviation) for financial losses by web server type
web_loss_variability <- df_filtered_clean %>%
  group_by(WebServer_Base) %>%
  summarise(Average_Loss = mean(Loss, na.rm = TRUE),
            Median_Loss = median(Loss, na.rm = TRUE),
            Max_Loss = max(Loss, na.rm = TRUE),
            SD_Loss = sd(Loss, na.rm = TRUE)) %>%
  arrange(desc(Average_Loss))

# Prints the summary statistics for financial loss by web server type
print(web_loss_variability)

# Plots the distribution of financial losses for each web server type using a scatter plot
ggplot(df_filtered_clean, aes(x = WebServer_Base, y = Loss, color = WebServer_Base)) +
  geom_jitter(width = 0.2, height = 0, alpha = 0.7) +  # Adds jitter to avoid overlapping points
  geom_text(stat = "summary", fun = median, aes(label = round(..y.., 1)), color = "black", size = 3, vjust = -1) +
  labs(title = "Variability of Financial Loss by Web Server Type", color = "Web Server Type",
       x = "Web Server Type", y = "Financial Loss") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Apache, Microsoft-IIS, and the “Unknown” category show the most condensed financial loss distributions, with data points ranging from 0 to over 15,000 units. This indicates frequent and highly variable financial impacts. LiteSpeed and Nginx have moderately dense distributions, with losses mainly between 0 and about 4,000 units, highlighting the need for robust security measures for these servers.

Zeus and Varnish have noticeably fewer data points, indicating fewer recorded incidents. However, their incidents tend to involve higher financial losses on average, as shown by their higher median values. This may reflect specialized use cases or specific vulnerabilities.

Overall, Apache, Microsoft-IIS, and "Unknown" servers face the highest frequency and variability in financial losses. LiteSpeed and Nginx experience less severe but frequent incidents. Although Zeus and Varnish are less commonly targeted, they tend to experience higher financial consequences per incident. These insights emphasize the need for tailored security measures, especially for widely used servers like Apache and Microsoft-IIS, to mitigate potential financial losses from breaches.



Analysis 2-5: Do certain server OS and web server combinations show a higher risk of hacking vulnerability?

A frequency analysis was conducted to determine the vulnerability of specific server OS and web server combinations to hacking. The results were visualized using a heatmap, with each tile representing the number of incidents for a given OS-Web Server combination.

```
# Analysis 2-5: Do certain server OS and web server combinations show a higher risk of hacking vulnerability?

# Computes the frequency of hacking incidents for each combination of server OS and web server type
os_web_counts <- df_filtered %>%
  count(OS_Base, WebServer_Base, sort = TRUE)

# Prints the frequency of hacking incidents for each OS-Web Server combination
print(os_web_counts)

# Visualizes the frequency of hacking incidents for each OS-Web Server combination using a heatmap
ggplot(os_web_counts, aes(x = OS_Base, y = WebServer_Base, fill = n)) +
  geom_tile() +   # Each cell represents the frequency of incidents for a combination
  geom_text(aes(label = n), color = "white", size = 3) +
  labs(title = "Hacking Incidents by Server OS and Web Server", fill = "Number of Incidents",
       x = "Server OS", y = "Web Server Type") +
  scale_fill_gradientn(colors = brewer.pal(11, "RdYlGn")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

The Apache and Linux combination stands out with 121,439 incidents, likely due to the widespread use of Linux-based Apache servers. Similarly, Microsoft-IIS with Windows shows 29,547 incidents, reflecting its prevalence in business environments. Nginx paired with Linux has 4,646 incidents, highlighting its growing popularity and potential vulnerabilities. In contrast, web servers like Zeus, Varnish, and CherryPy have minimal incidents, possibly due to limited deployment or niche use. For example, Varnish with Android records only a single incident, indicating low risk or limited adoption.

The heatmap also shows significant activity in the “Unknown” OS category, particularly with web servers like Nginx (17,330 incidents). This could indicate gaps in data classification or diverse attack surfaces. Proper identification and categorization of these setups are crucial for accurate trend analysis and vulnerability assessments.

Combinations like Oracle Server with various OS types have very few incidents, suggesting either robust security measures or limited deployment. Meanwhile, less common configurations like Litespeed and Linux, with 3,015 incidents, point to vulnerabilities in smaller but still significant user bases.

Overall, widely used combinations like Apache-Linux and Microsoft-IIS-Windows are the most vulnerable to hacking, while newer combinations like Nginx-Linux require attention due to

emerging risks. This highlights the need for strong security measures and improved classification of unknown systems to better understand hacking trends and mitigate risks.

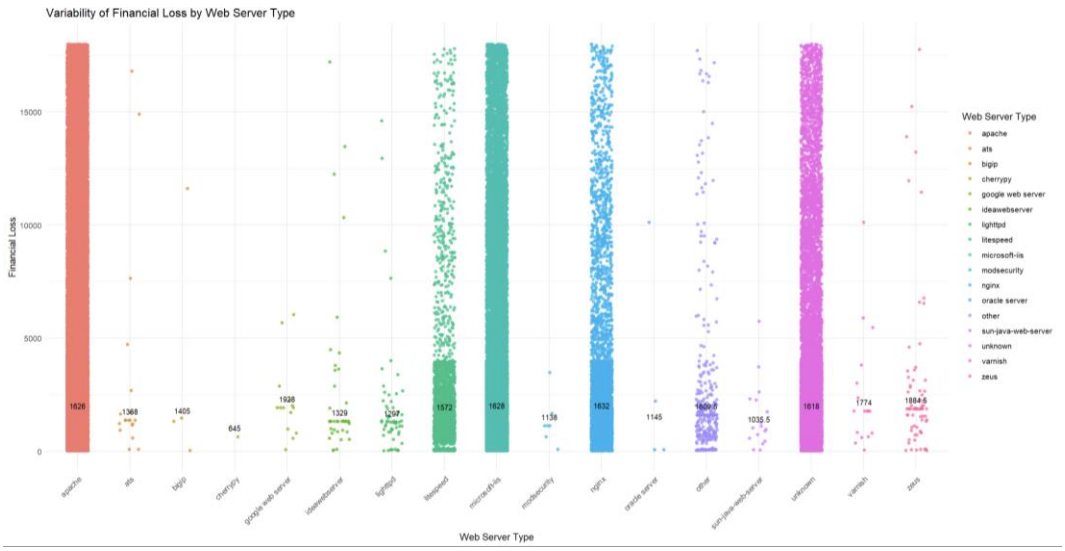
In conclusion, this analysis reveals that server OS and web server configurations significantly impact hacking vulnerability and associated financial losses, emphasizing the importance of OS-web server security considerations in mitigating hacking risks and financial damage.

Extra Features

```
# Extra Feature 1: Using geom_jitter() instead of geom_point() for the Scatter Plot
# The use of geom_jitter() in Analysis 2-4 enhances visualization by introducing random noise (jitter) to overlapping points. It
# ensures that points with similar financial losses do not overlap and are easier to interpret.

