```
library(tidyverse)


df1209 <- read_csv("/kaggle/input/exploring-cybersecurity-risk-via-2022-cisa-vulne/2022-12-09-enriched.csv")
df0704 <- read_csv("/kaggle/input/exploring-cybersecurity-risk-via-2022-cisa-vulne/2022-07-04-enriched.csv")
df0627 <- read_csv("/kaggle/input/exploring-cybersecurity-risk-via-2022-cisa-vulne/2022-06-27-enriched.csv")
df0609 <- read_csv("/kaggle/input/exploring-cybersecurity-risk-via-2022-cisa-vulne/2022-06-09-enriched.csv")
df0608 <- read_csv("/kaggle/input/exploring-cybersecurity-risk-via-2022-cisa-vulne/2022-06-08-enriched.csv")

all_df <- rbind(df0608, df0609, df0627, df0704, df1209)
```

```
Rows: 860 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (11): cve_id, vendor_project, product, vulnerability_name, short_descri...
dbl   (2): grp, cvss
date  (3): date_added, due_date, pub_date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 787 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (11): cve_id, vendor_project, product, vulnerability_name, short_descri...
dbl   (2): grp, cvss
date  (3): date_added, due_date, pub_date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 786 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (10): cve_id, vendor_project, product, vulnerability_name, short_descri...
dbl   (2): grp, cvss
lgl   (1): notes
date  (3): date_added, due_date, pub_date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 777 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (10): cve_id, vendor_project, product, vulnerability_name, short_descri...
```

```
dbl   (2): grp, cvss
lgl   (1): notes
date  (3): date_added, due_date, pub_date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 774 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (10): cve_id, vendor_project, product, vulnerability_name, short_descri...
dbl   (2): grp, cvss
lgl   (1): notes
date  (3): date_added, due_date, pub_date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(all_df)
```

A tibble: 6 × 16

cve_id	vendor_project	product	vulnerability_name	date_added	short_description	required_action	due_date	notes	grp
<chr>	<chr>	<chr>	<chr>	<date>	<chr>	<chr>	<date>	<chr>	<dbl>
CVE-2021-27104	accellion	FTA	Accellion FTA OS Command Injection Vulnerability	2021-11-03	Accellion FTA 9_12_370 and earlier is affected by OS command execution via a crafted POST request to various admin endpoints.	Apply updates per vendor instructions.	2021-11-17	NA	1
CVE-2021-27102	accellion	FTA	Accellion FTA OS Command Injection Vulnerability	2021-11-03	Accellion FTA 9_12_411 and earlier is affected by OS command execution via a local web service call.	Apply updates per vendor instructions.	2021-11-17	NA	1
CVE-2021-27101	accellion	FTA	Accellion FTA SQL Injection Vulnerability	2021-11-03	Accellion FTA 9_12_370 and earlier is affected by SQL injection via a crafted Host header in a request to document_root.html.	Apply updates per vendor instructions.	2021-11-17	NA	1
CVE-2021-27103	accellion	FTA	Accellion FTA SSRF Vulnerability	2021-11-03	Accellion FTA 9_12_411 and earlier is affected by SSRF via a crafted POST request to wmProgressstat.html.	Apply updates per vendor instructions.	2021-11-17	NA	1
CVE-		Acrobat	Adobe Acrobat and Reader Heap-based		Acrobat Reader DC versions versions 2020.013.20074 (and earlier), 2020.001.30018 (and earlier) and 2017.011.30188 (and earlier) are affected by a heap-based buffer overflow vulnerability. An	Apply updates per	2021-11-		

2021-21017	adobe	and Reader	Reader Heap based Buffer Overflow Vulnerability	2021-11-03	unauthenticated attacker could leverage this vulnerability to achieve arbitrary code execution in the context of the current user. Exploitation of this issue requires user interaction in that a victim must open a malicious file.	vendor instructions.	2021-11-17	NA	1
CVE-2021-28550	adobe	Acrobat and Reader	Adobe Acrobat and Reader Use-After-Free Vulnerability	2021-11-03	Acrobat Reader DC versions versions 2021.001.20150 (and earlier), 2020.001.30020 (and earlier) and 2017.011.30194 (and earlier) are affected by a Use After Free vulnerability. An unauthenticated attacker could leverage this vulnerability to achieve arbitrary code execution in the context of the current user. Exploitation of this issue requires user interaction in that a victim must open a malicious file.	Apply updates per vendor instructions.	2021-11-17	NA	1

```
dim(all_df)
```

3984 · 16

```
summary(all_df)
```

```
   cve_id          vendor_project       product          vulnerability_name
Length:3984        Length:3984        Length:3984        Length:3984
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character



  date_added         short_description  required_action
Min.   :2021-11-03   Length:3984        Length:3984
1st Qu.:2021-11-03   Class :character   Class :character
Median :2022-03-03   Mode  :character   Mode  :character
Mean   :2022-02-05
3rd Qu.:2022-03-28
Max.   :2022-12-05

   due_date            notes               grp           pub_date
Min.   :2020-01-29   Length:3984        Min.   : 1.00   Min.   :2002-06-25
1st Qu.:2022-03-24   Class :character   1st Qu.: 1.00   1st Qu.:2016-11-10
Median :2022-05-03   Mode  :character   Median :16.00   Median :2019-06-27
Mean   :2022-04-09                      Mean   :14.42   Mean   :2018-06-15
3rd Qu.:2022-06-10                      3rd Qu.:20.00   3rd Qu.:2021-02-11
Max.   :2022-12-26                      Max.   :64.00   Max.   :2022-06-03
                                                        NA's   :878
     cvss            cwe              vector           complexity
Min.   : 3.100   Length:3984        Length:3984        Length:3984
1st Qu.: 7.800   Class :character   Class :character   Class :character
Median : 8.800   Mode  :character   Mode  :character   Mode  :character
Mean   : 8.412
3rd Qu.: 9.800
Max.   :10.000
NA's   :1500
  severity
Length:3984
Class :character
Mode  :character
```

```
# CHeck the number of NA values for each column
sapply(all_df, function(x) sum(is.na(x)))
```

```
cve_id: 0 vendor_project: 0 product: 5 vulnerability_name: 0 date_added: 0 short_description: 30 required_action:
0 due_date: 0 notes: 3907 grp: 0 pub_date: 878 cvss: 1500 cwe: 894 vector: 1500 complexity: 1500
severity: 1500
```

```
# Remove "notes" column as it contains unuseful info and a lot of NA values
all_df <- subset(all_df, select = -c(notes))

sapply(all_df, function(x) sum(is.na(x)))
```

```
cve_id: 0 vendor_project: 0 product: 5 vulnerability_name: 0 date_added: 0 short_description: 30 required_action:
0 due_date: 0 grp: 0 pub_date: 878 cvss: 1500 cwe: 894 vector: 1500 complexity: 1500 severity:
1500
```

```
str(all_df)
```

```
tibble [3,984 × 15] (S3: tbl_df/tbl/data.frame)
 $ cve_id            : chr [1:3984] "CVE-2021-27104" "CVE-2021-27102" "CVE-2021-27101" "CVE-2021-27103" ...
 $ vendor_project    : chr [1:3984] "accellion" "accellion" "accellion" "accellion" ...
 $ product           : chr [1:3984] "FTA" "FTA" "FTA" "FTA" ...
 $ vulnerability_name: chr [1:3984] "Accellion FTA OS Command Injection Vulnerability" "Accellion FTA OS Command Injection Vulnerability" "Acc
 $ date_added        : Date[1:3984], format: "2021-11-03" "2021-11-03" ...
 $ short_description : chr [1:3984] "Accellion FTA 9_12_370 and earlier is affected by OS command execution via a crafted POST request to vari
 $ required_action   : chr [1:3984] "Apply updates per vendor instructions." "Apply updates per vendor instructions." "Apply updates per vendo
 $ due_date          : Date[1:3984], format: "2021-11-17" "2021-11-17" ...
 $ grp               : num [1:3984] 1 1 1 1 1 1 1 1 1 1 ...
 $ pub_date          : Date[1:3984], format: "2021-02-16" "2021-02-16" ...
 $ cvss              : num [1:3984] 9.8 7.8 9.8 9.8 8.8 8.8 9.8 9.8 9.8 8.8 ...
 $ cwe               : chr [1:3984] "CWE-78" "CWE-78" "CWE-89" "CWE-918" ...
 $ vector            : chr [1:3984] "NETWORK" "LOCAL" "NETWORK" "NETWORK" ...
 $ complexity        : chr [1:3984] "LOW" "LOW" "LOW" "LOW" ...
 $ severity          : chr [1:3984] "CRITICAL" "HIGH" "CRITICAL" "CRITICAL" ...
```

Imputation

After looking at the data frame throughly, i suspect that columns who share the same cve_id have identical values.

```
# Group the data by cve_id and count the number of unique values in each column
grouped_df <- all_df %>%
  group_by(cve_id) %>%
  summarize_all(n_distinct)

# Check if all columns have the same number of unique values for each cve_id
identical(grouped_df[-1], colSums(!is.na(grouped_df[-1]))[1])
```

```
FALSE
```

```
# Check which columns have different numbers of unique values
which(!identical(grouped_df[-1], colSums(!is.na(grouped_df[-1]))[1]))
```

```
1
```

Only first column has differetn values, which has the column names.

```
sapply(all_df, function(x) sum(is.na(x)))
```

