

# Analyse af logfiler

*"I skal åbne logfilerne og lave en rapport over aktiviteten på webserveren. Rapporten skal indeholde en optælling af aktive ip-adresser pr døgn. Plot med de mest aktive. whois-info på den mest aktive. Gruppering på 404, herunder en beskrivelse af "mistænksomme" requests. Det værste der kan ske, er at en mistænksom request returnerer 200. Overvej hvordan man kan fange dem."*

Indlæsning af log-filen (terminal) For at kunne analysere webserveraktiviteten som beskrevet i opgaven, er det nødvendigt at downloade og placere logfilerne korrekt, så de er klar til videre behandling. Dette kræver, at vi organiserer vores arbejdsfiler (hvilket vi gør via terminalen) og gør dem let tilgængelige for senere brug i R. Først navigeres til den mappe, hvor arbejdet med logfilerne skal foregå. I dette tilfælde er det mappen OLA4, som findes i strukturen.

## Indlæsning af log-filen (terminal)

For at kunne analysere webserveraktiviteten som beskrevet i opgaven, er det nødvendigt at downloade og placere logfilerne korrekt, så de er klar til videre behandling. Dette kræver, at vi organiserer vores arbejdsfiler (hvilket vi gør via terminalen) og gør dem let tilgængelige for senere brug i R. Først navigeres til den mappe, hvor arbejdet med logfilerne skal foregå. I dette tilfælde er det mappen OLA4, som findes i strukturen:

```
# Cd Documents/Dataprojekter/GitHubs/logfiler_
```

**cd** (Change Directory) bruges til at skifte til den specifikke mappe, hvor logfilerne skal placeres. Dette sikrer, at det videre arbejde foregår i den korrekte kontekst. For at holde filstrukturen organiseret, oprettes en separat mappe under OLA4 med navnet logfiler. Dette gøres med følgende kommando:

```
# mkdir logfiler
```

**mkdir** (Make Directory) bruges til at oprette en ny mappe. Navnet logfiler blev valgt for at tydeliggøre, at denne mappe udelukkende skal indeholde logfiler.

Flytning af logfilen til den nyoprettede mappe: Den downloadede fil log.tar, som ligger i standardmappen Downloads, skal flyttes til den nyoprettede mappe logfiler. Flytningen udføres med følgende kommando:

```
mv ~/Downloads/log.tar  
/Users/sevimkilinc/Documents/Dataprojekter/Github/logfiler_/log
```

**mv (Move)** bruges til at flytte eller omdøbe en fil. Her flyttes log.tar fra Downloads til den specifikke sti, hvor logfilerne skal opbevares.

Udpakning af logfilen: Når filen er placeret i mappen logfiler, navigeres der til mappen for at udpakke log.tar. Dette gøres med:

```
cd / User/ Sevimkilin/  
Documents/Dataprojekter/Github/logfiler_/log
```

Udpakning udføres derefter med:

```
tar -xvf log-tar
```

tar kommandoen anvendes til at håndtere arkivfiler. Flagene -xvf står for:

- -x: Udpakning (extract).
- -v: Vis detaljerede oplysninger under processen(verbose).
- -f: Specificerer filen, der skal behandles (her log.tar).

Slutvis kontrollerer vi zipfilen er udpakket korrekt vha. af "ls" (list) funktionen:

```
#  
access.log  
access.log.1  
access.log.2.gz  
access.log.3.gz  
...  
log.tar
```

Vi kan nu se, at log.tar er udpakket ud fra overstående output.

## 1. Dataindsamling

### Formål:

Hente logdata fra en webserver og samle dem til én datakilde.

