My methodology:

Q1 How many total events were logged in the file?

Q2 What is the full timestamp of the last event in the file?

Q3 What are the top 3 IP addresses found within the log file (in descending order)?

Format: IP, IP, IP

Q4 What is the count of the IP address with the highest connections?

Q5 How many different user agent strings are found in the log file?

Q6 Which user agent string sticks out as suspicious?

Q7 What is the line number of the log event that returned an HTTP status code of 404?

Q8 What web-based attack can you identify based on the URL query parameters?

Q9 What is the name of the vulnerable URL parameter?

Q10 The attacker attempted a number of different SQL injection attacks. Based on other indicators in the event logs, what was the full timestamp of the potentially successful attempt?

Important Commands:

$ file test.log

$ head -n 1 test.log

Most IP talks:

$cut test.log -d “ “ -f 1 | sort | uniq -c | grep -v “ 1 “ | sort -nr

Found suspicious User agent:

$ cut challenge.log -d "\"" -f 6 | sort | uniq -c

Find the extract value:

$ grep "Nmap Scripting Engine" access.log

$ grep "Nmap Scripting Engine" access.log | awk '{print $1}' | sort | uniq -c

Brute force trace:

⇒ site redirect 302 301 and most modern web 200 use

$ grep "Mozilla/5.0 (Hydra)" access.log | awk '{print $9}'

$ grep "Mozilla/5.0 (Hydra)" access.log | awk '$9 > 200'

$ grep "Mozilla/5.0 (Hydra)" access.log | grep -v "/login.php"

Grep:

```
$ grep -c "404" access.log

$ grep -n "404" access.log

$ grep -E '%3C|%3E|<|>' access.log

$ grep -E '\.\./|%2E%2E%2F|%2E%2E%2E%2E%2F%2F' access.log
```

🔍 Use `grep` Patterns for Web Attack Detection:

1. SQL Injection (SQLi)

```
grep -Ei "(\%27)|(\')|(\-\-)|(\%23)|(#)|(\bUNION\b)|(\bSELECT\b)|(\bINSERT\b)
|(\bUPDATE\b)|(\bDELETE\b)" access.log
```

2. Cross-Site Scripting (XSS)

```
grep -Ei "(\<script\>)|(%3Cscript%3E)|(\bon\w+=)|(\balert\b)|(\bconfirm\b)|
(\bdocument\.cookie\b)" access.log
```

3. Command Injection

```
grep -Ei "(;|\&\&|\|\||\`|\$\(.*\)|\bcat\b|\bwget\b|\bcurl\b|\bnc\b|\bping\b|\bpyt
hon\b|\bbash\b)" access.log
```

4. Local File Inclusion (LFI)

```
grep -Ei "(\.\./)|(%2e%2e%2f)|(/etc/passwd)|(/proc/self/environ)" access.log
```

5. Remote File Inclusion (RFI)

```
grep -Ei "(http[s]?:\/\/.*\.(php|txt|jpg|gif|png))" access.log
```

6. Path Traversal

```
grep -Ei "(\.\./|\.\.\\)" access.log
```

7. PHP Injection or Execution

```
grep -Ei "(\bphpinfo\b)|(\beval\b)|(\bsystem\b)|(\bexec\b)|(\bpopen\b)|(\bpass
```

```
thru\b)" access.log
```

8. User-Agent Based Attacks (like scanners or bots)

```
grep -Ei "(nikto|acunetix|sqlmap|nessus|nmap|curl|wget)" access.log
```

🔁 Combine With Other Filters

To find only **suspicious GET requests**:

```
grep -Ei "GET .*((\.\./)|(\<script\>)|(\bUNION\b))" access.log
```

To find requests by a specific IP:

```
grep "192.168.1.10" access.log | grep -Ei "(select|union|<script>)"
```

Jq uses:

$ jq . access.log

$ jq 'length' events.json

$ jq 'map(.event)' event.json

Child:

$ jq '.[] | select(.event.PROCESS_ID == 3532)' event.json

$ jq '.[] | select(.event.PROCESS_ID == 3532) | .event.HASH' event.json

Parent:

$ jq '.[] | select(.event.PROCESS_ID == 3532) | .event.PARENT' event.json

$ jq '.[] | select(.event.PROCESS_ID == 3532) | .event.PARENT.PROCESS_ID' event.json

Automation Code:

json_alert.sh ⇒ Use this script for automation