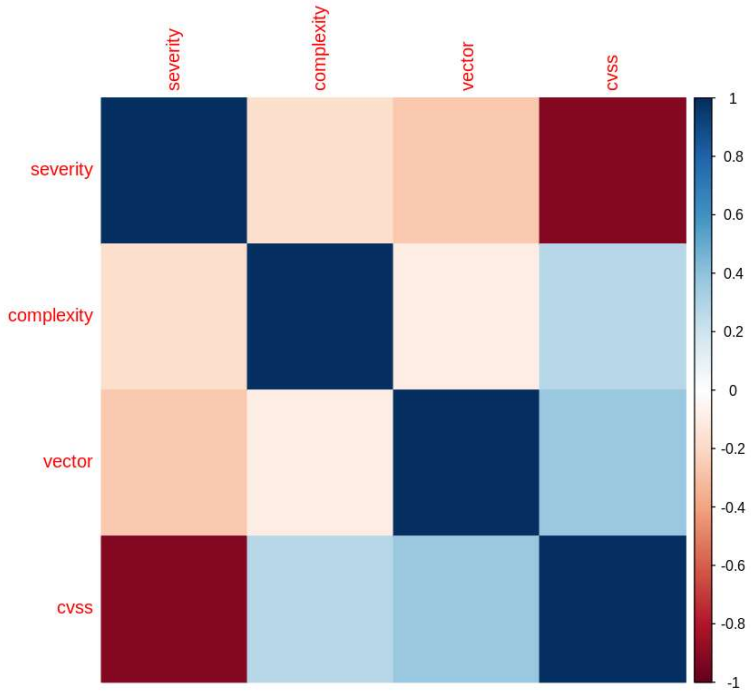
```
cve_id: 0 vendor_project: 0 product: 5 vulnerability_name: 0 date_added: 0 short_description: 30 required_action: 0 due_date: 0 grp: 0 pub_date: 878 cvss: 1500 cwe: 894 vector: 1500 complexity: 1500 severity: 1500
```

```
# Impute missing values in all columns
all_imp <- all_df %>%
  group_by(cve_id) %>%
  mutate(across(everything(), ~if_else(is.na(.), first(.[!is.na(.)]), .)))
```

```
sapply(all_imp, function(x) sum(is.na(x)))
```

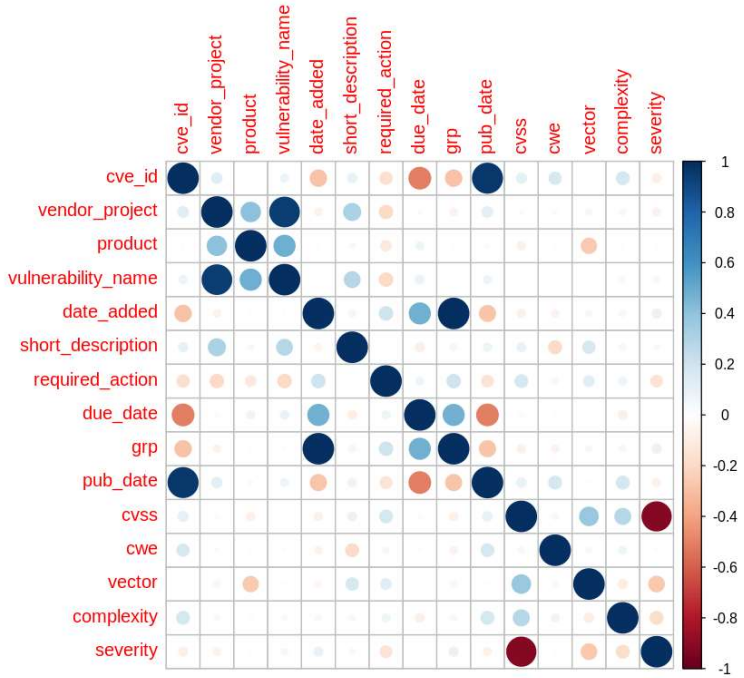
```
cve_id: 0 vendor_project: 0 product: 5 vulnerability_name: 0 date_added: 0 short_description: 30 required_action: 0 due_date: 0 grp: 0 pub_date: 98 cvss: 863 cwe: 103 vector: 863 complexity: 863 severity: 863
```

```
# create a copy of the original data frame for the new numeric one
all_num_omitted <- all_imp

# encode categorical variables using label encoding
all_num_omitted$product <- as.numeric(as.factor(all_df$product))
all_num_omitted$vulnerability_name <- as.numeric(as.factor(all_df$vulnerability_name))
all_num_omitted$short_description <- as.numeric(as.factor(all_df$short_description))
all_num_omitted$required_action <- as.numeric(as.factor(all_df$required_action))
all_num_omitted$cwe <- as.numeric(as.factor(all_df$cwe))
all_num_omitted$vector <- as.numeric(as.factor(all_df$vector))
all_num_omitted$complexity <- as.numeric(as.factor(all_df$complexity))
all_num_omitted$severity <- as.numeric(as.factor(all_df$severity))
all_num_omitted$vendor_project <- as.numeric(as.factor(all_df$vendor_project))
all_num_omitted$cve_id <- as.numeric(as.factor(all_df$cve_id))
all_num_omitted$date_added <- as.numeric(as.factor(all_df$date_added))
all_num_omitted$due_date <- as.numeric(as.factor(all_df$due_date))
all_num_omitted$pub_date <- as.numeric(as.factor(all_df$pub_date))

all_num_omitted <- na.omit(all_num_omitted)


install.packages("corrplot")


     Installing package into ‘/usr/local/lib/R/site-library’
     (as ‘lib’ is unspecified)



library(corrplot)

columns_of_interest <- c("severity", "complexity", "vector", "cvss")

subset_df <- all_num_omitted[, columns_of_interest]

correlation_matrix <- cor(subset_df)

corrplot(correlation_matrix, method = "color")
```

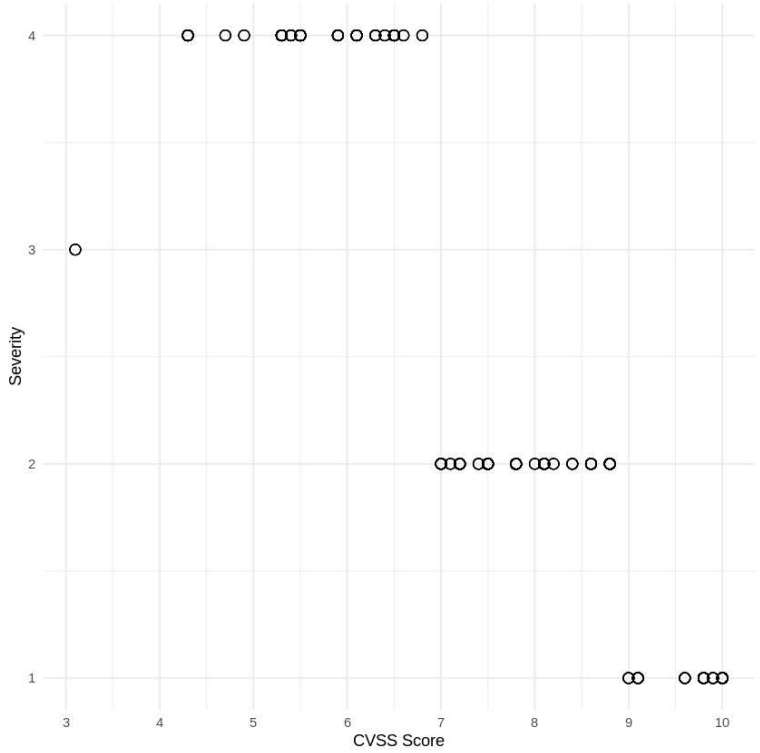


cvss and severity has a strong negative correlation.

```
correlation_matrix_all <- cor(all_num_omitted)

corrplot(correlation_matrix_all)
```




```
ggplot(all_num_omitted, aes(x = cvss, y = severity)) +
  geom_point(shape = 1, size = 3, color = "black") +
  labs(x = "CVSS Score", y = "Severity") +
  scale_x_continuous(breaks = seq(0, 10, 1)) +
  theme_minimal()
```



Identifying new imputation opportunities

Numerical notations of severity levels: CRITICAL = 1, HIGH = 2, LOW = 3, MEDIUM 4

