SOC Shift

Types	forensic
CTF	SunCTF

Writeup – SunCTF25 SIEM Log Analysis Challenge

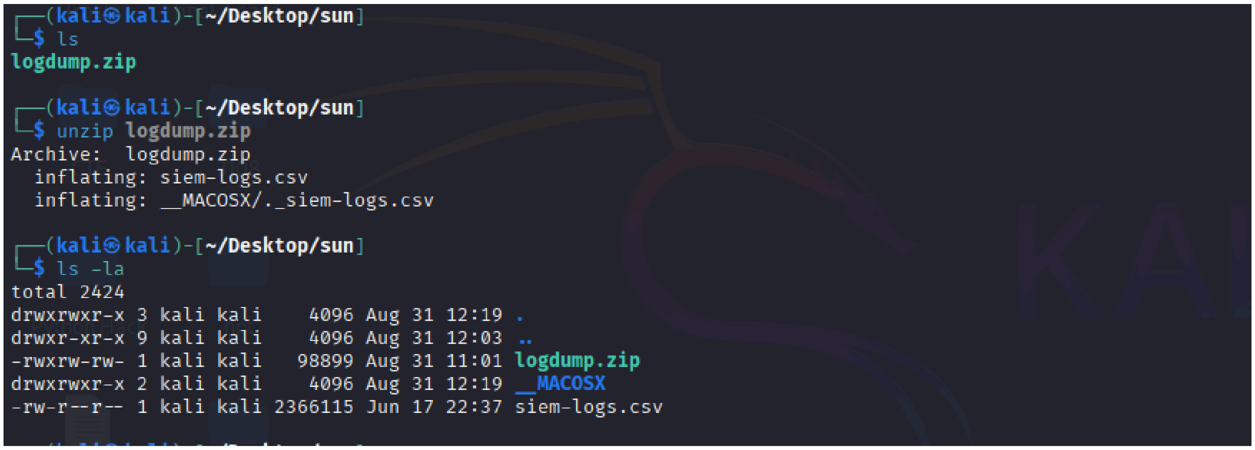
Category

🔍 **Forensics / SIEM Log Analysis**

Challenge Description

We were provided with a log dump (`logdump.zip`) containing SIEM data, alongside other files (`siem-logs.csv`, `photo.jpg`). Our task was to analyze the logs, identify suspicious activity, and eventually recover the hidden flag.

Step 1: Extract the Provided Files



```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ ls
logdump.zip

┌──(kali㉿kali)-[~/Desktop/sun]
└─$ unzip logdump.zip
Archive:  logdump.zip
  inflating: siem-logs.csv
  inflating: __MACOSX/._siem-logs.csv

┌──(kali㉿kali)-[~/Desktop/sun]
└─$ ls -la
total 2424
drwxrwxr-x 3 kali kali    4096 Aug 31 12:19 .
drwxr-xr-x 9 kali kali    4096 Aug 31 12:03 ..
-rwxrw-rw- 1 kali kali   98899 Aug 31 11:01 logdump.zip
drwxrwxr-x 2 kali kali    4096 Aug 31 12:19 __MACOSX
-rw-r--r-- 1 kali kali 2366115 Jun 17 22:37 siem-logs.csv
```

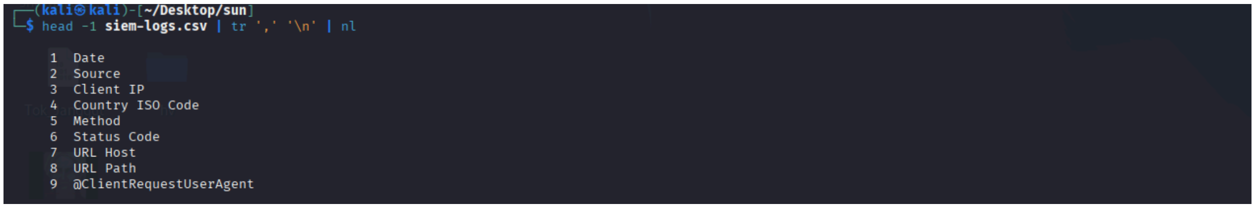
- `ls` → Lists files in the current directory. Here, we see `logdump.zip`.

