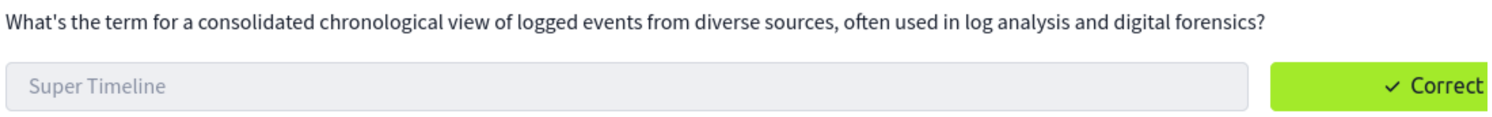
Intro to Log Analysis

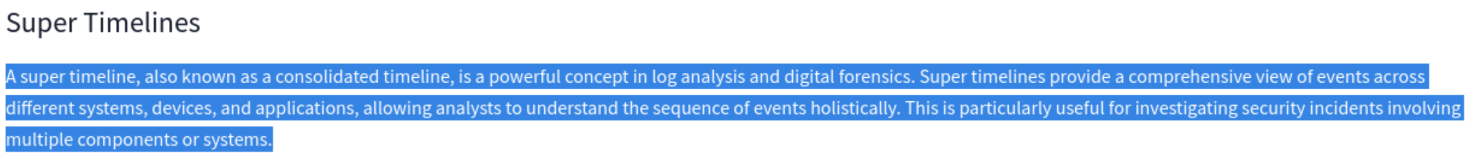
INTRODUCTION
Log analysis is an essential aspect of cyber security and system monitoring. log analysis examines and interprets log event data generated by various sources (devices, applications, and systems) to monitor metrics and identify security incidents. It involves collecting, parsing, and processing log files to turn data into actionable objectives. By adopting an effective log analysis strategy, security teams can more accurately respond to security incidents and gain proactive insights into potential threats.
This report entails my learning outcome and approaches I made to tackle each task in this room.
The first two tasks were without questions, but I focused on reading them thoroughly since they lay the foundations for log analysis.


What's the term for a consolidated chronological view of logged events from diverse sources, often used in log analysis and digital forensics?

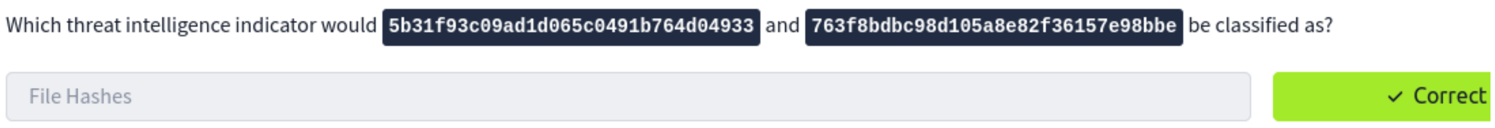
Super Timeline

✓ Correct

Read the explanation of the different timelines listed in this task. I was looking for a name for the timeline that displays and correlates the data from different systems that help analysts track an incident across the whole organisation.
From the image below, super timelines it the term.


Super Timelines

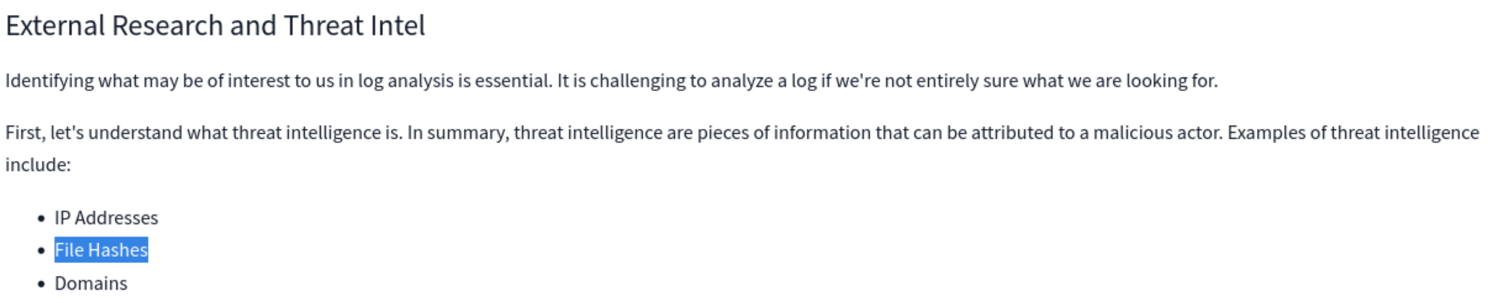
A super timeline, also known as a consolidated timeline, is a powerful concept in log analysis and digital forensics. Super timelines provide a comprehensive view of events across different systems, devices, and applications, allowing analysts to understand the sequence of events holistically. This is particularly useful for investigating security incidents involving multiple components or systems.