- Severity: x < 4 = LOW, 4 <= x < 7 = MEDIUM, 7 <= x < 9 = HIGH, x > 9 = CRITICAL
- pub_date = cve_id (pub_date is the same for cells who share the same cve_id)
- fuel cms = fuel cms (5 columns have vendor_project named "fuel cms" doesn't have a product name, so i decided to name them fuel cms as i don't want to omit the variables because of the missing product name)

```
# Impute product and short description columns
all_imp$product[is.na(all_imp$product)] <- "fuel cms"
all_imp$short_description[is.na(all_imp$short_description)] <- "na"

# Impute pub_date for NA values based on cve_id
all_imp <- all_imp %>%
  group_by(cve_id) %>%
  mutate(pub_date = ifelse(is.na(pub_date), max(pub_date, na.rm = TRUE), pub_date))


# Check if NA values in cvss and severity are in the same rows
same_na_rows <- all(is.na(all_imp$cvss) == is.na(all_imp$severity))
same_na_rows # :(

colSums(is.na(all_imp))
```

```
Warning message:
“There were 78 warnings in `mutate()`.
The first warning was:
i In argument: `pub_date = ifelse(is.na(pub_date), max(pub_date, na.rm = TRUE),
  pub_date)`.
i In group 29: `cve_id = "CVE-2010-2568"`.
Caused by warning in `max.default()`:
! no non-missing arguments to max; returning -Inf
i Run `dplyr::last_dplyr_warnings()` to see the 77 remaining warnings.”
TRUE
cve_id: 0 vendor_project: 0 product: 0 vulnerability_name: 0 date_added: 0 short_description: 0 required_action:
0 due_date: 0 grp: 0 pub_date: 0 cvss: 863 cwe: 103 vector: 863 complexity: 863 severity: 863
```

```
all_imp <- na.omit(all_imp)

colSums(is.na(all_imp))
```

```
cve_id: 0 vendor_project: 0 product: 0 vulnerability_name: 0 date_added: 0 short_description: 0 required_action:
0 due_date: 0 grp: 0 pub_date: 0 cvss: 0 cwe: 0 vector: 0 complexity: 0 severity: 0
```

```
all_num_clean <- all_imp

# encode categorical variables using label encoding
all_num_clean$product <- as.numeric(as.factor(all_imp$product))
all_num_clean$vulnerability_name <- as.numeric(as.factor(all_imp$vulnerability_name))
all_num_clean$short_description <- as.numeric(as.factor(all_imp$short_description))
all_num_clean$required_action <- as.numeric(as.factor(all_imp$required_action))
all_num_clean$cwe <- as.numeric(as.factor(all_imp$cwe))
all_num_clean$vector <- as.numeric(as.factor(all_imp$vector))
all_num_clean$complexity <- as.numeric(as.factor(all_imp$complexity))
all_num_clean$severity <- as.numeric(as.factor(all_imp$severity))
all_num_clean$vendor_project <- as.numeric(as.factor(all_imp$vendor_project))
all_num_clean$cve_id <- as.numeric(as.factor(all_imp$cve_id))
all_num_clean$date_added <- as.numeric(as.factor(all_imp$date_added))
all_num_clean$due_date <- as.numeric(as.factor(all_imp$due_date))
all_num_clean$pub_date <- as.numeric(as.factor(all_imp$pub_date))

# Verify the structure of the new data frame
str(all_num_clean)
```

```
          .. ..$ : int [1:5] 519 1135 1754 2382 3011
          .. ..$ : int [1:5] 581 1197 1816 2444 3073
          .. ..$ : int [1:5] 177 793 1412 2040 2669
          .. ..$ : int [1:5] 582 1198 1817 2445 3074
          .. ..$ : int [1:5] 498 1114 1733 2361 2990
          .. ..$ : int [1:5] 365 981 1600 2228 2857
          .. ..$ : int [1:5] 414 1030 1649 2277 2906
          .. ..$ : int [1:5] 349 965 1584 2212 2841
          .. ..$ : int [1:5] 348 964 1583 2211 2840
          .. ..$ : int [1:5] 317 933 1552 2180 2809
          .. ..$ : int [1:5] 347 963 1582 2210 2839
          .. ..$ : int [1:5] 413 1029 1648 2276 2905
          .. ..$ : int [1:5] 527 1143 1762 2390 3019
          .. ..$ : int [1:5] 180 796 1415 2043 2672
          .. ..$ : int [1:5] 412 1028 1647 2275 2904
          .. ..$ : int [1:5] 173 789 1408 2036 2665
          .. ..$ : int [1:5] 299 915 1534 2162 2791
          .. ..$ : int [1:5] 411 1027 1646 2274 2903
          .. ..$ : int [1:5] 410 1026 1645 2273 2902
          .. ..$ : int [1:5] 409 1025 1644 2272 2901
          .. ..$ : int [1:5] 408 1024 1643 2271 2900
          .. ..$ : int [1:5] 407 1023 1642 2270 2899
          .. ..$ : int [1:5] 406 1022 1641 2269 2898
          .. ..$ : int [1:5] 405 1021 1640 2268 2897
          .. ..$ : int [1:5] 404 1020 1639 2267 2896
          .. ..$ : int [1:5] 403 1019 1638 2266 2895
          .. ..$ : int [1:5] 479 1095 1714 2342 2971
          .. ..$ : int [1:5] 478 1094 1713 2341 2970
          .. ..$ : int [1:5] 225 841 1460 2088 2717
          .. ..$ : int [1:5] 298 914 1533 2161 2790
          .. .. [list output truncated]
          .. ..@ ptype: int(0)
          ..- attr(*, ".drop")= logi TRUE
        - attr(*, "na.action")= 'omit' Named int [1:863] 124 172 173 210 212 305 316 340 341 349 ...
          - attr(*, "names")= chr [1:863] "124" "172" "173" "210"
```

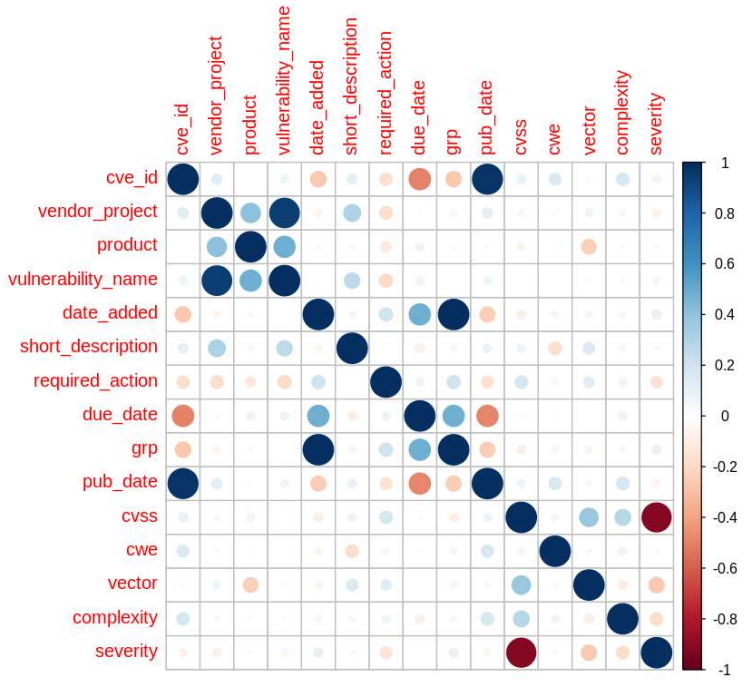
```
colSums(is.na(all_num_clean))
```

```
      cve_id:          0 vendor_project:          0 product:          0 vulnerability_name:          0 date_added:          0 short_description:          0 required_action:
           0 due_date:          0 grp:          0 pub_date:          0 cvss:          0 cwe:          0 vector:          0 complexity:          0 severity:          0
```

After doing all imputations we have two data frames at hand; all_num_clean and all_imp. all_num_clean only has numerical values while all_imp has original data types. We are going to use both data frames in the further investigations.

Correlation

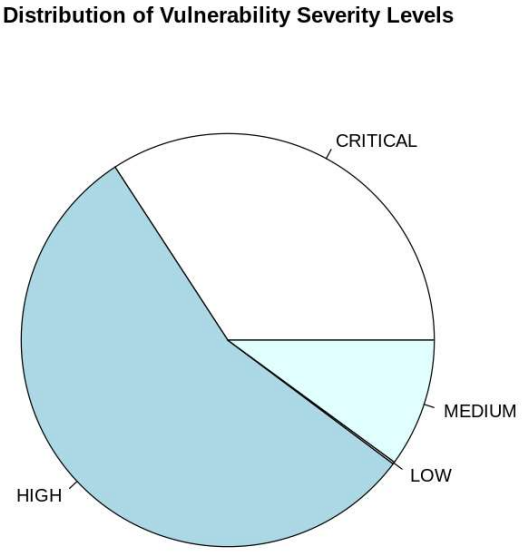
```
correlation_matrix2 <-  cor(all_num_clean)

corrplot(correlation_matrix2)
```



EDA

- ### Analysis of Vulnerability Severity Levels

```
severity_counts <- table(all_imp$severity)

pie(severity_counts, labels = names(severity_counts),
    main = "Distribution of Vulnerability Severity Levels")
```



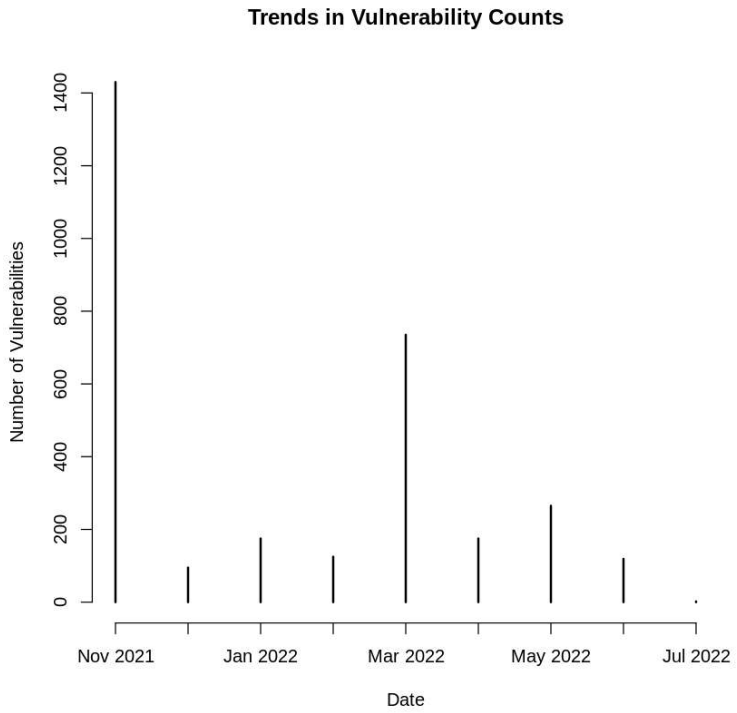
- ## Vulnerability Trends over Time

```
install.packages("zoo")
```

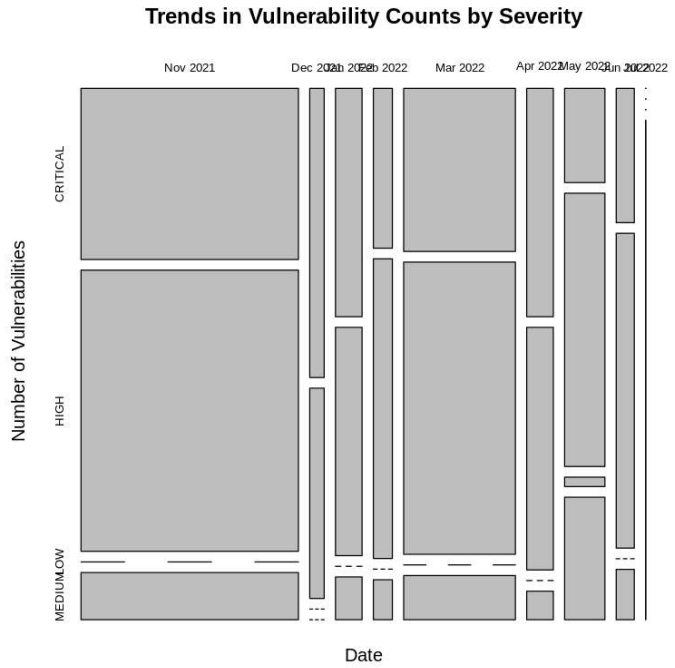
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(zoo)