- `unzip logdump.zip` → Extracts the contents of the ZIP file. This produced `siem-logs.csv` and a hidden macOS metadata folder `__MACOSX/`.
- `ls -la` → Lists files with permissions, ownership, and sizes. We confirm that `siem-logs.csv` is ~2.3MB.

Result: The SIEM logs are ready in `siem-logs.csv`.

Step 2: Inspect the CSV Structure

```
head -1 siem-logs.csv | tr ',' '\n' | nl
```



```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ head -1 siem-logs.csv | tr ',' '\n' | nl

     1  Date
     2  Source
     3  Client IP
     4  Country ISO Code
     5  Method
     6  Status Code
     7  URL Host
     8  URL Path
     9  @ClientRequestUserAgent
```

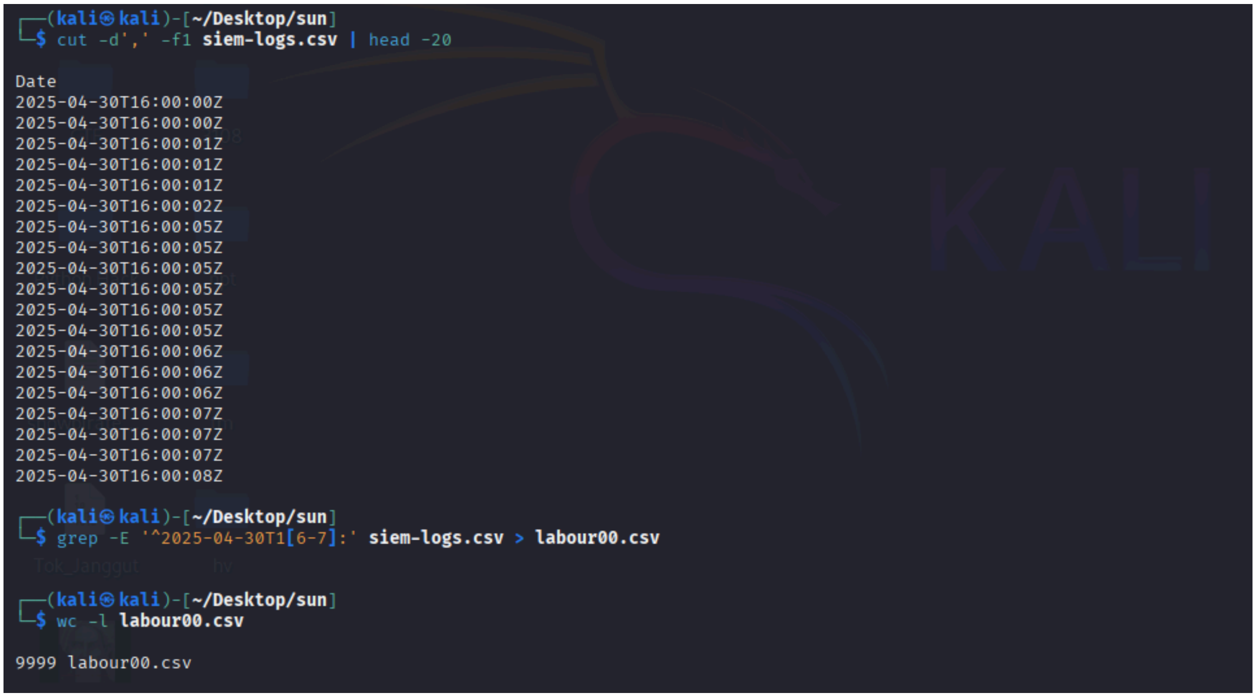
- `head -1` → Shows the first line (CSV headers).
- `tr ',' '\n'` → Replaces commas with newlines, making each column header appear on its own line.
- `nl` → Numbers each line for clarity.