Which threat intelligence indicator would `5b31f93c09ad1d065c0491b764d04933` and `763f8bdbc98d105a8e82f36157e98bbe` be classified as?

File Hashes

✓ Correct

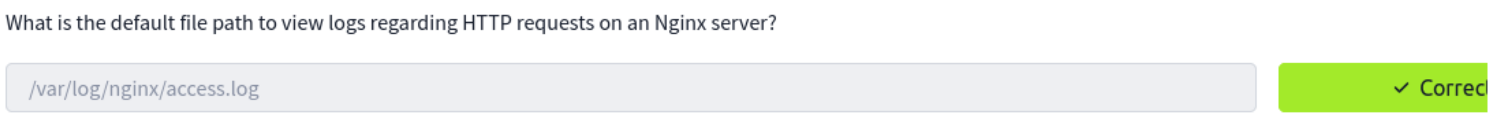
They look similar. Looking closely, they both consist of 32 hexadecimal characters, which is characteristic of an output from the MD5 algorithm. Looking under the "Threat Intel" paragraph, I found the answer to the question as seen below.


External Research and Threat Intel

Identifying what may be of interest to us in log analysis is essential. It is challenging to analyze a log if we're not entirely sure what we are looking for.

First, let's understand what threat intelligence is. In summary, threat intelligence are pieces of information that can be attributed to a malicious actor. Examples of threat intelligence include:

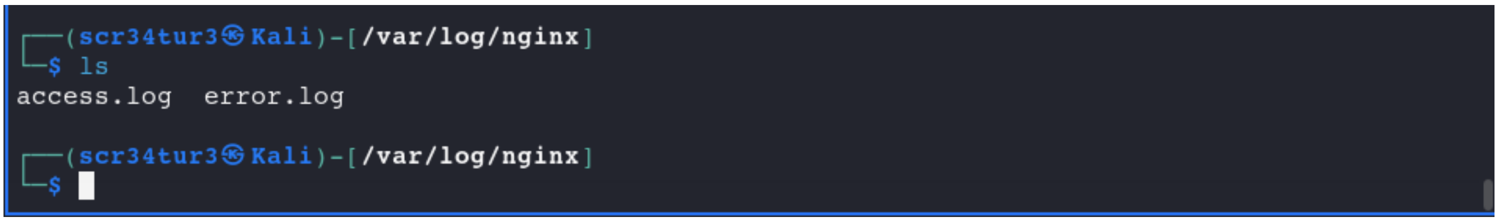
- IP Addresses
- File Hashes
- Domains


What is the default file path to view logs regarding HTTP requests on an Nginx server?

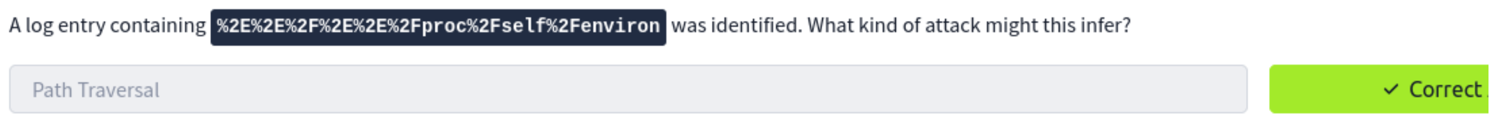
/var/log/nginx/access.log

✓ Correc

Personally I checked from my local machine as seen from the image below.