# Convert date_azdded column to year-month format
all_imp$date_added <- as.yearmon(all_imp$date_added)

vuln_counts <- table(all_imp$date_added)

plot(vuln_counts, xlab = "Date", ylab = "Number of Vulnerabilities",
     main = "Trends in Vulnerability Counts")
```

```
Attaching package: ‘zoo’


The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric
```



```
severity_counts <- table(all_imp$date_added, all_imp$severity)

plot(severity_counts, xlab = "Date", ylab = "Number of Vulnerabilities",
     main = "Trends in Vulnerability Counts by Severity")
```



- ## Proportion of Vulnerabilities with Due Dates Met

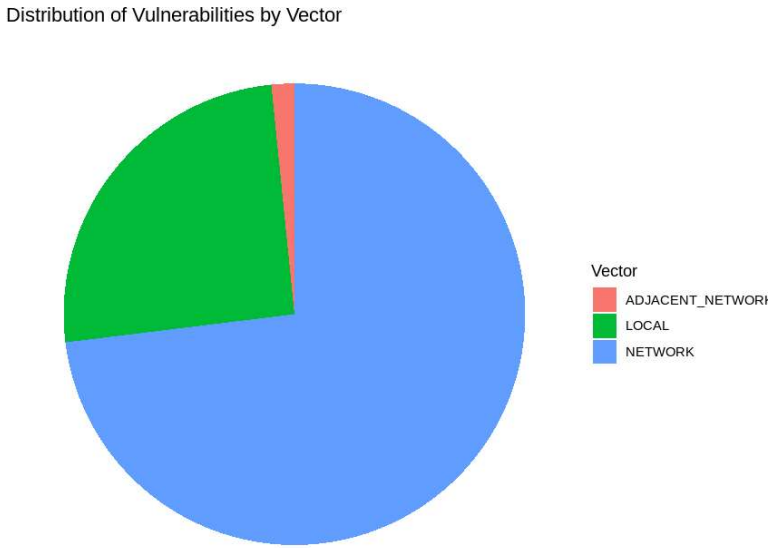
```
# Calculate the proportion of vulnerabilities with due dates met
proportion_due_dates_met <- sum(!is.na(all_imp$due_date)) / nrow(all_imp) * 100

print(paste("Proportion of vulnerabilities with due dates met:", proportion_due_dates_met, "%"))
```

```
[1] "Proportion of vulnerabilities with due dates met: 100 %"
```

```
vulnerabilities_by_vector <- all_imp %>%
  group_by(vector) %>%
  summarise(count = n()) %>%
  drop_na()

vulnerabilities_by_vector

chart_vector <- ggplot(vulnerabilities_by_vector, aes(x = "", y = count, fill = vector)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  labels(fill = "Vector", title = "Distribution of Vulnerabilities by Vector") +
  theme_void()

chart_vector
```

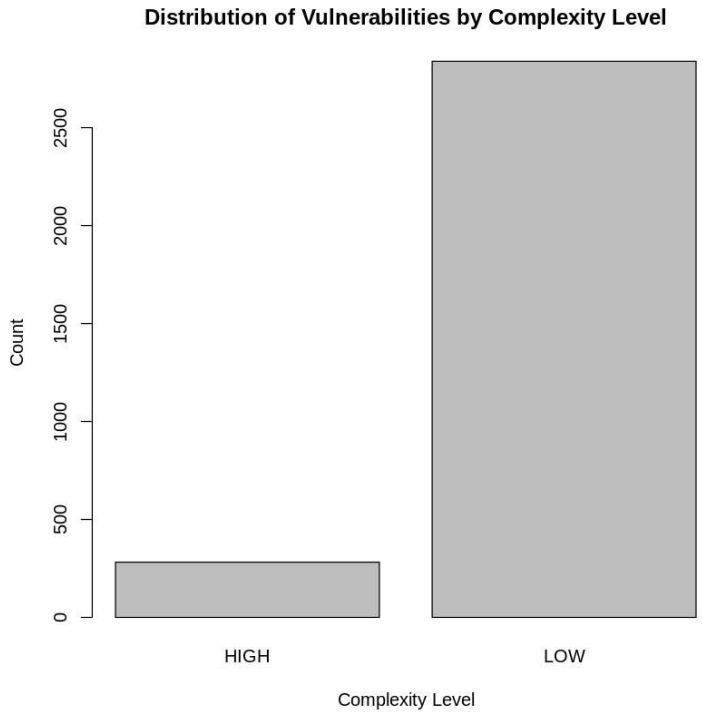
A tibble: 3 × 2

vector	count
<chr>	<int>
ADJACENT_NETWORK	50
LOCAL	791
NETWORK	2280



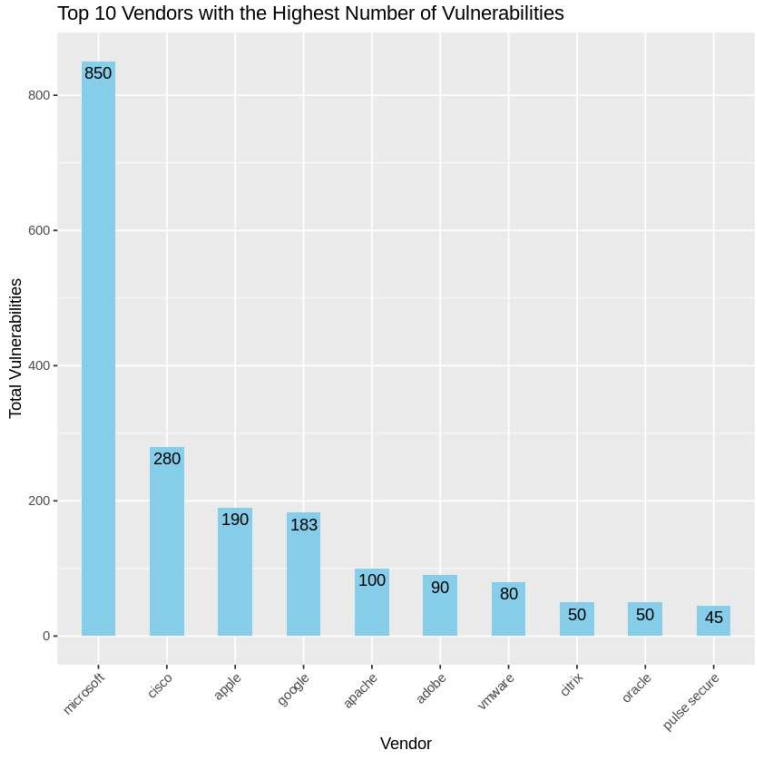

```
complexity_counts <- table(all_imp$complexity)
complexity_counts

barplot(complexity_counts, main = "Distribution of Vulnerabilities by Complexity Level", xlab = "Complexity Level", ylab = "Count")
```

```
HIGH  LOW
 282 2839
```

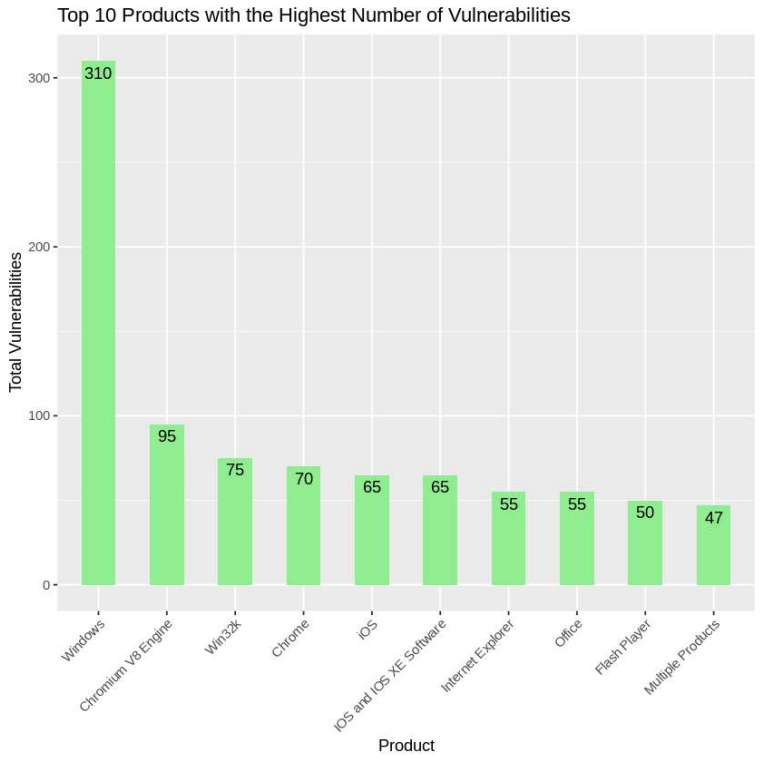


- Top Vendors with the Highest Number of Vulnerabilities

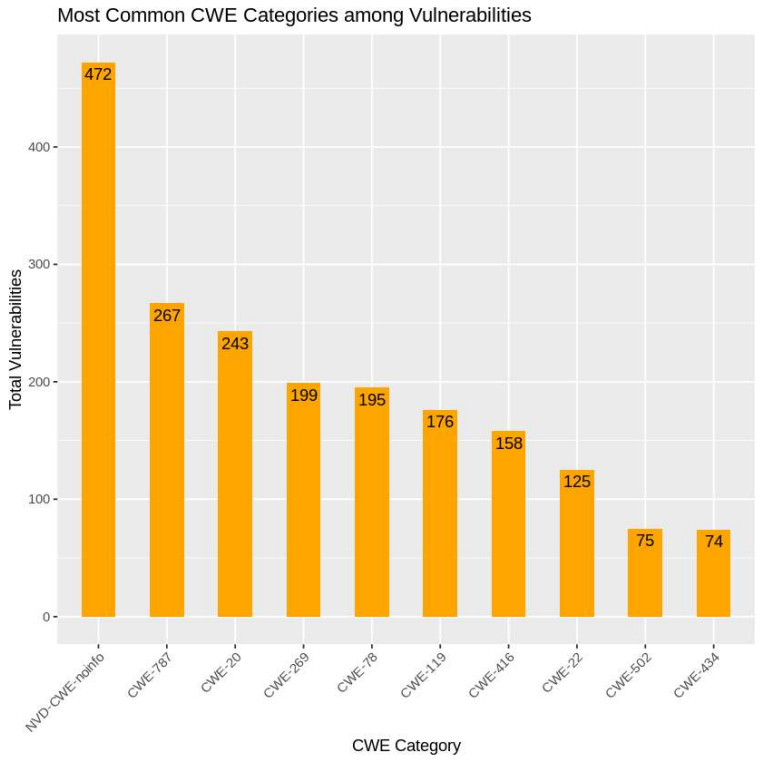
```
top_vendors <- all_imp %>%
  group_by(vendor_project) %>%
  summarize(total_vulnerabilities = n()) %>%
  arrange(desc(total_vulnerabilities)) %>%
  head(10)

top_vendors_chart <- ggplot(top_vendors, aes(x = reorder(vendor_project, -total_vulnerabilities), y = total_vulnerabilities)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.5) +
  geom_text(aes(label = total_vulnerabilities), hjust = 0.5, vjust = 1.5) +
  labs(title = "Top 10 Vendors with the Highest Number of Vulnerabilities",
       x = "Vendor",
       y = "Total Vulnerabilities") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

top_vendors_chart
```



```
top_products <- all_imp %>%
  group_by(product) %>%
  summarize(total_vulnerabilities = n()) %>%
  arrange(desc(total_vulnerabilities)) %>%
  head(10)