- Setwd(): Bruges til at angive stien til mappen med logfiler
- list.files(): Identificerer alle filer, der matcher mønsteret "access\*", dvs. filer relateret til adgangslogfiler.
- lapply(Logfiles, readLines): Læser indholdet af hver logfil linje for linje og gemmer dem som en liste i variabelen log\_content.
- do.call(c, log\_content): Samler alle linjer fra alle logfiler til én samlet variabel all\_logs

```
setwd("/Users/sevimkilinc/Documents/Dataprojekter/OLA/OLA4/logfiler")
file.exists("/Users/sevimkilinc/Documents/Dataprojekter/OLA/OLA4/logfiler") |
Logfiles <- list.files(pattern = "access*", path = ".")
log_content <- lapply(Logfiles, readLines)

# Kombiner indholdet af alle logfiler
all_logs <- do.call(c, log_content)
```

## 2. Datarensning:

**Formål:** Uddrage de nødvendige informationer fra de rå logfiler. - extract\_log\_data(): En "selv defineret" funktion, der udtrækker de centrale elementer fra hver linje i logfilen:

- IP-adresse: Den første del af linjen.
- Statuskode: Den HTTP-statuskode, der angiver succes eller fejl (fx 200, 404).
- Path: URL-stien, der blev forespurgt (fx /login, /wp-admin).
- Tidspunkt: Hvornår forespørgslen blev foretaget.

- `t(sapply(all_logs, extract_log_data))`: Anvender funktionen på hver linje i logfilen og gemmer resultatet i en matrix

```
extract_log_data <- function(raw_log) {  
  ip <- str_extract(raw_log, "^\\S+") # Første ord: IP-adresse  
  status <- str_extract(raw_log, "\\s\\d{3}\\s") # HTTP-statuskode  
  path <- str_extract(raw_log, "\\\"(GET|POST|HEAD)\\s(?:.?)\\sHTTP") # Path fra forespørgsel  
  time <- str_extract(raw_log, "\\[(.?)\\]") # Tidsstempel  
  return(c(ip, time, path, status))  
}  
  
parsed_logs <- t(sapply(all_logs, extract_log_data))  
colnames(parsed_logs) <- c("IP", "Time", "Path", "Status")  
logs_prep <- as.data.frame(parsed_logs, stringsAsFactors = FALSE)
```

### 3. Dataforberedelse

**Formål:** Klargøre data til analyse ved at formatere og tilføje nye variable.

- **Dato og tid:**
  - `sub(":(.*)", "", logs_prep$Time)`: Fjerner tidsoplysninger (hh:mm:ss) fra datoen.
  - `gsub("\\[", "", logs_prep$Date)`: Fjerner specialtegn som venstre parenteser.
  - `as.Date(logs_prep$Date, format = "%d/%b/%Y")`: Konverterer datoer til en standardiseret Date-type.
- **Nøjagtig tid:**
  - `sub("^.*:(\\d{2}:\\d{2}:\\d{2}).*", "\\1", logs_prep$Time)`: Uddrager kun tidskomponenten (hh:mm:ss) fra loglinjen.

### 4. Dataanalyse

**Formål:** Oprette en struktureret og analyserbar version af logdata.

- **structured\_logs <- logs\_prep**: Den færdige data gemmes i en ny variabel `structured_logs`, som nu kan bruges til videre analyser (fx optælling af IP'er, mistænksomme requests, analyse af statuskoder osv.).

#### Aktive IP adresser pr. døgn

- **Optælling af aktive IP-adresser pr. dag:**
  - Vi har grupperet forespørgslerne efter dato og talt antallet af unikke IP-adresser pr. dag. Dette giver et overblik over, hvordan aktiviteten varierer fra dag til dag, og hvilke datoer der har højeste aktivitet.

- **Identificering af de mest aktive IP-adresser:**

- Ved at tælle antallet af forespørgsler for hver IP-adresse har vi identificeret de mest aktive IP'er. Disse IP'er kan stå for en betydelig del af trafikken og kan være relevante for videre undersøgelse.

**Bemærkninger om optælling af aktive IP-adresser pr. dag:**

- Denne fremgangsmåde giver et klart overblik over, hvor mange forskellige enheder der har kommunikeret med serveren i løbet af en given dag. Det er vigtigt at bemærke, at hver unik IP kun tælles én gang pr. dag, uanset hvor mange gange den samme IP har sendt forespørgsler til serveren.
- **Kodeanvendelse:** Funktionen `aggregate()` anvendes, hvilket sikrer, at hver IP kun tælles én gang pr. dato.