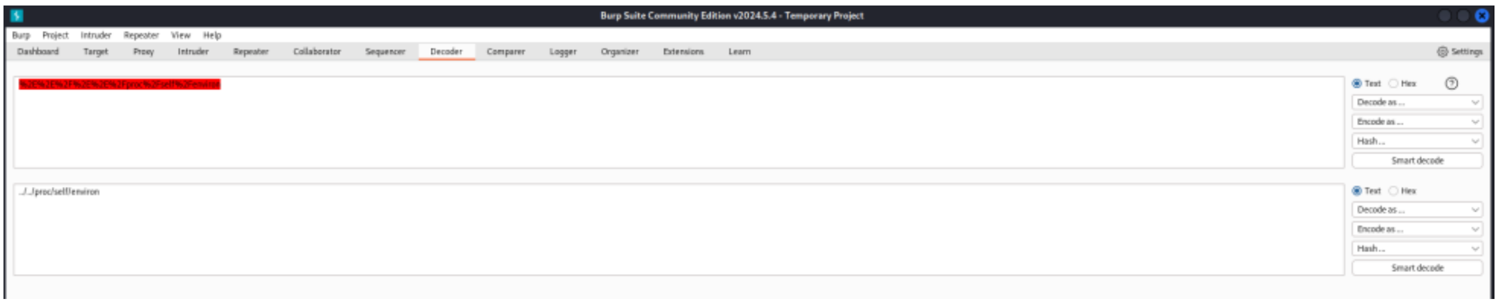
```
┌──(scr34tur3㉿Kali)-[/var/log/nginx]
└─$ ls
access.log  error.log

┌──(scr34tur3㉿Kali)-[/var/log/nginx]
└─$ 
```



A log entry containing `%2E%2E%2F%2E%2E%2Fproc%2Fself%2Fenviron` was identified. What kind of attack might this infer?

Path Traversal

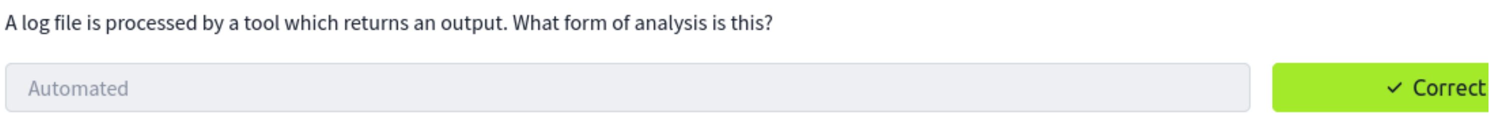
This string contains multiple %2E and %2F characters, which indicates a URL encoding. Decoding the string yields "**../../proc/self/environ**". the **'../../'** portion of the string indicates an attempt to back out of the original directory and "**proc/self/environ"** indicates an attempt to access the environmental variables of a process. I tried to url decode it as seen from the image below.



Burp Suite Community Edition v2024.5.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Text Hex
Decode as ...
Encode as ...
Hash ...
Smart decode

```
%2E%2E%2F%2E%2E%2Fproc%2Fself%2Fenviron
```

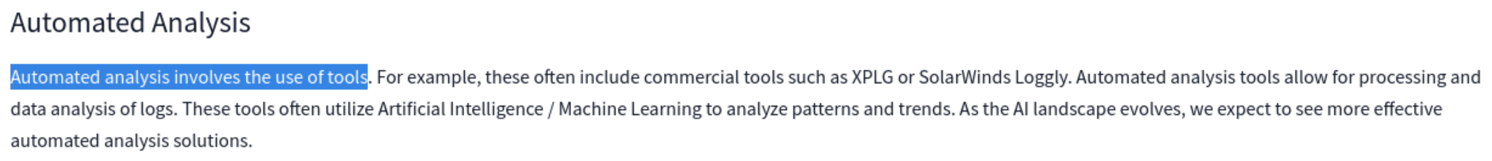
```
../../proc/self/environ
```



A log file is processed by a tool which returns an output. What form of analysis is this?