Result: We see the CSV has 9 fields:

1. Date
2. Source
3. Client IP
4. Country ISO Code
5. Method
6. Status Code
7. URL Host
8. URL Path

9. @ClientRequestUserAgent

Step 3: Extract Time Window of Interest



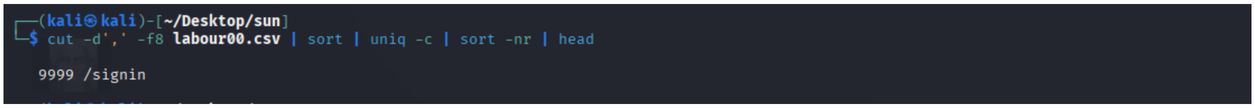
```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ cut -d',' -f1 siem-logs.csv | head -20

Date
2025-04-30T16:00:00Z
2025-04-30T16:00:00Z
2025-04-30T16:00:01Z
2025-04-30T16:00:01Z
2025-04-30T16:00:01Z
2025-04-30T16:00:02Z
2025-04-30T16:00:05Z
2025-04-30T16:00:05Z
2025-04-30T16:00:05Z
2025-04-30T16:00:05Z
2025-04-30T16:00:05Z
2025-04-30T16:00:05Z
2025-04-30T16:00:06Z
2025-04-30T16:00:06Z
2025-04-30T16:00:06Z
2025-04-30T16:00:07Z
2025-04-30T16:00:07Z
2025-04-30T16:00:07Z
2025-04-30T16:00:08Z

┌──(kali㉿kali)-[~/Desktop/sun]
└─$ grep -E '^2025-04-30T1[6-7]:' siem-logs.csv > labour00.csv

┌──(kali㉿kali)-[~/Desktop/sun]
└─$ wc -l labour00.csv

9999 labour00.csv
```

```
cut -d',' -f1 siem-logs.csv | head -20
grep -E '^2025-04-30T1[6-7]:' siem-logs.csv > labour00.csv
wc -l labour00.csv
```

- `cut -d',' -f1` → Extracts only the first column (timestamps).
- `head -20` → Displays the first 20 timestamps.
- `grep -E '^2025-04-30T1[6-7]:'` → Filters logs between **16:00 and 17:59** on `2025-04-30`.
- `> labour00.csv` → Saves the filtered logs into `labour00.csv`.
- `wc -l` → Counts lines in the filtered dataset.

Result: We extracted **9999 logs** from the relevant time range.

Step 4: Identify the Targeted Endpoint



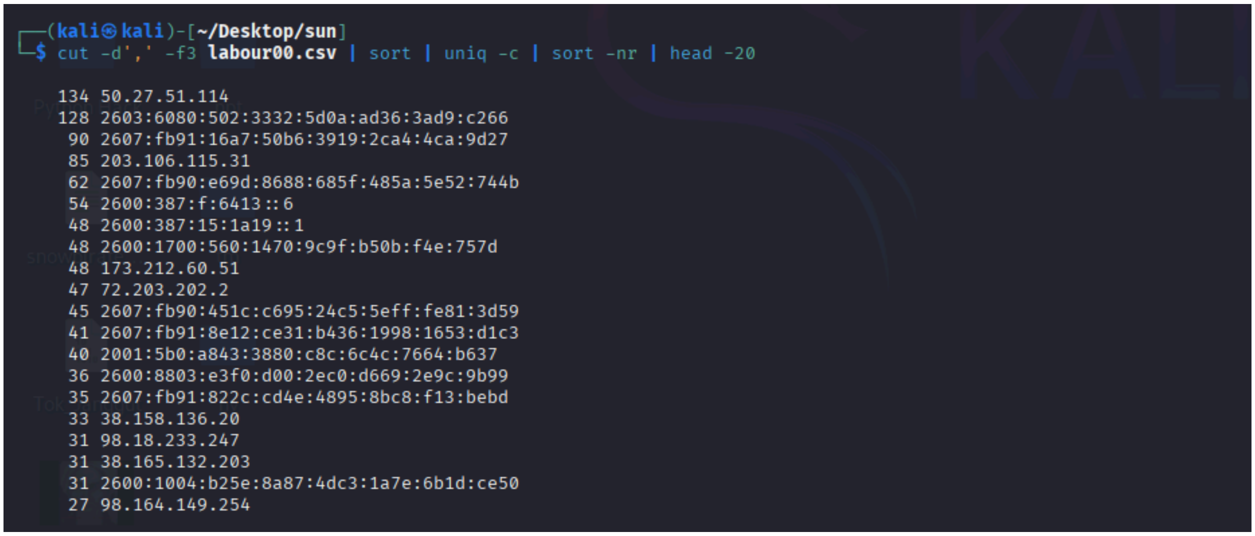
```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ cut -d',' -f8 labour00.csv | sort | uniq -c | sort -nr | head

   9999 /signin
```

```
cut -d',' -f3 labour00.csv | sort | uniq -c | sort -nr | head -20
```