Denne struktur gør det lettere at navigere i dokumentationen og præsenterer informationen på en klar og organiseret måde.

```
active_ips_per_day <- aggregate(IP ~ Date, data = structured_logs, FUN = function(x) length(unique(x)))  
colnames(active_ips_per_day) <- c("Date", "UniqueIPs")
```

Nedenstående viser resultatet dataframen "active\_ips\_per\_day" som viser, hvordan aktiviteten varierer dagligt, og hvilke datoer der har højest IP-aktivitet:

OLA4.opg3.R		active_ips_per_day	
	Date	UniqueIPs	
1	2022-11-08	167	
2	2023-10-26	143	
3	2023-10-27	165	
4	2023-10-28	163	
5	2023-10-29	139	
6	2023-10-30	157	
7	2023-10-31	137	
8	2023-11-01	198	
9	2023-11-02	246	
10	2023-11-03	179	
11	2023-11-04	161	
12	2023-11-05	74	
13	2023-11-06	86	
14	2023-11-07	93	
15	2023-11-08	91	
16	2023-11-09	34	

### Identificering af de mest aktive IP-adresser:

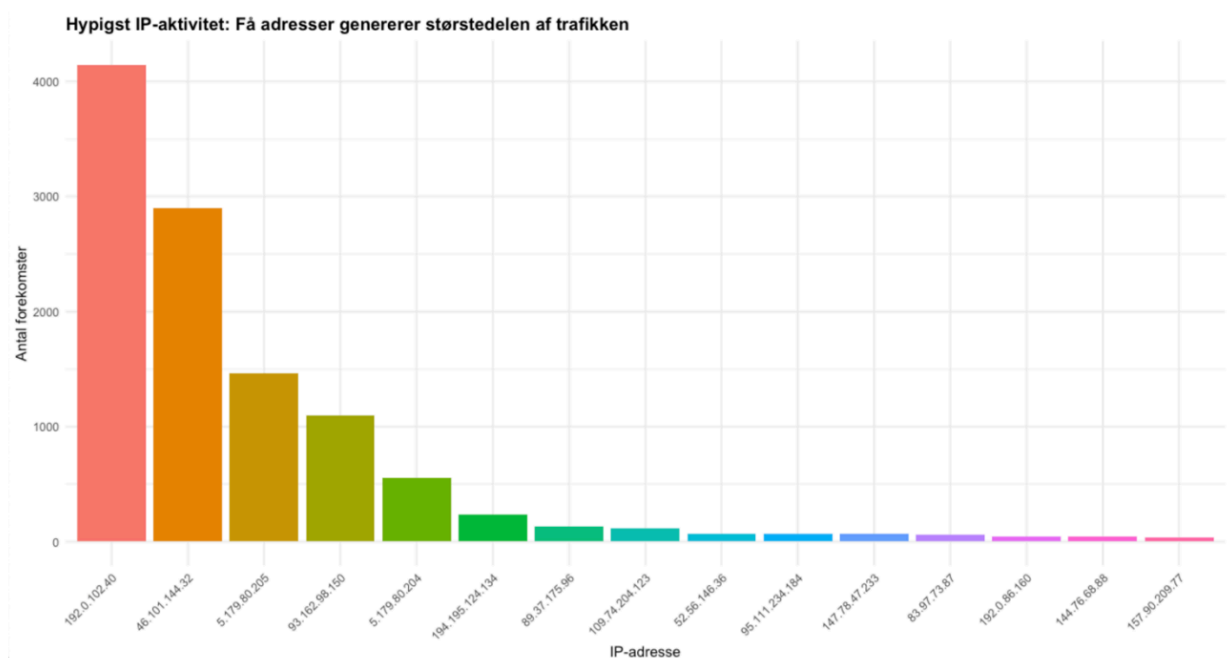
Vi har sorteret de samlede forekomster af hver IP-adresse i logfilen, hvilket giver et klart billede af de IP'er, der genererer størstedelen af trafikken. Ved at bruge funktionen `table()` har vi talt antallet af forespørgsler for hver IP, og de mest aktive IP'er er identificeret ved at sortere i faldende rækkefølge:

```
# Optælling af forekomster for hver IP-adresse
IP_antal <- table(structured_logs$IP)
sorted_IP_antal <- sort(IP_antal, decreasing = TRUE)
Antal_forekomster_IP <- as.data.frame(sorted_IP_antal)
colnames(Antal_forekomster_IP) <- c("IP", "Antal")
```