# Bar chart for top products
top_products_chart <- ggplot(top_products, aes(x = reorder(product, -total_vulnerabilities), y = total_vulnerabilities)) +
  geom_bar(stat = "identity", fill = "lightgreen", width = 0.5) +
  geom_text(aes(label = total_vulnerabilities), hjust = 0.5, vjust = 1.5) +
  labs(title = "Top 10 Products with the Highest Number of Vulnerabilities",
       x = "Product",
       y = "Total Vulnerabilities") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

top_products_chart
```



```
common_cwe <- all_imp %>%
  group_by(cwe) %>%
  summarize(total_vulnerabilities = n()) %>%
  arrange(desc(total_vulnerabilities)) %>%
  head(10)

# Bar chart for common CWE categories
common_cwe_chart <- ggplot(common_cwe, aes(x = reorder(cwe, -total_vulnerabilities), y = total_vulnerabilities)) +
  geom_bar(stat = "identity", fill = "orange", width = 0.5) +
  geom_text(aes(label = total_vulnerabilities), hjust = 0.5, vjust = 1.5) +
  labs(title = "Most Common CWE Categories among Vulnerabilities",
       x = "CWE Category",
       y = "Total Vulnerabilities") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

common_cwe_chart
```

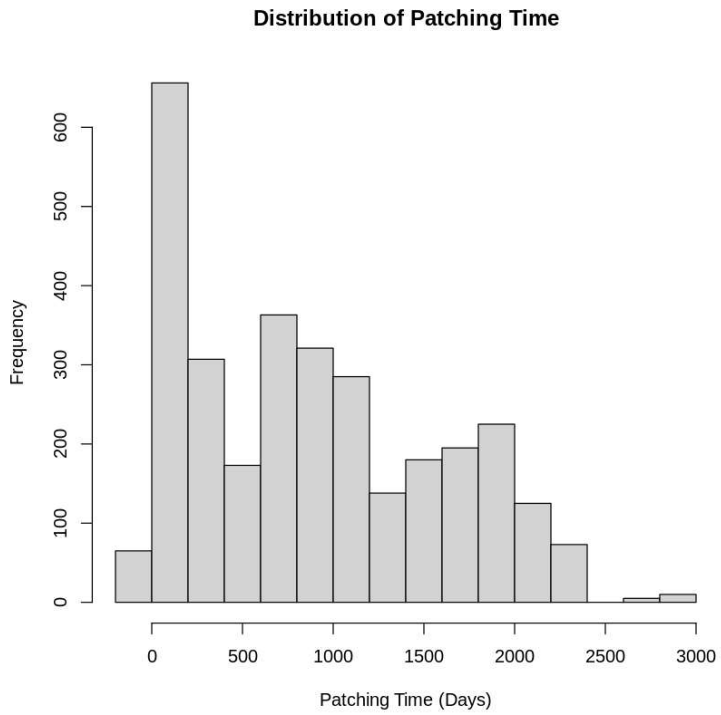


Analysis on Patching Speed

```
all_imp$patching_time <- all_imp$due_date - all_imp$pub_date

all_imp$patching_time <- as.numeric(all_imp$patching_time)

summary(all_imp$patching_time)
```

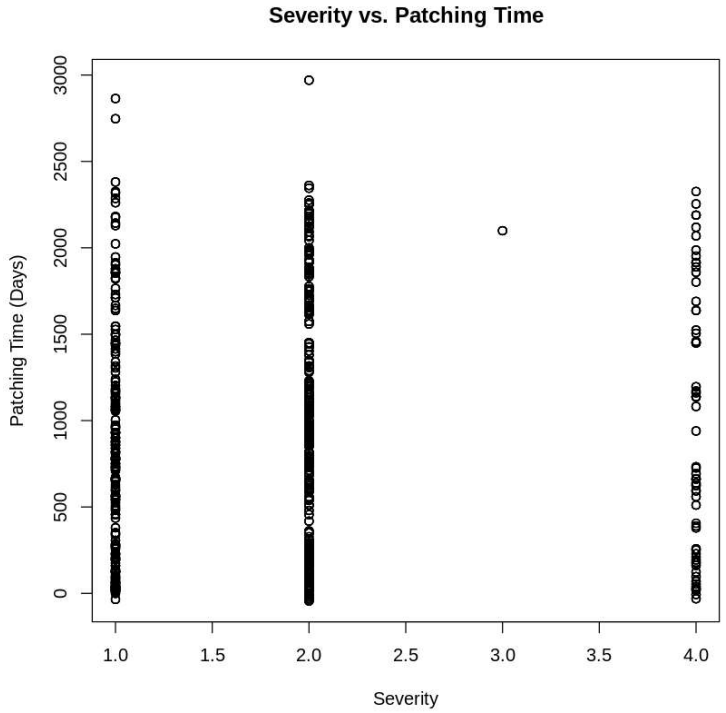
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -44.0   215.0   796.0   885.8  1445.0  2970.0
```

```
hist(all_imp$patching_time, breaks = 20, xlab = "Patching Time (Days)", main = "Distribution of Patching Time")
```



Severity vs Patching Time

```
plot(all_num_clean$severity, all_imp$patching_time, xlab = "Severity", ylab = "Patching Time (Days)", main = "Severity vs. Patching Time")
```




```
cor(all_num_clean$severity, all_imp$patching_time)
```

```
0.0644033619516698
```

```
install.packages("igraph")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(igraph)

# Create an edge list
edges <- data.frame(source = all_imp$vendor_project, target = all_imp$product)

# Remove missing values or empty strings
edges <- na.omit(edges)
edges <- edges[edges$source != "", ]
edges <- edges[edges$target != "", ]

# Create a graph object
g <- graph_from_data_frame(edges, directed = FALSE)

# Calculate degree centrality
centrality <- degree(g)

# Get the nodes with the highest degree centrality
most_central <- names(centrality)[centrality == max(centrality)]

plot(g, vertex.size = 10, vertex.label.cex = 0.6, edge.arrow.size = 0.5, edge.width = 0.5)
```

```
Attaching package: ‘igraph’

The following objects are masked from ‘package:lubridate’:

    %--%, union

The following objects are masked from ‘package:dplyr’:

    as_data_frame, groups, union

The following objects are masked from ‘package:purrr’:

    compose, simplify

The following object is masked from ‘package:tidyr’:

    crossing

The following object is masked from ‘package:tibble’:

    as_data_frame

The following objects are masked from ‘package:stats’:

    decompose, spectrum

The following object is masked from ‘package:base’:

    union
```

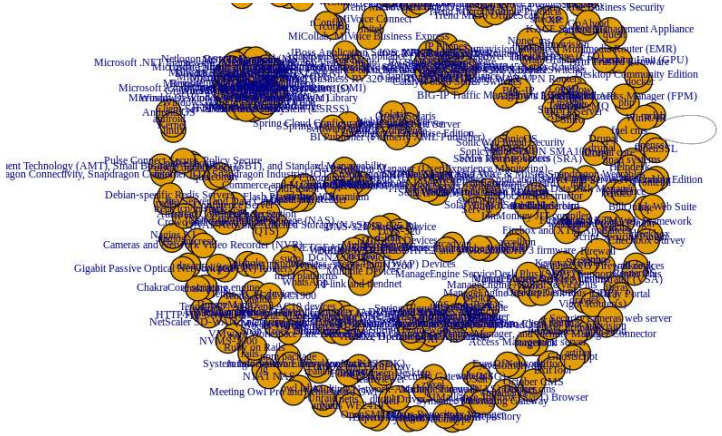


Chart looks very crowded because of the names but bear with me.

There is a clustering of nodes at the lower part inside the circle, as well as at the top edge, indicating product - vendor relationships.

```
install.packages("networkD3")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘htmlwidgets’
```

```
library(networkD3)

nd <- igraph_to_networkD3(g)

# Create a Group column and assign a default value
nd$nodes$Group <- 1

# Create an interactive plot
forceNetwork(Links = nd$links, Nodes = nd$nodes, Source = "source", Target = "target",
             NodeID = "name", Group = "Group", width = 800, height = 600,
             opacity = 0.8, linkDistance = 100,
             bounded = TRUE, linkColour = "#999999",
             colourScale = JS("d3.scaleOrdinal().range(['#66CCFF'])"),
             legend = TRUE)
```

Here is the animated and interactive versison of the same chart using networkD3 library. This plot requires some GPU power but it shows a better and more understandable relationship between vendors and products

- Most Central Vendors in the Vulnerability Network

```
graph <- graph.data.frame(all_imp, directed = FALSE)

degree_centrality <- degree(graph, mode = "all")

sorted_degree_centrality <- sort(degree_centrality, decreasing = TRUE)

top_vendors <- names(sorted_degree_centrality)[1:10]
top_vendors
```

```
Error in graph.data.frame(all_imp, directed = FALSE): could not find function "graph.data.frame"
Traceback:
```

EXPLAIN ERROR

The most central entities:

- Microsoft - Cisco - Apple - Google - Apache - Adobe - VMware - Citrix - Oracle - Pulse Secure

These entities exhibited a high degree of centrality, indicating that they had a significant number of vulnerabilities associated with them. This suggests that these vendors and products are potentially more prone to cybersecurity risks and require careful attention and proactive security measures.