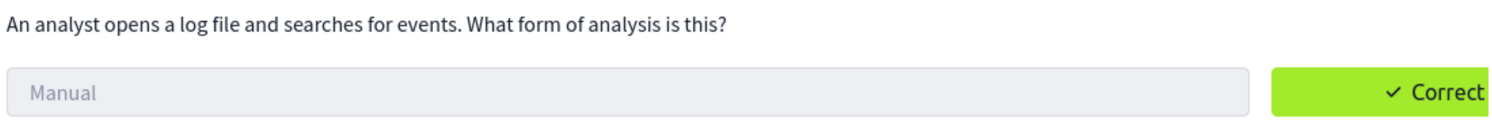
Automated

Reading and understand the text as seen below, I was able to answer the task above.



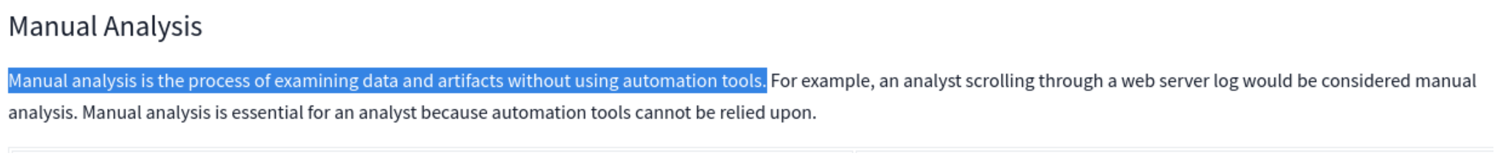
Automated Analysis

Automated analysis involves the use of tools. For example, these often include commercial tools such as XPLG or SolarWinds Loggly. Automated analysis tools allow for processing and data analysis of logs. These tools often utilize Artificial Intelligence / Machine Learning to analyze patterns and trends. As the AI landscape evolves, we expect to see more effective automated analysis solutions.



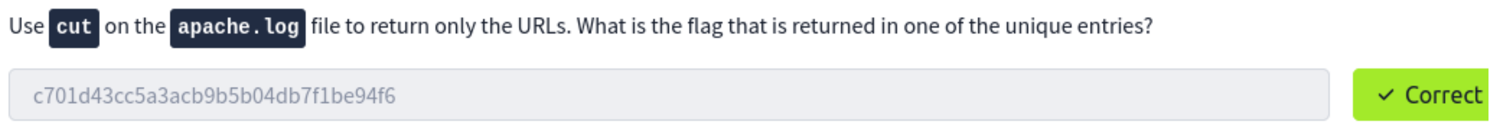
An analyst opens a log file and searches for events. What form of analysis is this?