Overstående resulteret i nedenstående dataframe "Antal\_forekomster\_IP":

OLA4.opg3.R* x Antal_forekomster_IP x		
Filter		
	IP	Antal
1	192.0.102.40	4145
2	46.101.144.32	2902
3	5.179.80.205	1464
4	93.162.98.150	1094
5	5.179.80.204	555
6	194.195.124.134	233
7	89.37.175.96	128
8	109.74.204.123	115
9	52.56.146.36	70
10	95.111.234.184	66
11	147.78.47.233	64
12	83.97.73.87	58
13	192.0.86.160	44
14	144.76.68.88	42
15	157.90.209.77	39
16	159.65.197.63	38
17	162.240.239.98	36
18	64.227.108.223	30
19	103.237.86.234	28
20	157.245.40.165	28
21	192.0.86.85	28

For at visualiserer aktive IP- adresser pr. døgn, har vi udvalgt op 15 trafikerede IP-adresser:





**PLOT 1: GRAFEN VISER, AT EN LILLE GRUPPE IP-ADRESSER STÅR FOR LANGT STØRSTEDELEN AF TRAFIKKEN, HVOR DEN MEST AKTIVE IP ADRESSE ALENE HAR OVER 4000 REQUEST.**

Plot af mest aktive IP adresse Nedenstående kode illustrerer, hvordan vi ved hjælp af GET()-funktionen fra httr-pakken sender en GETforespørgsel til den konstruerede URL "http://ipinfo.io/". Denne forespørgsel anmoder om data fra ipinfo.io for den specifikke IP-adresse. Funktionen fromJSON(rawToChar(response\$content)) konverterer de modtagne rådata fra JSON-format til en liste "response", hvilket gør det nemmere at arbejde med dataene i R.

```
Top_1_active_IP <- "192.0.102.40"

# Hent oplysninger for IP-adressen
response <- GET(paste0("https://ipinfo.io/", Top_1_active_IP, "/json"))
data <- fromJSON(rawToChar(response$content))
cat(paste("IP:", data$ip, "\nCITY:", data$city, "\nREGION:",
          data$region, "\nCOUNTRY:", data$country, "\nLOCATION:", data$loc, "\n"))
```

Efter vores API-kald via IPinfo kunne vi indhente detaljerede oplysninger om den mest aktive IP-adresse: 192.0.102.40 Disse data inkluderer blandt andet geografisk placering (latitude og longitude) samt by, region og land (bliver printet i consolen ud af overstående loop vha. cat funktionen, fx koordinaterne som er med til at definere latitude og longitude).

IP Location	 United States Marina Del Rey Early Registration Addresses
ASN	 AS2635 AUTOMATTIC, US (registered Oct 01, 2012)
Whois Server	whois.arin.net
IP Address	192.0.102.40

Denne server, tilknyttet IP-adressen 192.0.102.40, er registreret under AUTOMATTIC, en velkendt amerikansk hostingudbyder, og er placeret i Marina Del Rey, USA. Det høje aktivitetsniveau gør serveren til den mest aktive IP-adresse i vores data, hvilket potentielt kan skyldes legitime forespørgsler, hvis serveren f.eks. bruges til en populær tjeneste som WordPress.

Dog kan det også indikere mistænkelig aktivitet, hvis serveren er blevet kompromitteret eller misbrugt til at generere unormal trafik. Dette aspekt vil blive analyseret og gennemgået nærmere i en senere del af opgaven. Her fokuserer vi alene på at identificere, at 192.0.102.40 er den mest aktive IP-adresse i vores analyse.