- `cut -d',' -f8` → Extracts the **URL Path** column.
- `sort` → Sorts entries alphabetically.
- `uniq -c` → Counts unique occurrences.
- `sort -nr` → Sorts results numerically in descending order.
- `head` → Shows the top results.

Result: All logs (9999) target `/signin`. This suggests a brute force or credential-stuffing attack on the login page.

Step 5: Analyze Attacker IPs



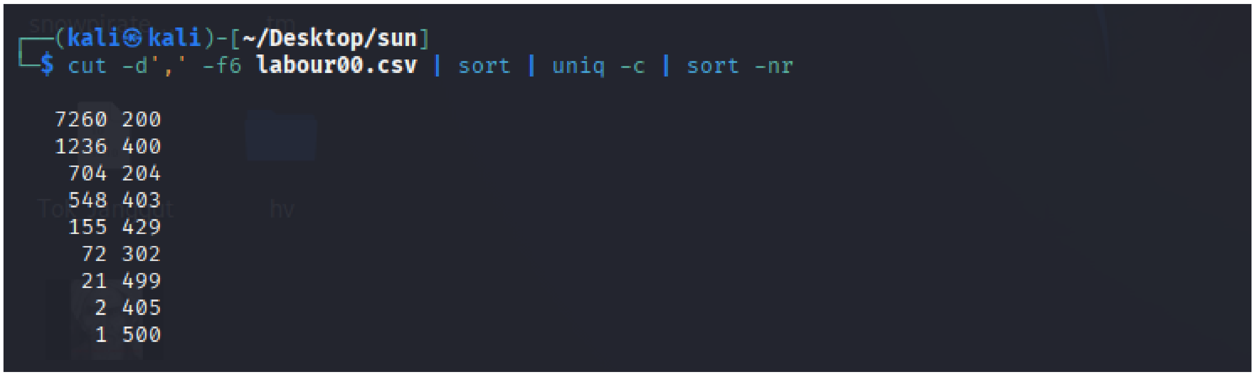
```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ cut -d',' -f3 labour00.csv | sort | uniq -c | sort -nr | head -20

    134 50.27.51.114
    128 2603:6080:502:3332:5d0a:ad36:3ad9:c266
     90 2607:fb91:16a7:50b6:3919:2ca4:4ca:9d27
     85 203.106.115.31
     62 2607:fb90:e69d:8688:685f:485a:5e52:744b
     54 2600:387:f:6413::6
     48 2600:387:15:1a19::1
     48 2600:1700:560:1470:9c9f:b50b:f4e:757d
     48 173.212.60.51
     47 72.203.202.2
     45 2607:fb90:451c:c695:24c5:5eff:fe81:3d59
     41 2607:fb91:8e12:ce31:b436:1998:1653:d1c3
     40 2001:5b0:a843:3880:c8c:6c4c:7664:b637
     36 2600:8803:e3f0:d00:2ec0:d669:2e9c:9b99
     35 2607:fb91:822c:cd4e:4895:8bc8:f13:bebd
     33 38.158.136.20
     31 98.18.233.247
     31 38.165.132.203
     31 2600:1004:b25e:8a87:4dc3:1a7e:6b1d:ce50
     27 98.164.149.254
```

```
cut -d',' -f3 labour00.csv | sort | uniq -c | sort -nr | head -20
```

- `cut -d',' -f3` → Extracts the **Client IP** field.
- `sort | uniq -c` → Groups and counts IPs.
- `sort -nr` → Displays most frequent IPs first.