# Extra Feature 2: Using a Heatmap (geom_tile) for OS-Web Server Combinations
# The heatmap visualization using `geom_tile()` in Analysis 2-5 allows for a comparative analysis of hacking incidents across
# combinations of server operating systems and web server types. The heatmap provides a quick visual reference to identify high-risk
# combinations, enhancing interpretability and aiding decision-makers in prioritizing security efforts.
```

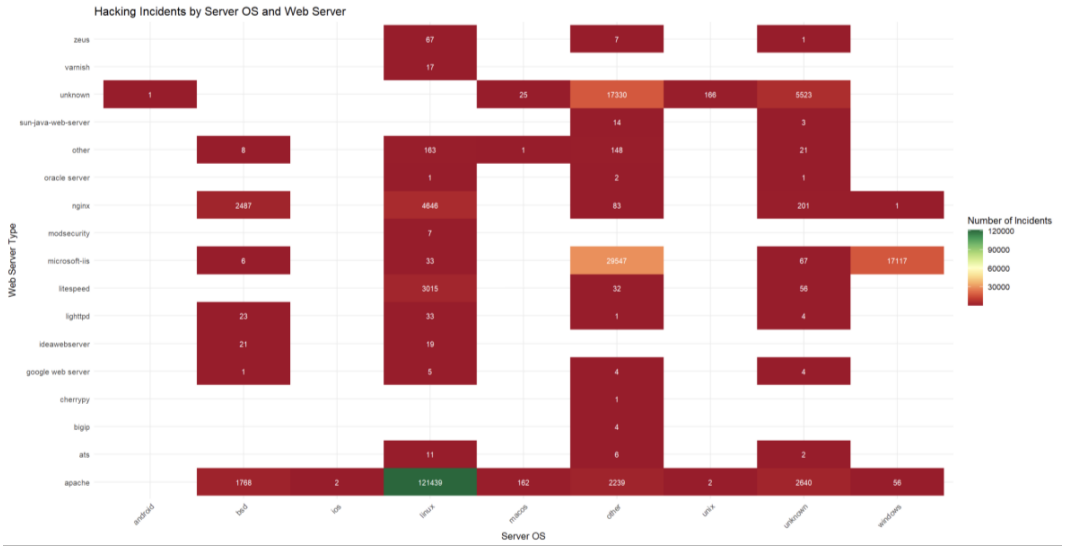
1. In Analysis 2-4, geom_jitter() was used instead of geom_point() to prevent overlapping data points in the scatter plot. This added slight random variation in point placement, ensuring clusters of similar financial loss values were visible and improving readability for a more accurate interpretation of financial loss distribution across web server types.

```
# Plots the distribution of financial losses for each web server type using a scatter plot
ggplot(df_filtered_clean, aes(x = WebServer_Base, y = Loss, color = WebServer_Base)) +
  geom_jitter(width = 0.2, height = 0, alpha = 0.7) +  # Adds jitter to avoid overlapping points
  geom_text(stat = "summary", fun = median, aes(label = round(..y.., 1)), color = "black", size = 3, vjust = -1) +
  labs(title = "Variability of Financial Loss by Web Server Type", color = "Web Server Type",
       x = "Web Server Type", y = "Financial Loss") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



2. In Analysis 2-5, a heatmap using geom_tile() visualized hacking incident frequencies for each OS-Web Server combination, highlighting high-risk configurations. Labels with exact incident counts improved clarity, helping analysts quickly identify server combinations needing immediate attention.

```r
# Visualizes the frequency of hacking incidents for each OS-Web Server combination using a heatmap
ggplot(os_web_counts, aes(x = OS_Base, y = WebServer_Base, fill = n)) +
  geom_tile() +   # Each cell represents the frequency of incidents for a combination
  geom_text(aes(label = n), color = "white", size = 3) +
  labs(title = "Hacking Incidents by Server OS and Web Server", fill = "Number of Incidents",
       x = "Server OS", y = "Web Server Type") +
  scale_fill_gradientn(colors = brewer.pal(11, "RdYlGn")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



3.3 **Narina Kaur Sidhu, TP079398** — Objective 3: To investigate the relationship between ransom demands and encoding methods on financial loss.

Analysis 3-1: What is the relationship between ransom demands and financial loss?

This analysis examines the relationship between ransom amounts and corresponding financial losses in ransomware attacks, focusing on whether higher ransom demands are associated with greater financial losses. Categorising ransom amounts into low, medium, and high tiers simplifies result interpretation and ensures that each category is adequately represented, even in the presence of uneven distribution. This approach provides clearer insights into the distribution of losses across the different ransom categories.

```
# Categorizing Ransom Amounts
df_filtered <- df_filtered %>%
  mutate(ransom_category = cut(
    Ransom,
    breaks = c(-Inf, 1546, 2300, Inf),
    labels = c("Small", "Medium", "High")
  ))
```

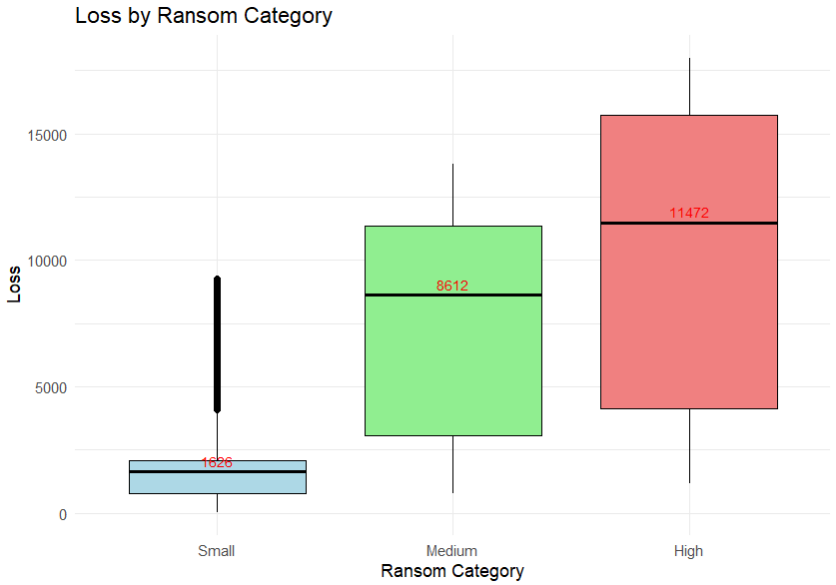
The summary statistics of ransom are:

- 1546: All quantiles share this value
- 2300: This value falls between the third quartile and the maximum.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 100    1546    1546    1546    1546    3000
```