### Mistænksomme request:

Vi har valgt at fokusere på antallet af forespørgsler ud fra statuskode 404 (da gruppering af denne statuskode både kræves i opgaven og kan være indikativ for mistænkelig adfærd) som den primære



indikator for mistænkelig aktivitet. Statuskode 404 afslører hurtigt mønstre, der ofte forbindes med bot- og hackerangreb. Højfrekvente forespørgsler mod paths som wp-admin, .env, og git/config indikerer typisk automatiserede angreb eller forsøg på at udnytte sårbarheder. En alternativ tilgang kunne have været at analysere forespørgsler baseret på nøgleord som wp, php, og git ved at ekstrahere disse oplysninger fra path-kolonnen.

## Metodisk tilgang til analyse af mistænkelige aktiviteter:

For at identificere potentielt mistænkelig aktivitet i logfilerne har vi anvendt en struktureret tilgang, der fokuserer på HTTP-forespørgsler med statuskode 404. Denne metode gør det muligt at fremhæve mønstre, der kan indikere automatiserede angreb eller forsøg på at finde sårbare filer. Den overordnede fremgangsmåde kan beskrives som følger:

### Filtrering af statuskode 404:

Logfilerne filtreres, så kun forespørgsler med statuskode 404 inkluderes. Dette isolerer de relevante data:

```
logs_404 <- structured_logs[structured_logs$Status == 404, ]
```

### Gruppering og optælling

Dataene grupperes efter kombinationen af Path og IP, hvorefter antallet af 404-fejl (Count404) tælles for hver kombination. Denne tilgang giver mulighed for at identificere de specifikke IP-adresser og paths, der oftest er forbundet med fejl. Det høje antal forespørgsler mod ressourcer som .env, wp-admin, og .git/config kan indikere automatiserede scanninger eller angrebsmønstre.

```
Path_404_summary <- aggregate(Status ~ Path + IP, data = logs_404, FUN = length)
colnames(Path_404_summary) <- c("Path", "IP", "Count404")
```

Resultatet sorteres efter antallet af fejl for at fremhæve de mest mistænksomme kombinationer, hvilket udgør tabellen sorted\_path\_404:



		IP	Count404
1	"POST /wp-admin/admin-ajax.php HTTP	93.162.98.150	331
2	"POST /wp-admin/admin-ajax.php HTTP	5.179.80.204	41
3	"GET /actuator/gateway/routes HTTP	83.97.73.87	25
4	"GET /.git/config HTTP	170.64.220.120	15
5	"GET /robots.txt HTTP	157.90.209.77	14
6	"GET /docker-compose.yml HTTP	159.100.22.187	14
7	"GET /sitemap HTTP	157.90.209.77	12
8	"GET /sitemap.txt HTTP	157.90.209.77	12
9	"POST /core/.env HTTP	162.240.239.98	12
10	"GET /favicon.ico HTTP	5.179.80.205	12

Ved at inkludere tællingerne (Count404) prioriteres de mest mistænksomme kombinationer, hvilket hjælper med at fokusere på de IP-adresser og paths, der er mest sandsynlige kilder til scanninger eller angreb.

### Udvælgelse af mistænksomme IP'er:

- Listen over unikke IP'er genereres for at sikre, at hver IP kun inkluderes én gang.
- De 10 IP-adresser med flest 404-fejl vælges fra den tidligere grupperede og sorterede tabel (sorted\_path\_404), da disse vurderes til at være dem med højeste frekvens statuskode 404.

```
Top_10_suspicious <- head(sorted_path_404, 10)
suspicious_ips <- unique(Top_10_suspicious$IP)
```

Nedestående funktion muliggør, at vi kan indhente information på vores top 10 mistænklige IP-adresser. Find forklaring i tidligere opgave om aktivitet.

```
# Funktion til at hente IP-data
fetch_ip_details <- function(ip) {
  response <- GET(paste0("https://ipinfo.io/", ip, "/json"))
  if (status_code(response) == 200) {
    data <- fromJSON(rawToChar(response$content))
    loc <- strsplit(data$loc, ",")[[1]]
    return(data.frame(
      IP = ip,
      City = data$city,
      Region = data$region,
      Country = data$country,
      Latitude = as.numeric(loc[1]),
      Longitude = as.numeric(loc[2]),
      stringsAsFactors = FALSE
    ))
  }
  return(NULL)
}

# Hent data for de 10 mistænksomme IP'er
top_10_suspicious_IP_details <- do.call(rbind, lapply(top_10_suspicious_IPs, fetch_ip_details))
```