Result: We observe repeated activity from certain IPv4/IPv6 addresses such as `50.27.51.114` and `203.106.115.31`. These are likely attacker IPs.

Step 6: Review Status Codes



```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ cut -d',' -f6 labour00.csv | sort | uniq -c | sort -nr

   7260 200
   1236 400
    704 204
    548 403
    155 429
     72 302
     21 499
      2 405
      1 500
```

```
cut -d',' -f6 labour00.csv | sort | uniq -c | sort -nr
```

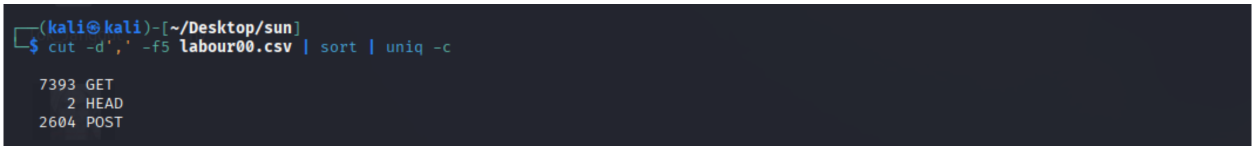
- `cut -d',' -f6` → Extracts **HTTP Status Codes**.
- `sort | uniq -c` → Groups and counts each unique code.
- `sort -nr` → Displays most frequent codes first.

Result:

- **200 (OK)** – Successful requests.
- **204 (No Content)** – Possibly valid login with no response body.
- **400, 403, 429, 499, 500** – Errors and blocked attempts (common in brute force attacks).

This confirms suspicious login attempts.

Step 7: Check HTTP Methods



```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ cut -d',' -f5 labour00.csv | sort | uniq -c

   7393 GET
      2 HEAD
   2604 POST
```

```
cut -d',' -f5 labour00.csv | sort | uniq -c
```

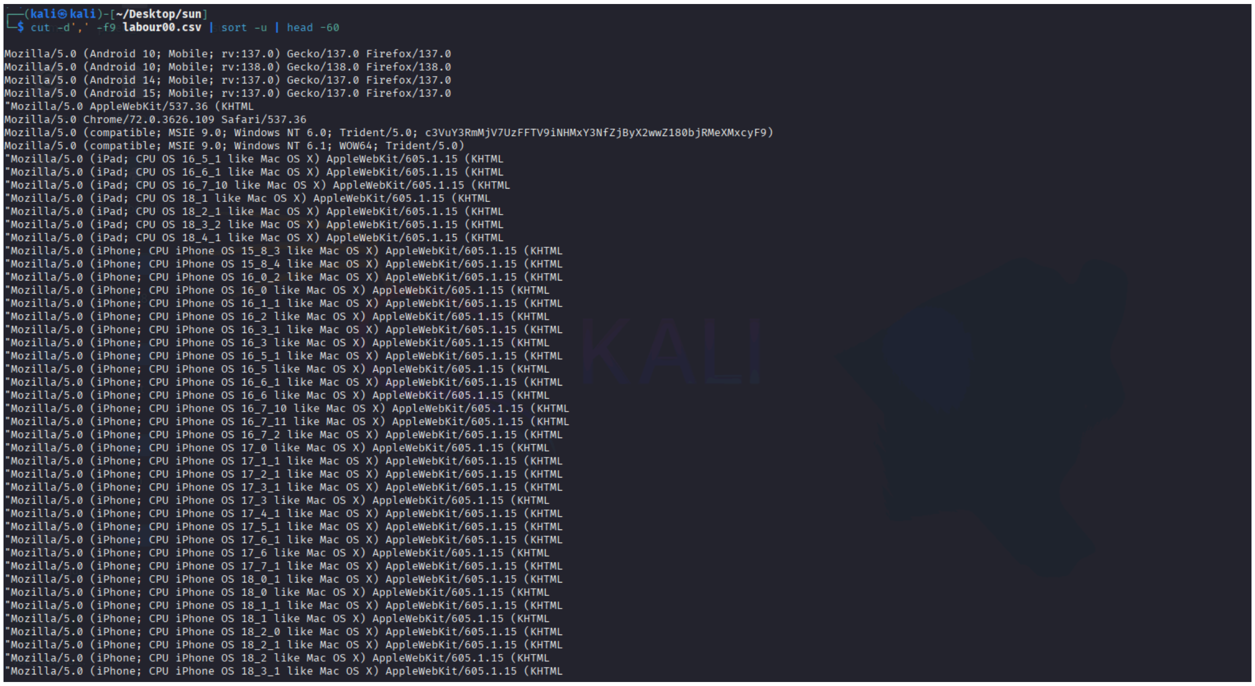
- `cut -d',' -f5` → Extracts **HTTP Method.**
- `sort | uniq -c` → Groups and counts.