A box plot was created to show the distribution of 'loss across ransom categories.', highlighting the median, range, and variations in losses within each group. It compares the ransom categories and their respective loss distributions, where the length indicates the variation in financial loss within each category:

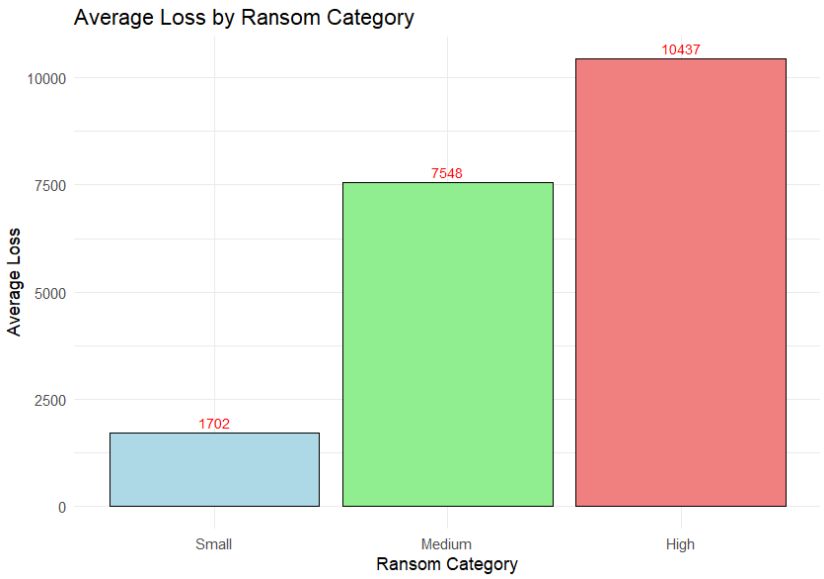
```r
# Boxplot of Loss by Ransom Category
ggplot(df_filtered, aes(x = ransom_category, y = Loss, fill = ransom_category)) +
  geom_boxplot(color = "black") +
  stat_summary(fun = median, geom = "text", aes(label = round(after_stat(y), 0)),
               color = "red", vjust = -0.5, size = 3) +
  labs(title = "Loss by Ransom Category",
       x = "Ransom Category",
       y = "Loss") +
  scale_fill_manual(values = c("Small" = "lightblue", "Medium" = "lightgreen", "High" = "lightcoral")) +
  theme_minimal() +
  theme(legend.position = "none")
```



- Low ransom category: Median loss is 1625.
- Medium ransom category: Median loss is 8612.
- High ransom category: Median loss is 11,486.

To further validate the results, a bar plot was generated to show the average loss for each ransom category, offering a clearer comparison of how losses fluctuate with different ransom amounts. This visual allows for straightforward analysis of loss trends over varying ransom amounts:

```r
# Bar Chart of Average Loss by Ransom Category
print(
  ggplot(df_filtered, aes(x = ransom_category, y = Loss, fill = ransom_category)) +
    stat_summary(fun = mean, geom = "bar", color = "black") +
    stat_summary(fun = mean, geom = "text", aes(label = round(after_stat(y), 0)),
                 color = "red", vjust = -0.5, size = 3) +
    labs(title = "Average Loss by Ransom Category",
         x = "Ransom Category",
         y = "Average Loss") +
    scale_fill_manual(values = c("Small" = "lightblue", "Medium" = "lightgreen", "High" = "lightcoral")) +
    theme_minimal() +
    theme(legend.position = "none")
)
```



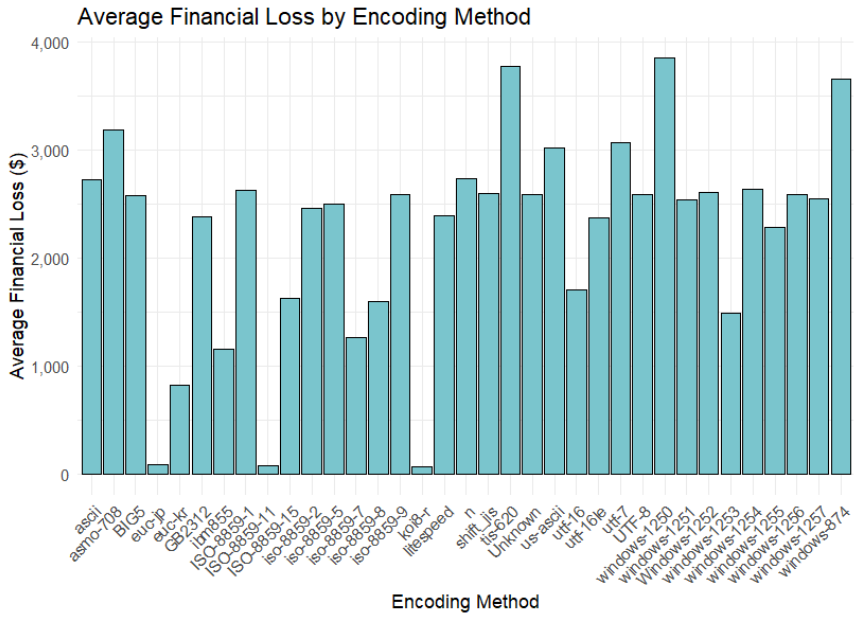
- Low ransom category: Mean loss is 1702.
- Medium ransom category: Mean loss is 7551.
- High ransom category: Mean loss is 10,444.

In conclusion, both the box and bar plots demonstrate that as ransom categories progress from low to medium to high, the median and mean losses increase. This indicates a positive relationship between ‘Ransom’ and ‘Loss,’ confirming that higher ransom demands lead to higher financial losses. The trend shown by these visualisations validates the idea that ransom size plays a significant role in predicting financial impact.

Analysis 3-2: Do specific encoding methods result in higher financial losses?