Vi kører nedenstående kode for at behandle data fra IPinfo. Under analysen blev det tydeligt, at visse IPadresser rapporteres med samme geografiske koordinater (longitude og latitude). Disse duplikater kan skyldes delte servere eller netværk. For at sikre en præcis analyse fjernes disse dubletter, så hver lokation kun fremgår én gang i datasættet. Efter denne behandling står vi tilbage med 7 unikke IP-adresser, som repræsenterer de mest mistænksomme lokationer.

```
# Hent data for de mistænksomme IP'er
top_suspicious_IP_details <- do.call(rbind, lapply(top_10_suspicious_IPs, fetch_ip_details))

# Fjern dubletter baseret på Latitude og Longitude
top_suspicious_IP_details <- top_suspicious_IP_details[!duplicated(top_suspicious_IP_details[, c("Latitude", "Longitude")]), ]
```

vi ender dermed med følgende dataframe:

	IP	City	Region	Country	Latitude	Longitude
1	93.162.98.150	Copenhagen	Capital Region	DK	55.6759	12.5655
3	83.97.73.87	Moscow	Moscow	RU	55.7522	37.6156
4	170.64.220.120	Sydney	New South Wales	AU	-33.9092	151.1940
5	157.90.209.77	Falkenstein	Saxony	DE	50.4779	12.3713
6	159.100.22.187	Frankfurt am Main	Hesse	DE	50.1155	8.6842
7	162.240.239.98	Provo	Utah	US	40.2338	-111.6585

Ud fra overstående dataframe, kan vi nu lave en interaktiv graf vha. Leaflet pakken i R



Figur 2: Kortet viser globale hotspots for potentielt skadelig IP-aktivitet baseret på de 7 mest mistænksomme IP-adresser. Hver marker repræsenterer en unik lokation, hvorfra der er registreret et højt antal 404-fejl på specifikke ressourcer. Aktiviteten

## Andre mønstre mistænkelige request

Som tidligere nævnt ville vi undersøge følgende IP-adresse 192.0.102.40. Da denne IP-adresse ikke havde en eneste statuskode 404, men derimod kun 200, blev den ikke medtaget i den tidligere analyse af mistænksomme forespørgsler. Den inddrages her for at demonstrere, hvordan mistænkelig aktivitet kan variere i form og mønster.

Den mistænkelige aktivitet ved IP-adressen 192.0.102.40 kan sammenfattes således:

- Ekstrem høj aktivitet: IP-adressen genererede 4115 forespørgsler, hvilket er en betydelig del af de samlede forespørgsler i logfilen. Blandt de 15.000 samlede forespørgsler var 12.289 markeret som statuskode 200, svarende til 81,9% af alle forespørgsler. At én IP-adresse bidrager så markant til dette antal er i sig selv mistænkeligt.