It is interesting to note that these central entities encompass a wide range of industries, including software, technology, and networking. This highlights the pervasive nature of cybersecurity vulnerabilities across different sectors and emphasizes the need for comprehensive security measures across the board.

- Most Central Products in the Vulnerability Network

```
install.packages("ggrepel")
```

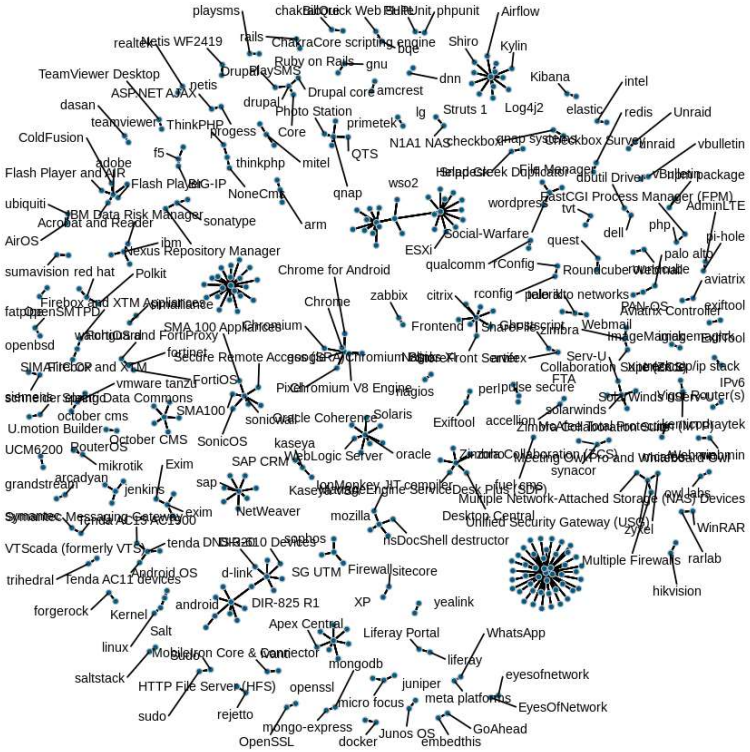
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘Rcpp’
```

```
install.packages("ggraph")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘tweenr’, ‘polyclip’, ‘RcppEigen’, ‘gridExtra’, ‘RcppArmadillo’, ‘ggforce’, ‘igraph’, ‘viridis’, ‘tidygraph’,
```

```
library(ggrepel)
library(ggraph)

network_data <- all_imp[, c("vendor_project", "product")]

# Create edge list
edge_list <- as.matrix(network_data)

graph <- graph_from_edgelist(edge_list, directed = FALSE)

# Create layout using Fruchterman-Reingold algorithm
layout <- layout_with_fr(graph)

# Convert graph layout to data frame
layout_df <- data.frame(layout)

# Add vendor/product names to layout data frame
layout_df$names <- V(graph)$name

graph_plot <- ggraph(graph, layout = "fr") +
geom_edge_link() +
geom_node_point(shape = 21, fill = "#005c87", color = "gray") +
geom_node_text(aes(label = name), repel = TRUE, box.padding = 0.5, size = 3, max.overlaps = 20) +
theme_void()


# Alternative with all texts displayed but there are a lot of overlaps
# graph_plot <- ggraph(graph, layout = "fr") +
#    geom_edge_link() +
#    geom_node_point(shape = 21, fill = "skyblue", color = "black") +
#    geom_node_text(aes(label = name), repel = TRUE, box.padding = 0.5, size = 3, max.overlaps = 1000) +
#    theme_void()

graph_plot
```

```
Warning message:
“ggrepel: 219 unlabeled data points (too many overlaps). Consider increasing max.overlaps”
```

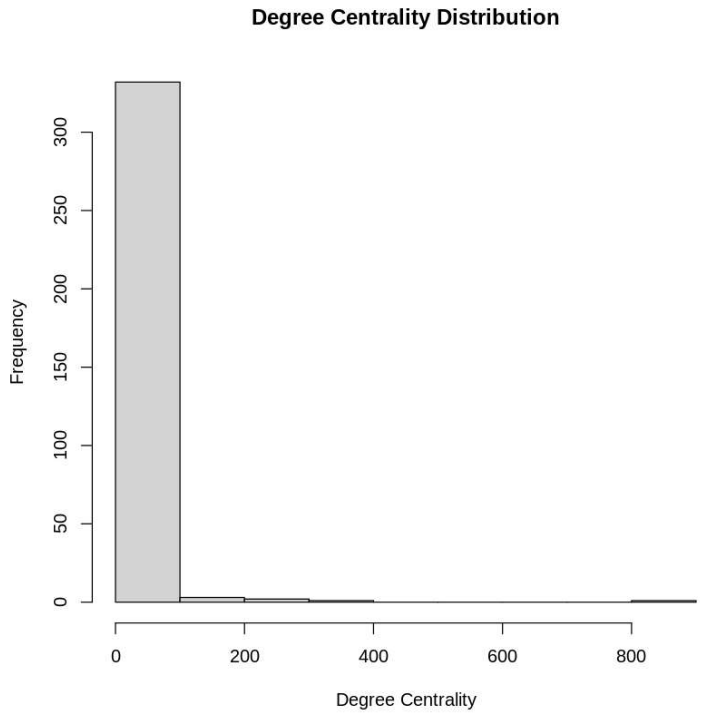


Looking at the resulting chart, we observe nodes with branches connecting them. Some nodes have a few branches, while others form clusters branching out to another cluster. There is a significant cluster of nodes, indicating a group of products that are closely interconnected.

- Are there Vendors/Products that are Highly Interconnected with other Vendors/Products, Indicating a Higher Level of Vulnerability?

In this section, we aim to explore the interconnectedness of vendors and products within the dataset. This analysis can provide insights into the level of vulnerability associated with certain vendors or products, as highly interconnected nodes may indicate a higher risk of cybersecurity issues.

```
g2 <- graph(edges = numeric(0))

for (i in 1:nrow(all_num_clean)) {
  vendor <- all_num_clean$vendor_project[i]
  product <- all_num_clean$product[i]

  # Add the vendor and product as vertices if they don't already exist
  if (!(vendor %in% V(g2))) {
    g2 <- add_vertices(g2, vendor)
  }
  if (!(product %in% V(g2))) {
    g2 <- add_vertices(g2, product)
  }

  g2 <- add_edges(g2, c(vendor, product))
}


degree_centrality2 <- degree(g2)

hist(degree_centrality2, main = "Degree Centrality Distribution", xlab = "Degree Centrality")
```




```
graph <- graph.empty(directed = FALSE)

nodes <- unique(c(all_imp$vendor_project, all_imp$product))
graph <- add.vertices(graph, length(nodes), name = nodes)

edges <- cbind(all_imp$vendor_project, all_imp$product)
graph <- add.edges(graph, edges)

degree_centrality <- degree(graph)

top_interconnected <- names(degree_centrality)[degree_centrality == max(degree_centrality)]

top_interconnected
```

Microsoft has the most interconnectedness with other products and vendors.

```
install.packages("visNetwork")
library(visNetwork)

# Create a data frame of nodes
nodes <- data.frame(id = unique(c(all_imp$vendor_project, all_imp$product)),
                    label = unique(c(all_imp$vendor_project, all_imp$product)))

# Create a data frame of edges
edges <- data.frame(from = all_imp$vendor_project, to = all_imp$product)

# Create the visNetwork object with nodes and edges data
network <- visNetwork(nodes, edges)

# Set the network options
network <- visOptions(network, width = "100%", height = "500px")
network <- visInteraction(network, hover = TRUE, hoverConnectedEdges = TRUE)
network <- visPhysics(network, enabled = TRUE)

network
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

Using the visNetwork() function, we create a visNetwork object, passing in the nodes and edges data frames. We can then customize the appearance and behavior of the network using various visOptions such as width, height, hover, and physics.

Finally, we display the interactive chart using the visIgraph() function, which renders the network visualization in the notebook.

You can zoom in to see which node represents which object.

Outcome

The network analysis reveals the interconnectedness of vendors and products within the dataset. By examining the nodes with the highest degree centrality, we can identify vendors or products that have a higher level of vulnerability due to their extensive connections with other nodes. The interactive chart allows for further exploration of the network structure and relationships between vendors and products.

When we visualize the network using the interactive chart, we can observe clusters and cliques of nodes connecting with each other. These clusters represent groups of vendors and products that have a higher level of interconnectedness among themselves. Nodes within the same cluster are more likely to have direct connections or dependencies on each other.

Now, regarding the observation of some clusters being in motion while others remain static, it is important to note that the visualization captures the dynamic nature of the network. The movements of the nodes in the chart are visual representations of the relationships between the nodes.

The motion of clusters in the chart indicates that there are dynamic relationships or interactions among the nodes within those clusters. It suggests that the vendors and products within those clusters are actively connected, collaborating, or dependent on each other. This dynamic nature can be due to various factors such as shared vulnerabilities, common dependencies, or ongoing collaborations.

On the other hand, the static clusters indicate a relatively stable or less dynamic relationship among the nodes within those clusters. It means that the vendors and products in those clusters may have fewer connections or dependencies with each other. This does not necessarily imply that they are less important or less vulnerable, but rather that their connections within the network are relatively stable or less active.

By observing the motion and static nature of the clusters, we gain insights into the varying levels of interconnectedness and dynamics present in the network. This information can help in understanding the overall structure and relationships within the network of cybersecurity risks, and potentially identify areas that require closer attention or further analysis.