This analysis examines the relationship between encoding methods and financial losses, aiming to determine whether specific encoding methods are associated with significantly higher financial losses.

```
# Bar plot for all encoding methods
ggplot(df_filtered, aes(x = Encoding, y = Loss)) +
  stat_summary(fun = "mean", geom = "bar", fill = "cadetblue3", color = "black") +
  labs(
    title = "Average Financial Loss by Encoding Method",
    x = "Encoding Method",
    y = "Average Financial Loss ($)"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +  # Rotate x-axis labels
  scale_y_continuous(labels = scales::comma)  # Format y-axis labels
```

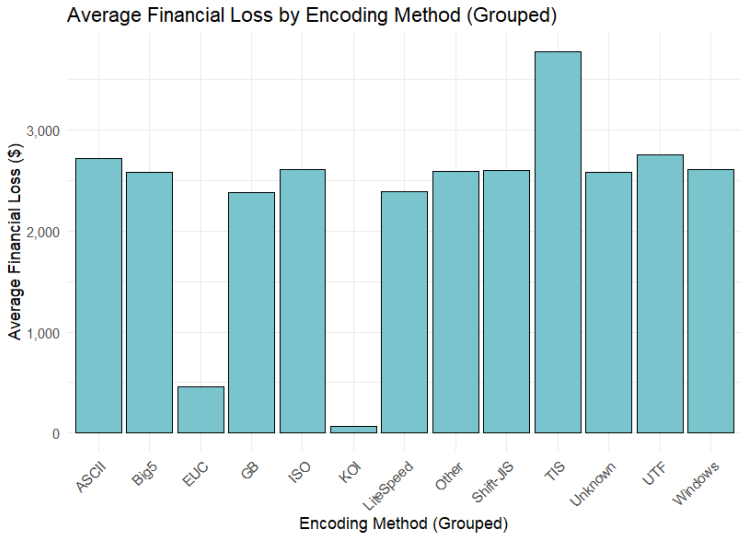


The bar plot above presents the average financial losses for every individual encoding method. It revealed indistinct patterns due to some encoding types having similar mean values.

Analysing individual encoding methods alone may not fully capture broader patterns. To address this, encoding methods were grouped into broader categories to enhance clarity and reduce complexity.

```r
# Group encoding methods into broader categories
df_filtered <- df_filtered %>%
  mutate(Encoding_Grouped = case_when(
    str_detect(Encoding, "windows") ~ "Windows",
    str_detect(Encoding, "utf") ~ "UTF",
    str_detect(Encoding, "iso|ISO") ~ "ISO",
    str_detect(Encoding, "ascii") ~ "ASCII",
    str_detect(Encoding, "big5|BIG5") ~ "Big5",
    str_detect(Encoding, "euc") ~ "EUC",
    str_detect(Encoding, "gb2312|GB2312") ~ "GB",
    str_detect(Encoding, "koi8-r") ~ "KOI",
    str_detect(Encoding, "tis|TIS") ~ "TIS",
    str_detect(Encoding, "shift_jis") ~ "Shift-JIS",
    str_detect(Encoding, "litespeed") ~ "LiteSpeed",
    str_detect(Encoding, "Unknown") ~ "Unknown",
    TRUE ~ "Other"
  ))
```

Various encoding methods are categorized under broader groups such as "Windows", "UTF", and "ISO", while other categories like "TIS", "Shift-JIS", and "KOI" are also established. Encoding methods that do not fit predefined categories are labeled as "Unknown" or "Other" to maintain data consistency.



The bar plot above consolidates encoding methods into broader categories. From the graph, it is evident that some encoding methods are associated with significantly higher financial losses than others. Encoding methods like 'TIS' and 'ASCII' are linked to higher average financial losses, whereas 'EUC' and 'KOI' show lower average losses.

In conclusion, the variation in mean financial loss across encoding types is minimal, indicating a weak relationship between encoding method and financial loss. Therefore, while

some encoding methods may show a slight association with higher losses, the encoding type does not appear to be a significant predictor of financial loss in this context.

Analysis 3-3: How does the frequency of different encoding methods vary across ransom attacks?

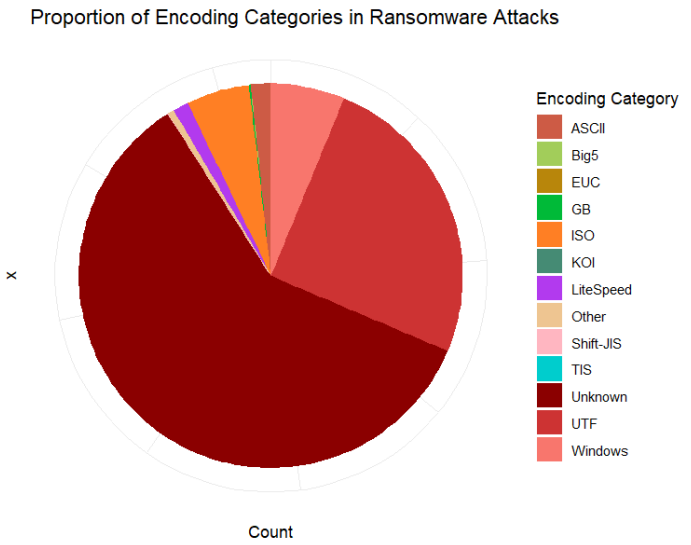
This analysis investigates which encoding methods are most frequently used in ransomware attacks and whether certain encoding methods are dominant. Understanding the distribution of encoding methods can provide insights into potential risk factors that could contribute to financial losses.

```
# Count the frequency of each encoding category
encoding_grouped_frequency <- df_filtered %>%
  group_by(Encoding_Grouped) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  mutate(Percentage = Count / sum(Count) * 100,  # Calculate percentage
         Label = paste0(Encoding_Grouped, "\n", Count, " (", round(Percentage, 1), "%)"))  # Create label text

# Display the frequency table
print(encoding_grouped_frequency, n = 13)
```

```
# A tibble: 13 × 4
   Encoding_Grouped  Count Percentage Label
   <chr>             <int>      <dbl> <chr>
 1 Unknown          124710  59.6      "Unknown\n124710 (59.6%)"
 2 UTF               52626  25.2      "UTF\n52626 (25.2%)"
 3 Windows           13033   6.23     "Windows\n13033 (6.2%)"
 4 ISO               11151   5.33     "ISO\n11151 (5.3%)"
 5 ASCII              3483   1.66     "ASCII\n3483 (1.7%)"
 6 LiteSpeed          2700   1.29     "LiteSpeed\n2700 (1.3%)"
 7 Other              1195   0.571    "Other\n1195 (0.6%)"
 8 GB                  254   0.121    "GB\n254 (0.1%)"
 9 Big5                 71   0.0339   "Big5\n71 (0%)"
10 Shift-JIS            13   0.00621  "Shift-JIS\n13 (0%)"
11 TIS                   5   0.00239  "TIS\n5 (0%)"
12 EUC                   2   0.000956 "EUC\n2 (0%)"
13 KOI                   1   0.000478 "KOI\n1 (0%)"
```