Manual



Manual Analysis

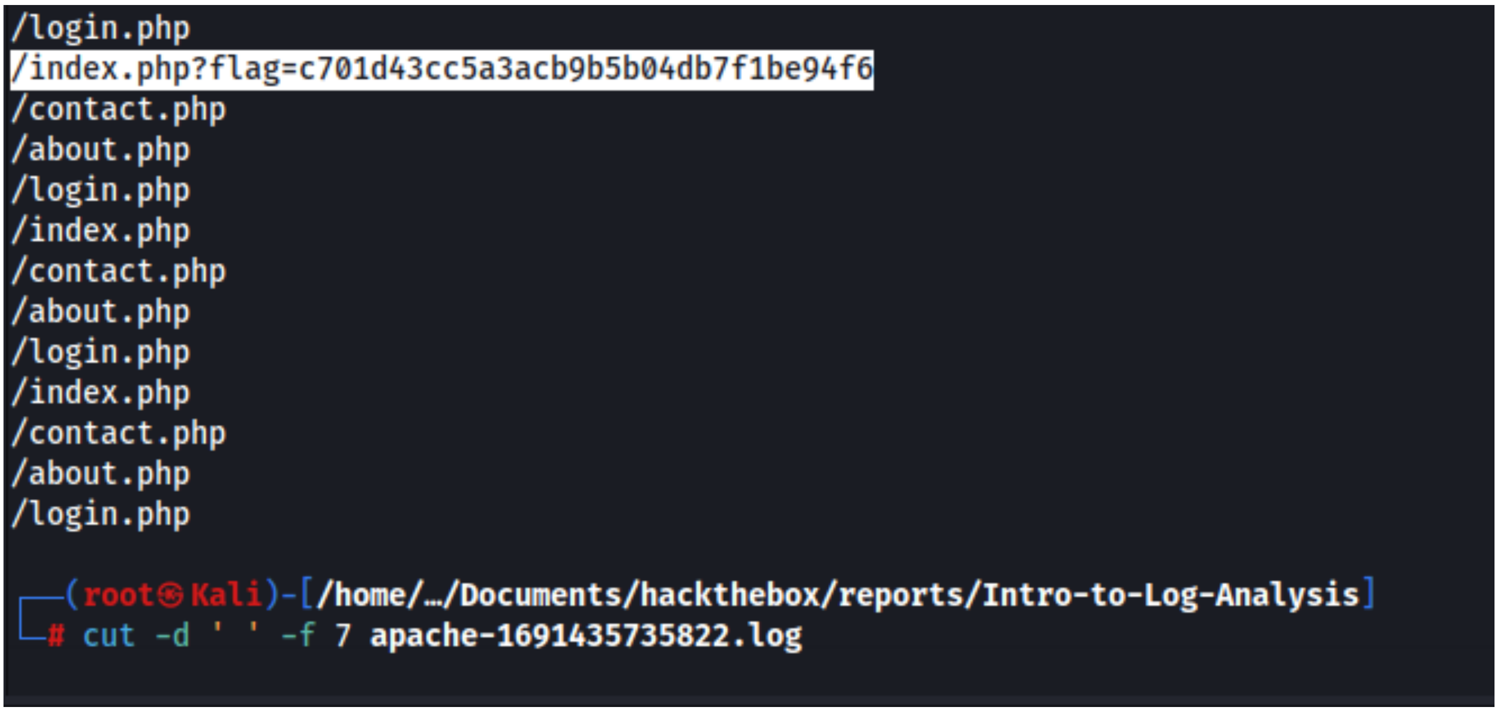
Manual analysis is the process of examining data and artifacts without using automation tools. For example, an analyst scrolling through a web server log would be considered manual analysis. Manual analysis is essential for an analyst because automation tools cannot be relied upon.



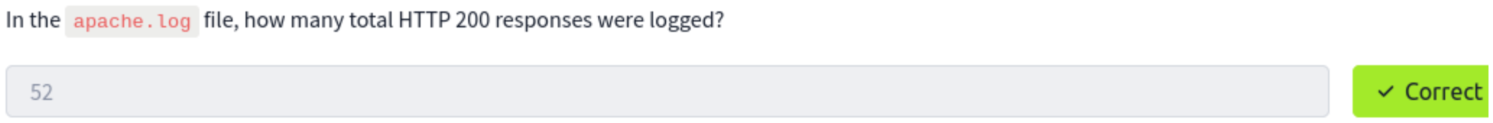
Use `cut` on the `apache.log` file to return only the URLs. What is the flag that is returned in one of the unique entries?

c701d43cc5a3acb9b5b04db7f1be94f6

Using the cut cmd alongside -f flag to specify the field, I was able to retreive the flag as shown below.