OLA4.opg3.R\* x structured\_logs x

Filter

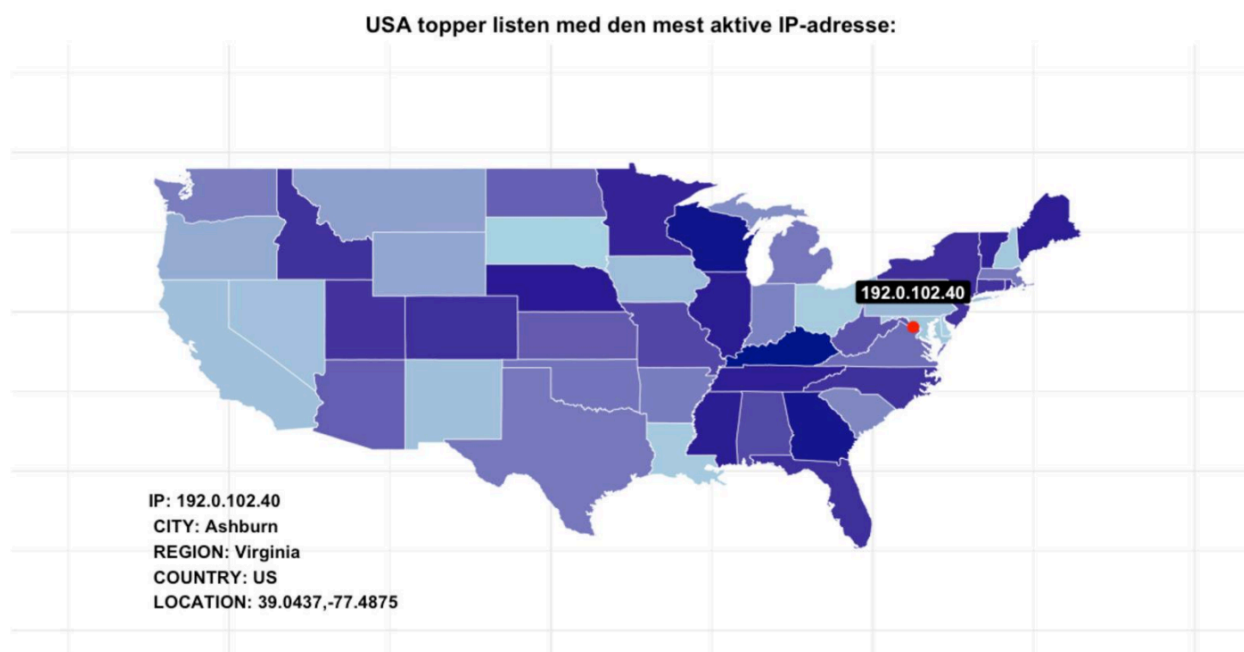
	IP	Time	Path	Status	Date	Exacttime
1	192.0.102.40	[09/Nov/2023:00:04:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:04:38
2	192.0.102.40	[09/Nov/2023:00:09:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:09:38
3	192.0.102.40	[09/Nov/2023:00:14:39 +0000]	"HEAD / HTTP	200	2023-11-09	00:14:39
4	192.0.102.40	[09/Nov/2023:00:19:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:19:38
5	192.0.102.40	[09/Nov/2023:00:24:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:24:38
6	192.0.102.40	[09/Nov/2023:00:29:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:29:38
7	192.0.102.40	[09/Nov/2023:00:34:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:34:38
8	192.0.102.40	[09/Nov/2023:00:39:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:39:38
9	192.0.102.40	[09/Nov/2023:00:44:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:44:38
10	192.0.102.40	[09/Nov/2023:00:49:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:49:38
11	192.0.102.40	[09/Nov/2023:00:54:37 +0000]	"HEAD / HTTP	200	2023-11-09	00:54:37
12	192.0.102.40	[09/Nov/2023:00:59:38 +0000]	"HEAD / HTTP	200	2023-11-09	00:59:38
13	192.0.102.40	[09/Nov/2023:01:04:38 +0000]	"HEAD / HTTP	200	2023-11-09	01:04:38
15	192.0.102.40	[09/Nov/2023:01:09:37 +0000]	"HEAD / HTTP	200	2023-11-09	01:09:37
16	192.0.102.40	[09/Nov/2023:01:14:38 +0000]	"HEAD / HTTP	200	2023-11-09	01:14:38
17	192.0.102.40	[09/Nov/2023:01:19:38 +0000]	"HEAD / HTTP	200	2023-11-09	01:19:38
18	192.0.102.40	[09/Nov/2023:01:24:38 +0000]	"HEAD / HTTP	200	2023-11-09	01:24:38
19	192.0.102.40	[09/Nov/2023:01:29:39 +0000]	"HEAD / HTTP	200	2023-11-09	01:29:39
20	192.0.102.40	[09/Nov/2023:01:34:36 +0000]	"HEAD / HTTP	200	2023-11-09	01:34:36
21	192.0.102.40	[09/Nov/2023:01:39:37 +0000]	"HEAD / HTTP	200	2023-11-09	01:39:37
22	192.0.102.40	[09/Nov/2023:01:44:38 +0000]	"HEAD / HTTP	200	2023-11-09	01:44:38
23	192.0.102.40	[09/Nov/2023:01:49:37 +0000]	"HEAD / HTTP	200	2023-11-09	01:49:37
24	192.0.102.40	[09/Nov/2023:01:54:37 +0000]	"HEAD / HTTP	200	2023-11-09	01:54:37
26	192.0.102.40	[09/Nov/2023:01:59:38 +0000]	"HEAD / HTTP	200	2023-11-09	01:59:38