Most of the attacks used the “Unknown” encoding category (59.6%), which suggests that many encoding types were either unidentified or not categorised. The “UTF” category follows with 25.2%, indicating it is the most common identifiable encoding type used. Other encoding methods, such as “Windows” (6.2%) and “ISO” (5.3%), represent smaller yet significant proportions. Less frequently used encoding types include “ASCII” (1.7%) and “LiteSpeed” (1.3%), while categories like “TIS,” “EUC,” and “KOI” represent a negligible proportion of attacks.

```r
# Visualize the frequency distribution for grouped encoding categories using a pie chart
ggplot(encoding_grouped_frequency, aes(x = "", y = Count, fill = Encoding_Grouped)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +  # Convert to pie chart
  labs(
    title = "Proportion of Encoding Categories in Ransomware Attacks",
    fill = "Encoding Category"
  ) +
  scale_fill_manual(
    values = c(
      "Windows" = "#F8766D", "UTF" = "brown3", "ISO" = "chocolate1",
      "ASCII" = "coral3", "Big5" = "darkolivegreen3", "EUC" = "darkgoldenrod",
      "GB" = "#00BA38", "KOI" = "aquamarine4", "TIS" = "cyan3",
      "Shift-JIS" = "lightpink", "LiteSpeed" = "darkorchid2",
      "Unknown" = "darkred", "Other" = "burlywood2"
    )
  ) +
  theme_minimal() +
  theme(axis.text.x = element_blank())  # Remove x-axis labels for pie chart
```



The pie chart displays the proportion of encoding categories used in ransomware attacks. In conclusion, the frequency distribution suggests that certain encoding methods, such as 'UTF,' 'Windows,' and 'ISO,' are prevalent in ransomware attacks. These methods could serve as key indicators of higher financial vulnerability if attackers favour them to maximise their chances of extracting larger ransom payments.

Analysis 3-4: Does ransom amount and encoding method collectively influence the prediction of financial loss?

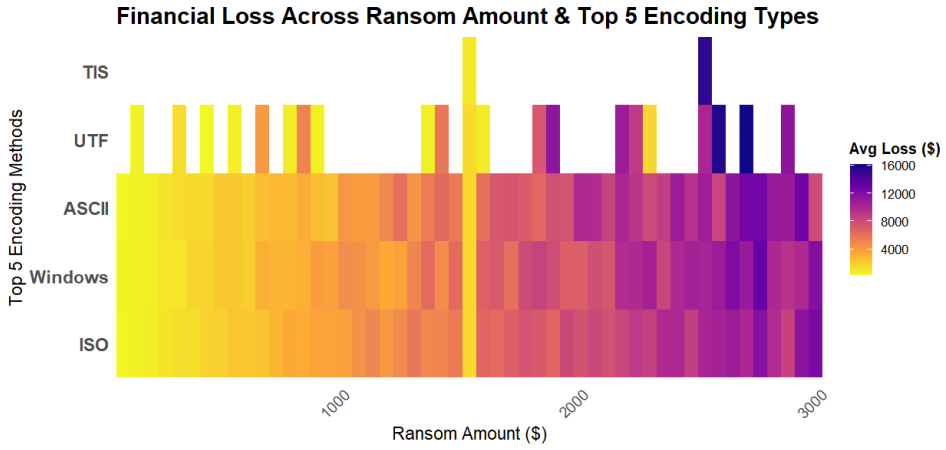
This analysis falls under exploratory data analysis, specifically a comparative analysis aimed at understanding the combined effect of ransom amount and encoding method on financial loss. The goal is to understand whether these two variables together can help estimate the expected loss.

```r
# Heat Map 1: Financial Loss Across Ransom Amount & Encoding Type
ggplot(df_filtered, aes(x = Ransom, y = Encoding_Grouped)) +
  stat_summary_2d(aes(z = Loss), bins = 50) +  # Create 2D bins for heat map
  scale_fill_viridis_c(option = "plasma", direction = -1) +  # Use plasma color scale
  labs(
    title = "Financial Loss Across Ransom Amount & Encoding Type",
    x = "Ransom Amount ($)",
    y = "Character Encoding Type",
    fill = "Avg Loss ($)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.text.y = element_text(size = 14, face = "bold"),  # Bold y-axis labels
    axis.text.x = element_text(size = 12, angle = 45, hjust = 1),  # Rotate x-axis labels
    plot.title = element_text(face = "bold", size = 18),  # Bold title
    legend.title = element_text(size = 12, face = "bold"),  # Bold legend title
    legend.text = element_text(size = 10),  # Legend text size
    panel.grid.major = element_blank(),  # Remove major grid lines
    panel.grid.minor = element_blank()   # Remove minor grid lines
  ) +
  scale_y_discrete(limits = rev(levels(df_filtered$Encoding_Grouped)), drop = TRUE) +  # Reverse y-axis order
  coord_cartesian(expand = FALSE)  # Remove padding around the plot
```

The heatmap above visualises financial loss across all character encoding types and ransom amounts, with the colour gradient effectively highlighting areas of higher or lower financial loss. This heatmap indicates that encoding type plays a relatively small role in determining financial loss compared to the impact of ransom amounts. The clear trend observed is that higher ransom demands lead to greater financial losses, regardless of the encoding method used. This suggests that ransom amount has a more significant influence on financial loss than encoding type. Nevertheless, certain encoding methods, such as 'TIS,' 'Windows,' and 'ASCII,' are closely linked to higher financial losses, as indicated by the darker shades on the heatmap.

```r
# Heat Map 2: Ransom vs Top 5 Encoding Groups (excluding Unknown/Other), colored by Loss
# Identify top 5 encoding groups by average loss (excluding Unknown/Other)
top_encodings <- df_filtered %>%
  filter(!(Encoding_Grouped %in% c("Unknown", "Other"))) %>%  # Exclude "Unknown" and "Other"
  group_by(Encoding_Grouped) %>%
  summarise(avg_loss = mean(Loss, na.rm = TRUE)) %>%
  arrange(desc(avg_loss)) %>%
  slice_head(n = 5) %>%
  pull(Encoding_Grouped)

# Filter dataset for only the top 5 encoding methods
df_filtered_clean <- df_filtered %>%
  filter(Encoding_Grouped %in% top_encodings)
```

```r
# Heat Map 2: Ransom vs Top 5 Encoding Groups
ggplot(df_filtered_clean, aes(x = Ransom, y = Encoding_Grouped)) +
  stat_summary_2d(aes(z = Loss), bins = 50) +  # Create 2D bins for heat map
  scale_fill_viridis_c(option = "plasma", direction = -1) +  # Use plasma color scale
  labs(
    title = "Financial Loss Across Ransom Amount & Top 5 Encoding Types",
    x = "Ransom Amount ($)",
    y = "Top 5 Encoding Methods",
    fill = "Avg Loss ($)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.text.y = element_text(size = 14, face = "bold"),  # Bold y-axis labels
    axis.text.x = element_text(size = 12, angle = 45, hjust = 1),  # Rotate x-axis labels
    plot.title = element_text(face = "bold", size = 18),  # Bold title
    legend.title = element_text(size = 12, face = "bold"),  # Bold legend title
    legend.text = element_text(size = 10),  # Legend text size
    panel.grid.major = element_blank(),  # Remove major grid lines
    panel.grid.minor = element_blank()   # Remove minor grid lines
  ) +
  scale_y_discrete(limits = rev(top_encodings), drop = TRUE) +  # Reverse y-axis order
  coord_cartesian(expand = FALSE)  # Remove padding around the plot
```