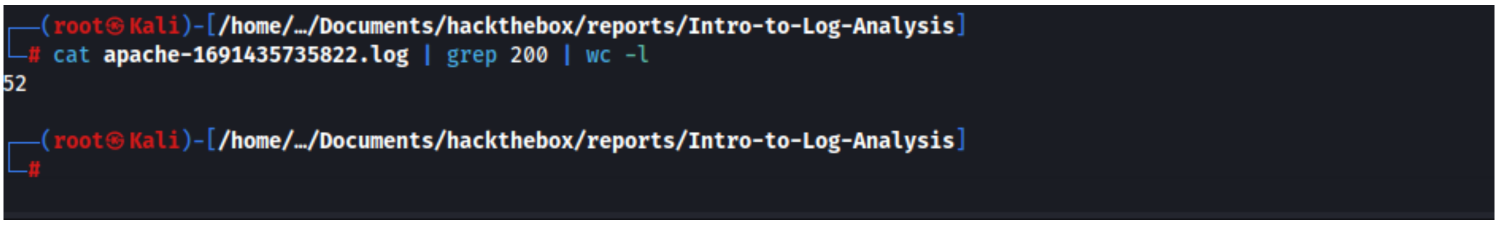
```
/login.php
/index.php?flag=c701d43cc5a3acb9b5b04db7f1be94f6
/contact.php
/about.php
/login.php
/index.php
/contact.php
/about.php
/login.php
/index.php
/contact.php
/about.php
/login.php

┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# cut -d ' ' -f 7 apache-1691435735822.log
```



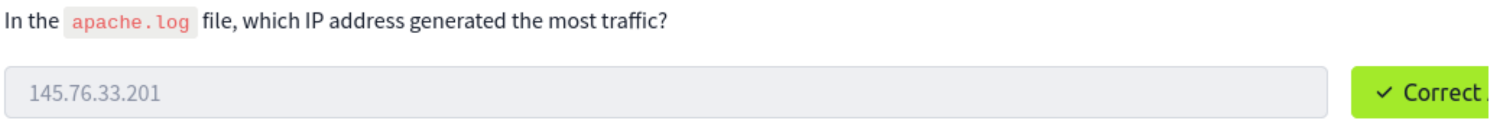
In the `apache.log` file, how many total HTTP 200 responses were logged?

52

When finding the total value, the wc cmd comes to be a handy tool. As seen from the image below, I used the grep cmd to grep for the specified status code; 200 and piped the output to wc cmd to count the number of requests with the 200 status code.



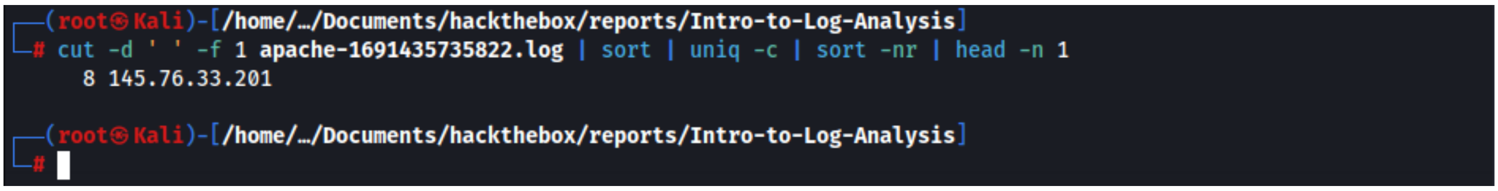
```
┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# cat apache-1691435735822.log | grep 200 | wc -l
52

┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─#
```



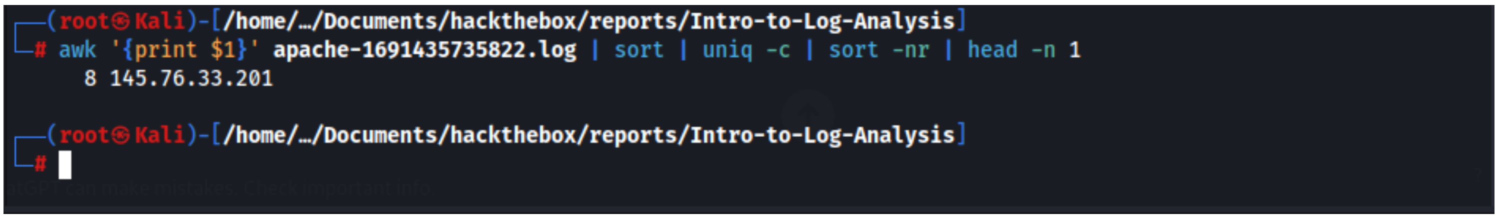
In the `apache.log` file, which IP address generated the most traffic?