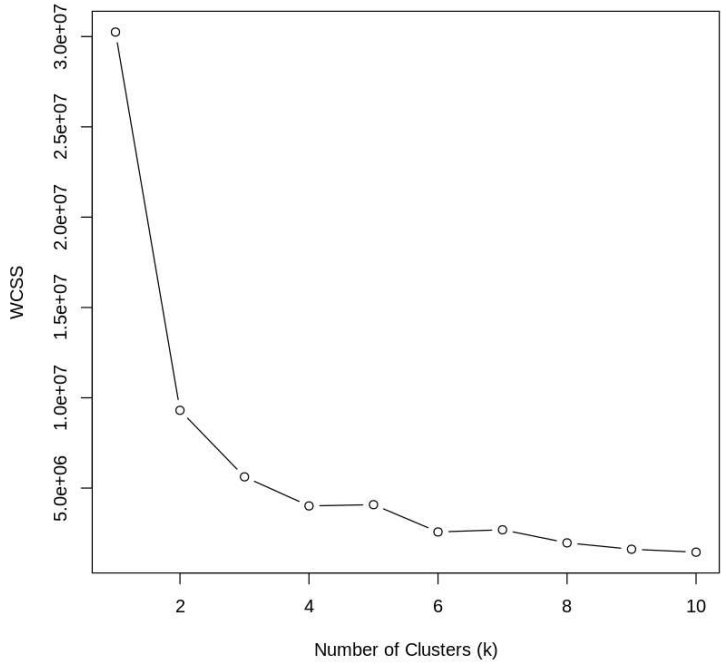
This network analysis provides valuable insights into the interconnected nature of vendors and products, highlighting potential areas of vulnerability and indicating which entities may have a greater impact on the overall network.

Clustering Analysis

Choosing the number of Clusters

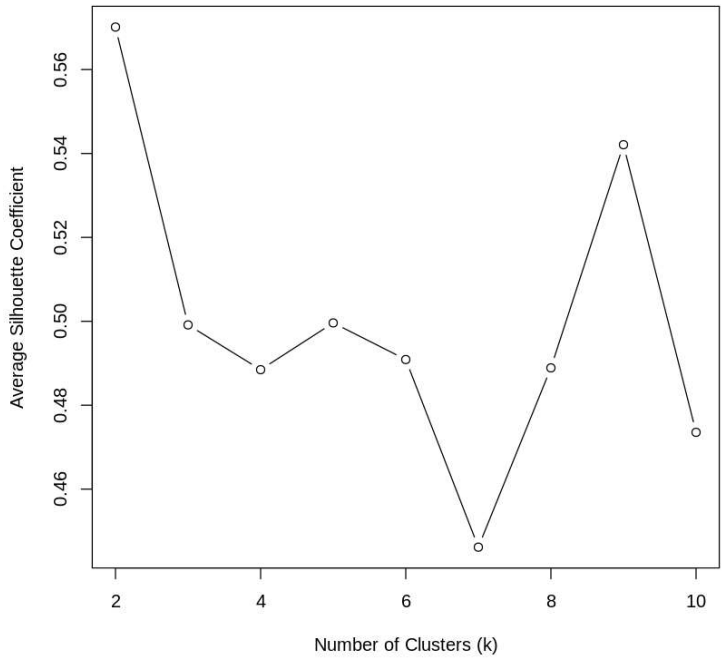
- #### Elbow Method

```
# Perform k-means clustering
library(cluster)

cluster_data <- all_num_clean[, c("vendor_project", "product")]

# Initialize empty vectors to store the WCSS values
wcss_values <- vector()

# Calculate WCSS for different values of k
for (k in 1:10) {
  kmeans_model <- kmeans(cluster_data, centers = k)
  wcss_values[k] <- kmeans_model$tot.withinss
}

# Plot the WCSS values against the number of clusters
plot(1:10, wcss_values, type = "b", xlab = "Number of Clusters (k)", ylab = "WCSS")
```



Silhoulette Method

```
# Calculate average silhouette coefficient for different values of k
silhouette_values <- list()

for (k in 2:10) {
  kmeans_model <- kmeans(cluster_data, centers = k)
  silhouette_obj <- silhouette(kmeans_model$cluster, dist(cluster_data))
  silhouette_values[[k]] <- mean(silhouette_obj[, 3])
}

# Plot the average silhouette coefficients against the number of clusters
plot(2:10, unlist(silhouette_values), type = "b", xlab = "Number of Clusters (k)", ylab = "Average Silhouette Coefficient")
```



Output

Elbow Method provides a clearer indication for choosing the number of clusters compared to the Silhouette Method. The Elbow Method shows a significant decline in the WCSS from k=1 to k=2, and the decrease continues as the number of clusters increases. This pattern suggests that adding more clusters beyond k=2 does not result in a substantial improvement in clustering quality.

On the other hand, the Silhouette Method shows relatively high silhouette coefficients across different values of k, including k=2, k=3, and k=4. This indicates that there is reasonable clustering structure present in the data for these values.

So i choose to go with 3.

Clusters/Communities of Vendors/Products that Share a High Number of Vulnerabilities

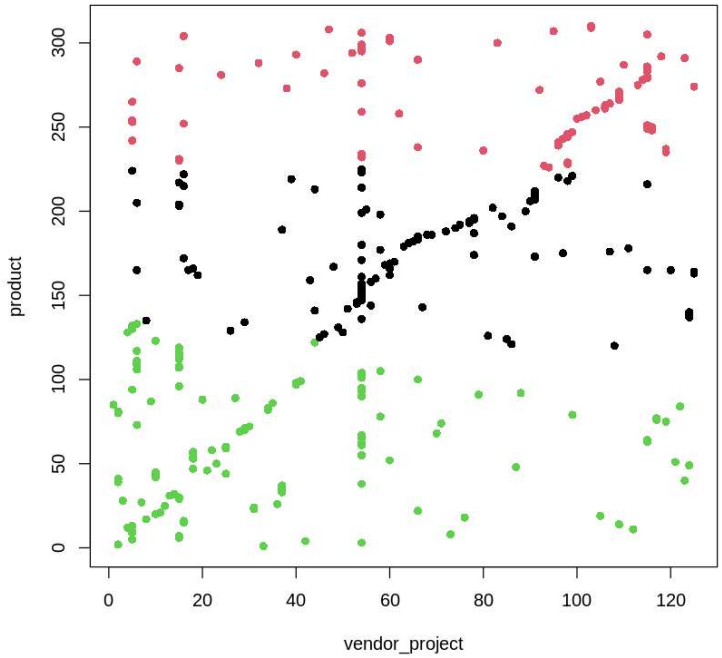
In this section, we performed a clustering analysis to identify clusters or communities of vendors and products that share a high number of vulnerabilities.

To begin the analysis, we selected the relevant columns from our dataset, namely "vendor_project" and "product," which represent the vendors and products associated with each vulnerability. Then applied the k-means clustering algorithm to these data points.

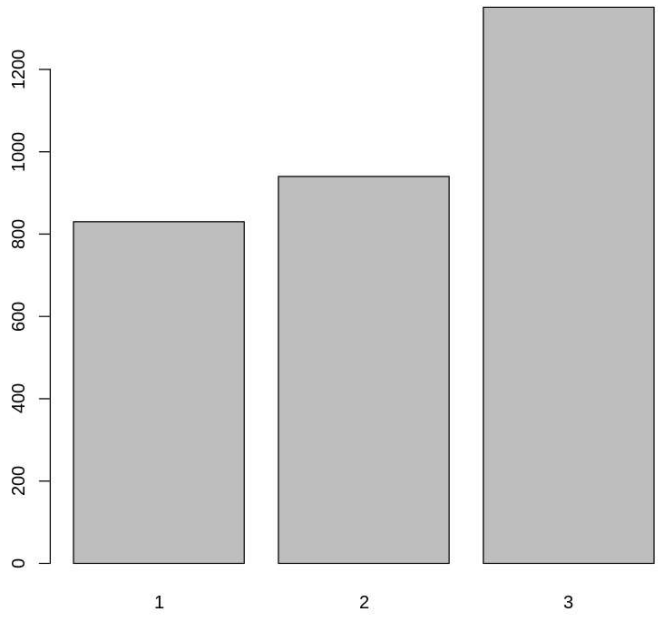
In our analysis, we chose to divide the data into three clusters (k = 3).

```
# Perform k-means clustering
library(cluster)

cluster_data <- all_num_clean[, c("vendor_project", "product")]

k <- 3  # Number of clusters
kmeans_model <- kmeans(cluster_data, centers = k)

cluster_assignments <- kmeans_model$cluster

cluster_counts <- table(cluster_assignments)
cluster_counts

plot(cluster_data, col = cluster_assignments, pch = 16)
```

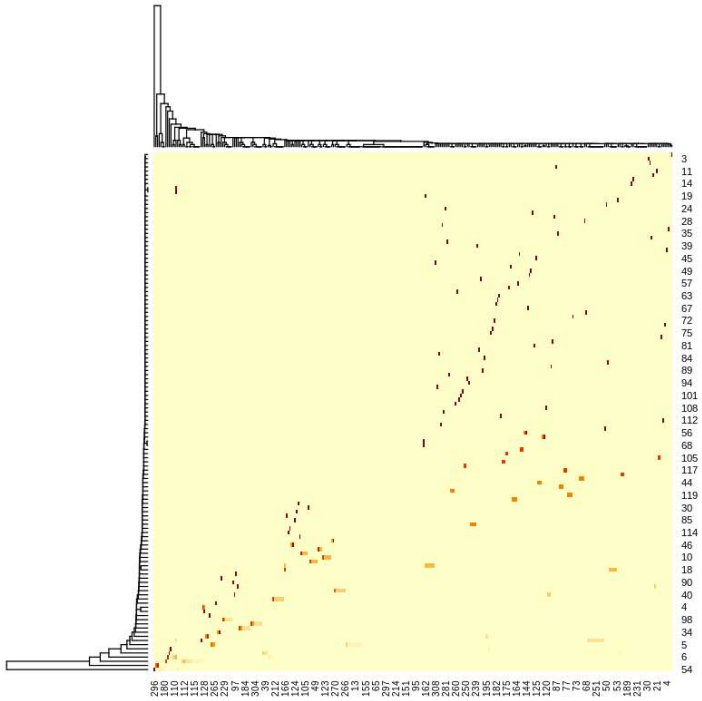
```
cluster_assignments
   1    2    3
 830  940 1351
```




```
barplot(cluster_counts)
```