The second heat map highlights the top five encoding methods (excluding "Unknown" and "Other") in relation to both ransom amounts and financial losses. It confirms the trend that 'TIS,' 'ASCII,' and 'ISO' are linked to higher financial losses.

In conclusion, while certain encoding methods are associated with higher financial losses, the ransom amount is the primary factor influencing the extent of these losses. The colour gradient, transitioning from yellow to purple, illustrates a clear trend: as ransom demands increase, financial losses also rise, regardless of the encoding method used. Therefore, while encoding type may have some association with financial loss, ransom amount plays a much more significant role in determining the overall financial impact.

3.4 **Azwa Al Islam, TP078098** – Objective 4: To investigate if financial loss is influenced by the amount of ransom paid and downtime experienced.

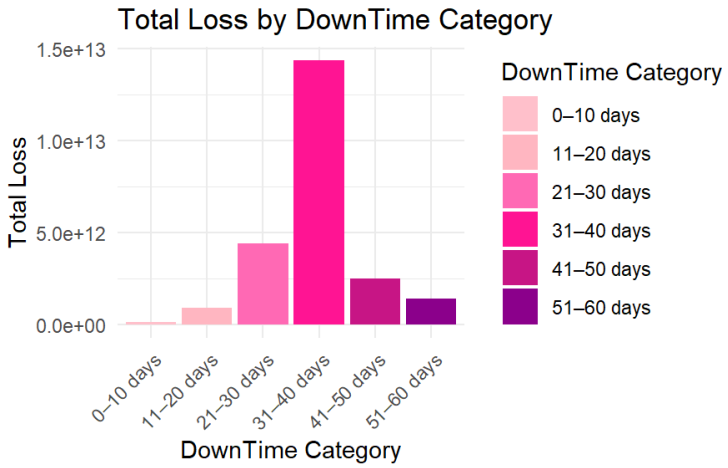
Analysis 4-1: Does an increased downtime cause a larger financial loss?

This analysis categorizes downtime durations into six groups (A-F) based on days of service disruption and examines their financial impact.

```
# Categorizing the Data
df_filtered <- df_filtered %>%
  mutate(downtime_category = cut(
    DownTime,
    breaks = c(-Inf, 10, 20, 30, 40, 50, Inf),
    labels = c("A", "B", "C", "D", "E", "F")
  )) %>%
  group_by(downtime_category) %>%
  mutate(total_loss = sum(Loss, na.rm = TRUE)) %>%
  ungroup()
```

“DownTime’ spans from 7 to 60 days, ensuring an even distribution across 6 categories. The data is processed by creating a new “downtime_category” variable and calculating the total loss for each category. A bar chart is used to visualize the relationship between downtime duration and financial losses.

```
ggplot(df_filtered, aes(x = downtime_category, y = total_loss, fill = downtime_category)) +
  geom_bar(stat = "identity") +
  scale_x_discrete(labels = downtime_labels) +  # Update x-axis labels
  scale_fill_manual(values = c("#FFC0CB", "#FFB6C1", "#FF69B4", "#FF1493", "#C71585", "#8B008B"),
                    labels = downtime_labels, name = "DownTime Category") +  # Update legend
  labs(
    title = "Total Loss by DownTime Category",
    x = "DownTime Category",
    y = "Total Loss"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readabilit
```



The trend reveals that financial losses steadily rise for downtimes lasting up to 40 days, reaching their peak in the 31–40 day category. However, beyond this threshold, losses drop sharply, suggesting that issues are typically resolved within this timeframe. This emphasizes the need to mitigate downtime before it extends into the 31–40 day range.

Ultimately, the findings indicate that prolonged downtime generally leads to higher financial losses, with the most severe impact occurring within the 31–40 day period. The decline in loss beyond this point suggests that intervention measures were taken or that additional factors played a significant role in shaping the distribution of losses.

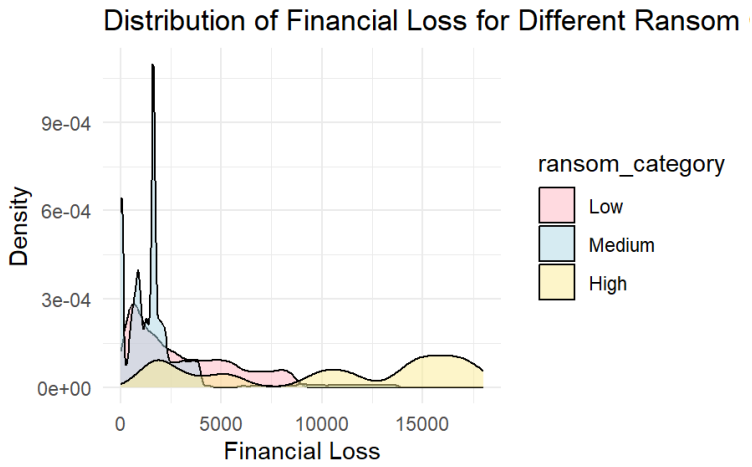
Analysis 4-2: How does the distribution of financial loss vary across different ransom amounts categorized as low, medium, and high?