Result:

- **7393 GET**
- **2604 POST**
- **2 HEAD**

The use of **POST** aligns with login attempts, since credentials are usually submitted via POST.

Step 8: Look for Anomalous User Agents



```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ cut -d',' -f9 labour00.csv | sort -u | head -60

Mozilla/5.0 (Android 10; Mobile; rv:137.0) Gecko/137.0 Firefox/137.0
Mozilla/5.0 (Android 10; Mobile; rv:138.0) Gecko/138.0 Firefox/138.0
Mozilla/5.0 (Android 14; Mobile; rv:137.0) Gecko/137.0 Firefox/137.0
Mozilla/5.0 (Android 15; Mobile; rv:137.0) Gecko/137.0 Firefox/137.0
"Mozilla/5.0 AppleWebKit/537.36 (KHTML
Mozilla/5.0 Chrome/72.0.3626.109 Safari/537.36
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0; c3VuY3RmMjV7UzFFTV9iNHMxY3NfZjByX2wwZ180bjRMeXMxcyF9)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
"Mozilla/5.0 (iPad; CPU OS 16_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPad; CPU OS 16_6_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPad; CPU OS 16_7_10 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPad; CPU OS 18_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPad; CPU OS 18_2_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPad; CPU OS 18_3_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPad; CPU OS 18_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 15_8_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 15_8_4 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_0_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_1_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_5 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_6_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_7_10 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_7_11 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_7_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_1_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_2_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_6_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 17_7_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_0_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_1_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_2_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_2_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML
"Mozilla/5.0 (iPhone; CPU iPhone OS 18_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML
cut -d',' -f9 labour00.csv | sort -u | head -60
```

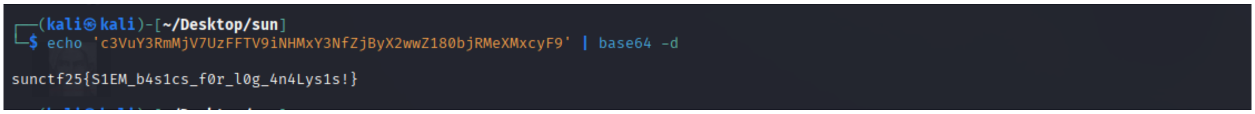
- `cut -d',' -f9` → Extracts the **User-Agent** field.
- `sort -u` → Sorts and removes duplicates.
- `head -60` → Shows the first 60 unique entries.

Result: Among normal browsers (Chrome, Safari, Firefox), one suspicious entry stands out:

```
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0; c3VuY3RmMjV7UzFFTV9iNHMxY3NfZjByX2wwZ180bjRMeXMxcyF9)
```

The random-looking string inside is **Base64-encoded**.

Step 9: Decode the Suspicious String



```
┌──(kali㉿kali)-[~/Desktop/sun]
└─$ echo 'c3VuY3RmMjV7UzFFTV9iNHMxY3NfZjByX2wwZ180bjRMeXMxcyF9' | base64 -d

sunctf25{S1EM_b4s1cs_f0r_l0g_4n4Lys1s!}
```

- `echo` → Prints the string.
- `base64 -d` → Decodes from Base64.

```
sunctf25{S1EM_b4s1cs_f0r_l0g_4n4Lys1s!}
```