Showing 1 to 24 of 4 145 entries. 6 total columns (filtered from 15 000 total entries)

Showing 1 to 24 of 4,145 entries, 6 total columns (filtered from 15,000 total entries)

- Forespørgselstype HEAD: Alle forespørgsler fra denne IP var af typen HEAD, som kun henter metadata fra serveren uden at indlæse selve indholdet (tag også de korte tidsintervaller i betragtning). Dette er usædvanligt for normal trafik, da typiske forespørgsler ofte kombinerer HEAD med GET. En sådan ensidig forespørgselstype er ofte karakteristisk for automatiserede scripts eller scanninger.
- Indikation på automatiseret aktivitet: Kombinationen af et stort antal forespørgsler (alle med HEAD) og fraværet af fejlstatusser (f.eks. 404) kan indikere, at denne IP er en del af et script eller en bot, der systematisk scanner serveren.

Denne analyse demonstrerer, hvordan mistænksom aktivitet ikke nødvendigvis kræver fejlstatusser, men kan identificeres gennem unormale forespørgselsmønstre og en markant overvægt af forespørgselstypen HEAD.



GRAF 3: GRAFEN VISER, AT USA, REPRÆSENTERET VED IP-ADRESSEN 192.0.102.40, TOPPER LISTEN SOM DEN MEST AKTIVE IPADRESSE MED LOKATION I ASHBURN, VIRGINIA. DENNE IP STÅR FOR EN BETYDELIG ANDEL AF TRAFIKKEN OG VIL BLIVE UNDERSØGT NÆRMERE SENERE I OPGAVER FOR POTENTIelt MISTÆNKELIG AKTIVITET

Denne specifikke IP-adresse nævnes som en del af en senere vurdering af mistænkelig aktivitet, hvor dens trafik vil blive diskuteret i forhold til, om den kan betragtes som mistænkelig eller legitim. Dette vil blive behandlet i en efterfølgende del af opgaven.

## Ekstra bemærkelser til mistænksomme request

### Statuskode 200

- **Normal funktion:** Statuskoden 200 betyder, at serveren har behandlet en forespørgsel korrekt og returneret den ønskede ressource, såsom adgang til en hjemmeside. Dette indikerer, at alt fungerer som forventet.
- **Udfordringer ved mistænksomme requests:** Hvis en mistænksom request (f.eks. forespørgsler til administrative endpoints som /wp-admin, /login, eller .php-filer) returnerer statuskoden 200, kan det indikere, at hackere eller bots potentielt har fået uautoriseret adgang til ressourcer, som de ikke burde have adgang til.

## ***Mistænkelige mønstre og IP-aktivitet***

### ***1. Gentagne forespørgsler fra en IP:***

- Hvis en IP-adresse gentagne gange sender forespørgsler til forskellige paths og modtager statuskoden 200, kan dette være mistænkeligt. Dette gælder især, hvis paths er kendt for at være administrative eller sensitive.

### ***2. Eksempler på mistænkelige paths:***

- Paths, der normalt kun er tilgængelige for autoriserede brugere:
  - /wp-admin
  - /login
  - /config.php

### ***3. Tidsmønstre:***

- Mistænkelige requests forekommer ofte uden for normal arbejdstid eller med meget høj frekvens.
- Brug af tidsstempler kan hjælpe med at identificere og analysere normale vs. unormale mønstre.

## ***Anbefalinger for at identificere mistænksomme requests***

- Analyser paths: Overvåg requests til administrative eller følsomme paths og tjek, om de returnerer statuskoden 200 uden at være autoriseret.
- Overvåg IP-aktivitet: Identificér IP'er med gentagne requests til mistænkelige paths.
- Brug tidsmønstre: Sammenlign tidspunkter for requests med normale arbejdstider for at finde uregelmæssigheder.