This analysis classifies ransom demands into three categories—Low, Medium, and High—based on monetary value, then examines their impact on financial loss.

```
#Grouping into Low, Medium, and High
#Low = (-infinite - 1000), medium = (1000-2000), high = (2000 - infinite)
# Categorize Ransom Values Into Low, Medium, and High
df_filtered$ransom_category <- cut(df_filtered$Ransom,
                                  breaks = c(-Inf, 1456, 2300, Inf),
                                  labels = c("Low", "Medium", "High"),
                                  right = FALSE)
```

The dataset is segmented into these categories, ensuring a clear distribution across different ransom levels. Instead of evaluating individual ransom, categorizing ransom demands into Low, Medium, and High allows for a clearer analysis of how different ransom levels impact financial loss. A density plot is used to visualize how financial losses are spread across each category.

```
ggplot(df_filtered, aes(x = Loss, fill = ransom_category)) +
  geom_density(alpha = 0.5) +
  labs(title = "Distribution of Financial Loss for Different Ransom Categories",
       x = "Financial Loss",
       y = "Density") +
  scale_fill_manual(values = c("Low" = "#FFB6C1", "Medium" = "#ADD8E6", "High" = "#FBEC94")) +
  theme_minimal()
```



The findings reveal that ransom demands categorized as ‘Medium’ occur most frequently and are concentrated around lower financial loss values, indicating that these are common cases with minimal impact. In contrast, ‘High’ ransom demands exhibit a broader spread toward higher financial loss values, suggesting a positive correlation—higher ransom demands tend to be associated with greater financial losses.

Overall, the results suggest that while most cases fall within the ‘Medium’ category with relatively small losses, increasing ransom demands generally lead to more severe financial consequences.

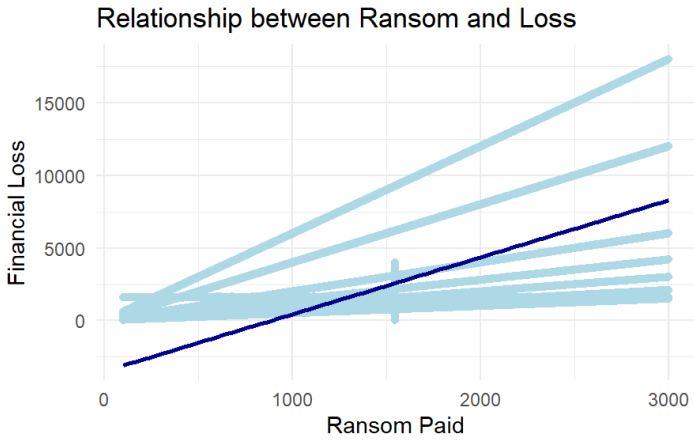
Analysis 4-3: Which factor—ransom paid or downtime—has a greater impact on financial loss?

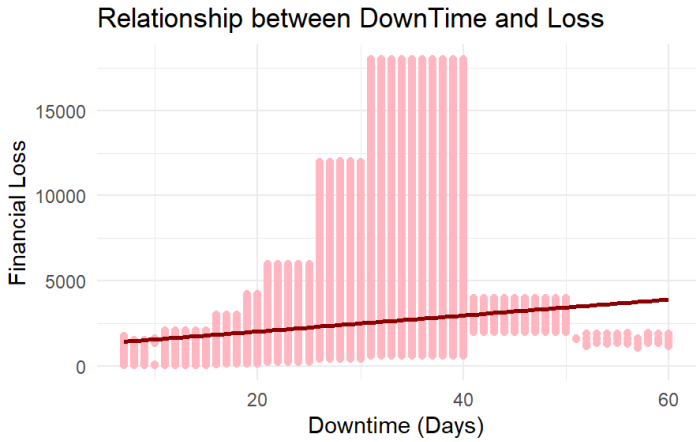
This analysis investigates whether ransom paid or downtime has a stronger influence on financial loss. To do this, Pearson correlation coefficients are calculated for both variables, measuring the strength of their relationships with financial loss.

```
# Calculating correlation between Loss and Ransom
cor_ransom_loss <- cor(df_filtered$Ransom, df_filtered$Loss, use = "complete.obs", method = "pearson")

# Calculating correlation between Loss and DownTime
cor_downtime_loss <- cor(df_filtered$DownTime, df_filtered$Loss, use = "complete.obs", method = "pearson")

# Printing the results
print(paste("Correlation between Ransom and Loss:", cor_ransom_loss))
print(paste("Correlation between DownTime and Loss:", cor_downtime_loss))
```

Additionally, scatter plots with regression lines visually illustrate these relationships, and linear regression models assess the extent to which ransom and downtime explain variations in financial loss.

```
#Visualization With Scatter Plots
# Scatter plot for Ransom vs Loss
ggplot(df_filtered, aes(x = Ransom, y = Loss)) +
  geom_point(color = "lightblue") +
  geom_smooth(method = "lm", color = "darkblue", se = FALSE) +
  labs(title = "Relationship between Ransom and Loss",
       x = "Ransom Paid",
       y = "Financial Loss") +
  theme_minimal()
# Scatter plot for DownTime vs Loss
ggplot(df_filtered, aes(x = DownTime, y = Loss)) +
  geom_point(color = "lightpink") +
  geom_smooth(method = "lm", color = "darkred", se = FALSE) +
  labs(title = "Relationship between DownTime and Loss",
       x = "Downtime (Days)",
       y = "Financial Loss") +
  theme_minimal()
```





From the correlation values, we can see which variable has a stronger association with financial loss. The larger the correlation, the larger the impact, but correlation does not imply causation. Therefore, linear regression models are used to quantify how much financial loss changes as ransom or downtime changes. The regression coefficients indicate the degree of each factor's impact.

```
#Assessing variance in Loss is explained by Ransom and DownTime.
# Linear regression for Ransom effect on Loss
lm_ransom <- lm(Loss ~ Ransom, data = df_filtered)
summary(lm_ransom)

# Linear regression for DownTime effect on Loss
lm_downtime <- lm(Loss ~ DownTime, data = df_filtered)
summary(lm_downtime)
```

```
> # Linear regression for DownTime effect on Loss
> lm_downtime <- lm(Loss ~ DownTime, data = df_filtered)
> summary(lm_downtime)

Call:
lm(formula = Loss ~ DownTime, data = df_filtered)

Residuals:
    Min      1Q  Median      3Q     Max
-2772.2 -1542.0 -1136.1   -95.1 15446.9

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1089.9435    17.0459   63.94   <2e-16 ***
DownTime      47.0041     0.4861   96.69   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3246 on 209242 degrees of freedom
Multiple R-squared:  0.04277,   Adjusted R-squared:  0.04277
F-statistic:  9349 on 1 and 209242 DF,  p-value: < 2.2e-16
```