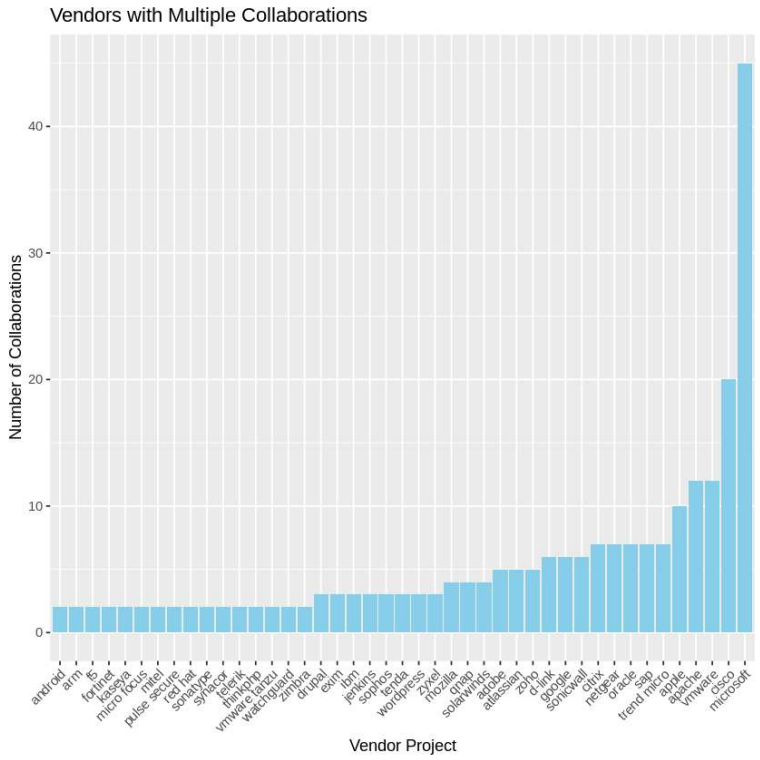

```
heatmap(table(cluster_data$vendor_project, cluster_data$product))
```



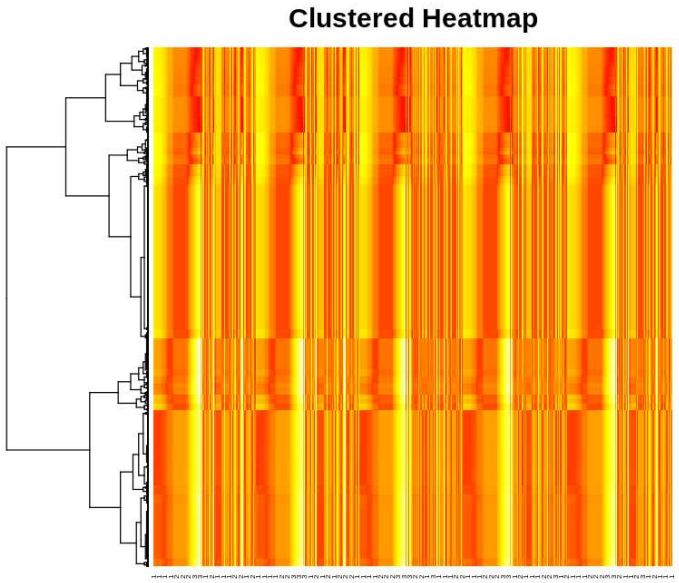
- Vendors and products with a higher frequency of vulnerabilities are represented by darker shades or higher cell values.
- Clusters of vendors and products that share a significant number of vulnerabilities appear as contiguous blocks of dark cells.
- Certain vendors or products might exhibit a higher vulnerability rate across multiple clusters, indicating their association with different types of vulnerabilities.

- Vendor Collaboration: Which vendors have collaborated on multiple projects or products? Are there any patterns or clusters of vendors working together?

```
vendor_collaboration <- all_imp %>%
  group_by(vendor_project) %>%
  summarise(collaborations = n_distinct(product)) %>%
  filter(collaborations > 1) %>%
  arrange(desc(collaborations))

ggplot(vendor_collaboration, aes(x = reorder(vendor_project, collaborations), y = collaborations)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(x = "Vendor Project", y = "Number of Collaborations") +
  ggtitle("Vendors with Multiple Collaborations") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
# Compute distance matrix
dist_matrix <- dist(all_num_clean$vendor_project)

# Perform hierarchical clustering
hc <- hclust(dist_matrix)

# Cut the dendrogram to obtain clusters
clusters <- cutree(hc, k = 3)

# Create a heatmap of the distance matrix with cluster assignments
heatmap(as.matrix(dist_matrix), Rowv = as.dendrogram(hc), Colv = NA, col = heat.colors(256),
        main = "Clustered Heatmap", labRow = "", labCol = clusters)
```

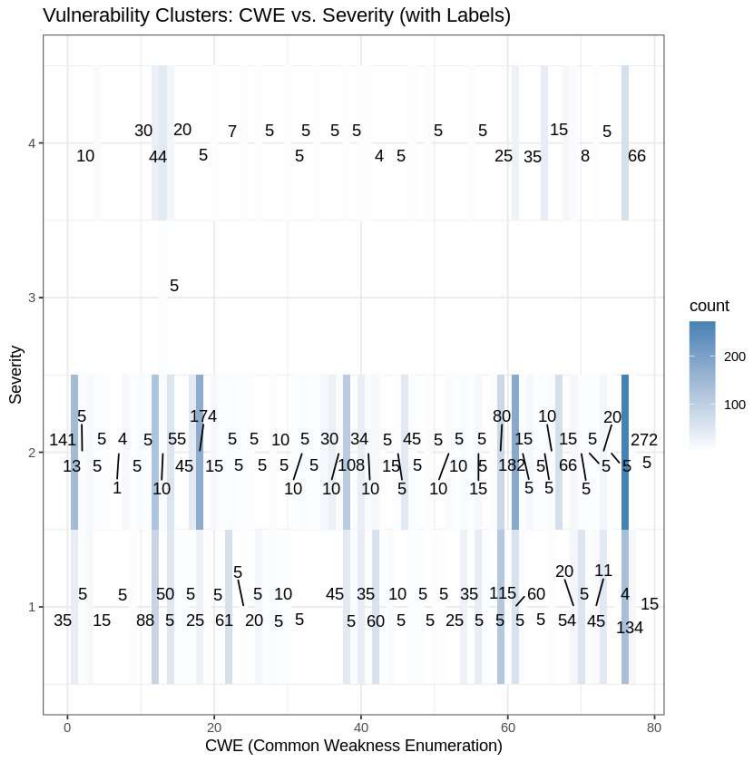


The hierarchical clustering and heatmap analysis help to identify clusters of vendor collaborations based on the similarity of their projects or products.

Vulnerability Networks: Identify Networks Or Clusters Of Vulnerabilities Based On Common Attributes

```
# Subset the data to include only the relevant columns
subset_df <- all_num_clean[, c("severity", "cwe", "vector")]


# CWE vs. Severity
cwe_severity_labels_plot <- subset_df %>%
  group_by(cwe, severity) %>%
  summarise(count = n(), .groups = 'drop') %>%
  ggplot(aes(x = cwe, y = severity, label = count, fill = count)) +
  geom_tile() +
  geom_text_repel() +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(x = "CWE (Common Weakness Enumeration)", y = "Severity") +
  ggtitle("Vulnerability Clusters: CWE vs. Severity (with Labels)") +
  theme_bw()

cwe_severity_labels_plot
```

```
Warning message:
“ggrepel: 8 unlabeled data points (too many overlaps). Consider increasing max.overlaps”
```



There is a wide distribution of vulnerabilities across different CWE categories and severity levels. Severity level 2 appears to be the most common severity level across various CWE categories. Some CWE categories have vulnerabilities that span multiple severity levels, indicating a diverse range of risks associated with those weaknesses. There are certain CWE categories where severity level 4 vulnerabilities are more prevalent, suggesting critical security risks in those specific areas.

```
vector_severity_labels_plot2 <- subset_df %>%
  group_by(vector, severity) %>%
  summarise(count = n(), .groups = 'drop')

vector_severity_labels_plot2
```

A tibble: 9 × 3

vector <dbl>	severity <dbl>	count <int>
1	1	5
1	2	20
1	4	25
2	2	731
2	4	60
3	1	1062
3	2	984
3	3	5
3	4	229

```
# Vector vs. Severity
vector_severity_labels_plot <- subset_df %>%
  group_by(vector, severity) %>%
  summarise(count = n(), .groups = 'drop') %>%
  ggplot(aes(x = vector, y = severity, label = count, fill = count)) +
  geom_tile() +
  geom_text_repel(max.overlaps = Inf) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(x = "Vector", y = "Severity") +
  ggtitle("Vulnerability Clusters: Vector vs. Severity (with Labels)") +
  theme_bw()

vector_severity_labels_plot
```

Vulnerability vector 1 is associated with severity levels 1, 2, and 4. It has a relatively low count for severity 1 and a higher count for severity 2 and 4. Vulnerability vector 2 is primarily associated with severity 2, with a significantly high count. Vulnerability vector 3 is associated with all severity levels, with a higher count for severity 1, 2, and 4.

These findings suggest that different vulnerability vectors may have varying impacts in terms of severity. Vulnerability vector 2 stands out as having a higher count of vulnerabilities at severity level 2, indicating a potentially widespread impact. On the other hand, vulnerability vector 1 shows a mix of vulnerabilities across severity levels, suggesting a diverse range of risks. Vector 3 has a very significant association with severity levels 1 and 2 also.

Are there any “super spreader” vulnerabilities that affect multiple vendors/products?