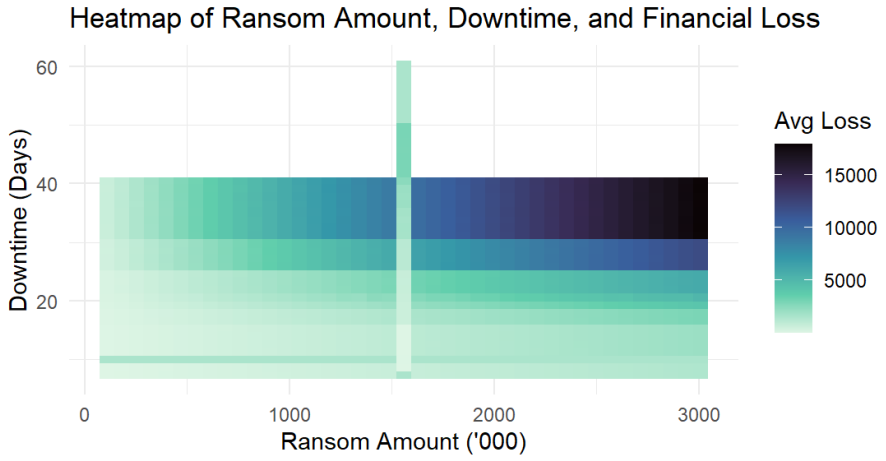
The regression results indicate that the intercept is approximately 1089.94, meaning that even when downtime is zero, there is still a baseline financial loss. The coefficient for downtime is 47.00, suggesting that for each additional day of downtime, financial loss increases by approximately 47.00 units. The p-value (< 2e-16) confirms that this relationship is statistically significant. However, the R-squared value (0.04277) indicates that downtime explains only about 4.3% of the variation in financial loss, suggesting that other factors also play a major role in determining financial losses. Thus, we can conclude Down-Time has a greater impact.

Analysis 4-4: Can financial loss be accurately predicted by combining the ransom amount and downtime?

A heatmap is used to visualize how ransom and downtime together influence financial loss.

```
#Heatmap to comprehend the impact of the combination of 'Ransom' and 'DownTime' on 'Loss'.
ggplot(df_filtered, aes(x = Ransom, y = DownTime)) +
  stat_summary_2d(aes(z = Loss), bins = 40) +
  scale_fill_viridis_c(option = "mako", direction = -1) +
  labs(title = "Heatmap of Ransom Amount, Downtime, and Financial Loss",
       x = "Ransom Amount ('000)",
       y = "Downtime (Days)",
       fill = "Avg Loss") +
  theme_minimal()
```

Darker shades indicate higher loss, while lighter shades represent lower loss, making it easier to spot patterns without focusing on exact numbers.



The results show that higher ransom amounts and longer downtimes lead to greater financial loss. The lowest losses occur when ransom is below 1,000 and downtime is under 20 days, while the highest losses are seen when ransom exceeds 2,000 and downtime surpasses 20 days. A noticeable concentration around the 1,500 ransom mark suggests a potential outlier or varying downtime effects. Most cases involve downtime below 40 days, with fewer instances of extreme delays. The highest losses are linked to ransom amounts between 2,500–3,000 and downtimes of 30–50 days, while the lowest losses occur when ransom is below 500 and downtime is between 0–10 days. Thus, the answer to our question is yes—ransom and downtime effectively predict financial loss on average, confirming that both factors are statistically significant.

4.0 Conclusion

The findings of this analysis support the hypothesis that organisations utilising certain web server versions, operating on specific server operating systems, and using web server types are more likely to experience higher financial losses due to defacement incidents. The results indicate that these factors contribute to increased hacking vulnerability, with widely used web servers and configurations being frequent targets. Additionally, higher ransom demands are associated with greater financial losses, and downtime further amplifies financial damage. The combined influence of ransom payments and service disruptions reinforces the significance of these variables in predicting financial loss, emphasising the need for proactive security measures.

4.1 Discussion on Findings

The analysis of Objective 1 identifies significant financial losses concentrated among widely used web servers, with Apache incurring the highest cumulative loss, representing over 60% of total recorded financial damages. Microsoft IIS also demonstrates substantial financial vulnerabilities, suggesting that attackers strategically target these platforms due to their prevalence and known security weaknesses. Moving on, the analysis for Objective 2 concluded that server OS and web server configurations significantly impact financial loss, emphasizing the importance of robust security considerations to mitigate risks and reduce financial damage. Next, the findings for Objective 3 reveal that higher ransom demands consistently lead to greater financial losses, with both median and mean values increasing across low, medium, and high ransom categories. Although some encoding methods, such as 'TIS' and 'ASCII,' are associated with higher losses, the impact of encoding type is minimal compared to ransom amount, which remains the most significant predictor of financial loss. Finally, the analysis for Objective 4 confirms that both ransoms paid, and downtime influence financial loss, with downtime having a greater impact. Losses peak at 31–40 days before declining, likely due to mitigation efforts. High ransom demands correlate with greater financial loss, while medium demands are most common but less severe. Combining ransom and downtime improves loss prediction, reinforcing their significance. These insights highlight the need for proactive cybersecurity measures.

4.2 Recommendations

Stakeholders should secure high-risk servers with regular patching, intrusion detection, and strengthened OS security. Ransomware mitigation through incident response plans, backups, and disaster recovery is vital for minimising financial damage. Organisations should adopt risk-based cybersecurity frameworks, conduct audits, use AI-driven threat monitoring, and comply with standards like ISO 27001 and NIST to prevent defacement, cyberattacks, and financial losses.

4.3 Limitations and Future Directions

The analyses are limited by incomplete data, simplified categorization, low R-squared values, and dependence on reported data, which may overlook some hacking incidents. To better understand cyberattack effects, future studies should use broader data sources, comprehensive datasets, and advanced modelling techniques.

(5745 words only)

5.0 Workload Matrix

Tasks	Azwa Al Islam (TP078098)	Lara Dominique Isidro Castro (TP072345)	Narina Kaur Sidhu (TP079398)	Ryan Yew Khy Hern (TP077536)
1.0 Introduction	25%	25%	25%	25%
2.0 Data Preparation	25%	25%	25%	25%
3.0 Data Analysis	25%	25%	25%	25%
4.0 Conclusion	25%	25%	25%	25%

6.0 References

Bobbitt, Z. (2019, March 28). *How to Create a Heatmap in R Using ggplot2*. Statology. https://www.statology.org/heatmap-r-ggplot2/?form=MG0AV3

Bobbitt, Z. (2022, July 22). *How to Jitter Points in ggplot2 (With Examples).* Statology. https://www.statology.org/ggplot-jitter/

Kidman, A. (2004, February 19). *Linux hacked more often than Windows.* ZDNET. https://www.zdnet.com/article/linux-hacked-more-often-than-windows/

Microsoft. (2025). *Internet Information Services (IIS)—Microsoft Lifecycle*. https://learn.microsoft.com/en-us/lifecycle/products/internet-information-services-iis

Reese, W. (2008). *Nginx: The high-performance web server and reverse proxy*. Linux J., 2008(173), 2:2.