145.76.33.201

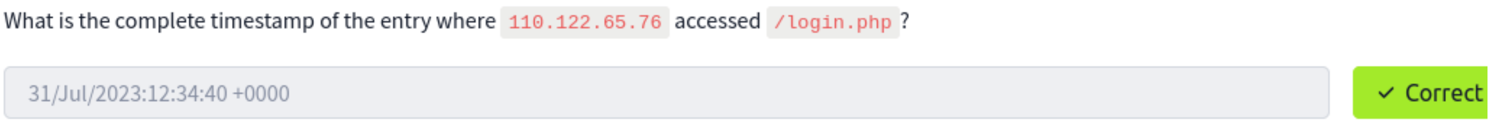
For this case, I could either use cut cmd or awk to specify the field of interest from the apache.log file. From the image below I was able to retrieve the IP address most generated.



```
┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# cut -d ' ' -f 1 apache-1691435735822.log | sort | uniq -c | sort -nr | head -n 1
      8 145.76.33.201

┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# 
```



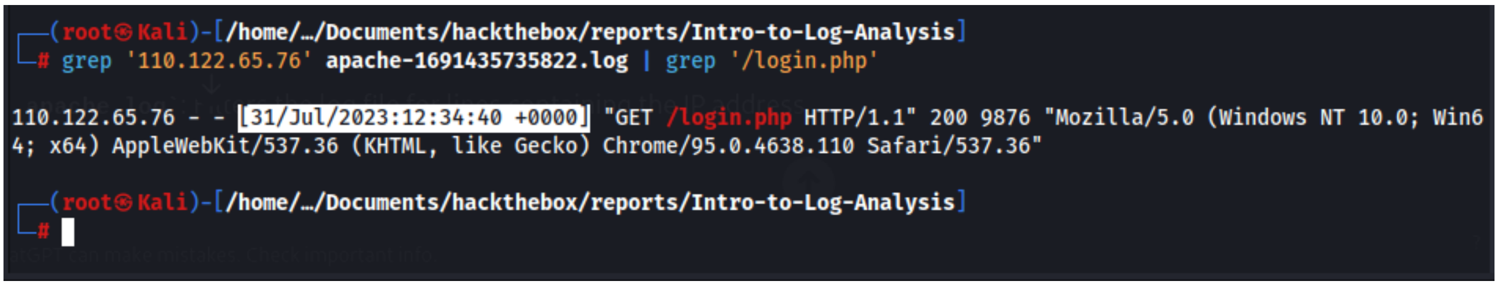
```
┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# awk '{print $1}' apache-1691435735822.log | sort | uniq -c | sort -nr | head -n 1
      8 145.76.33.201

┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# 
```



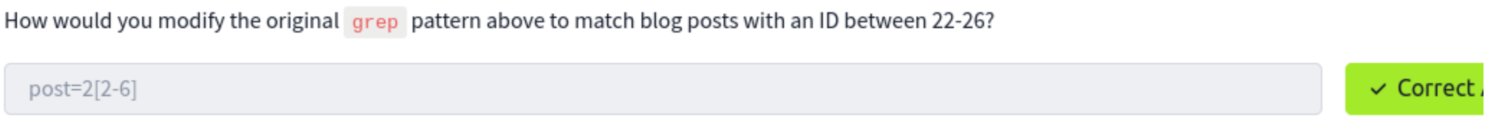
What is the complete timestamp of the entry where `110.122.65.76` accessed `/login.php` ?

31/Jul/2023:12:34:40 +0000

The grep cmd helped out to solve this task as all was needed from me was to grab the ip and path url as shown below.



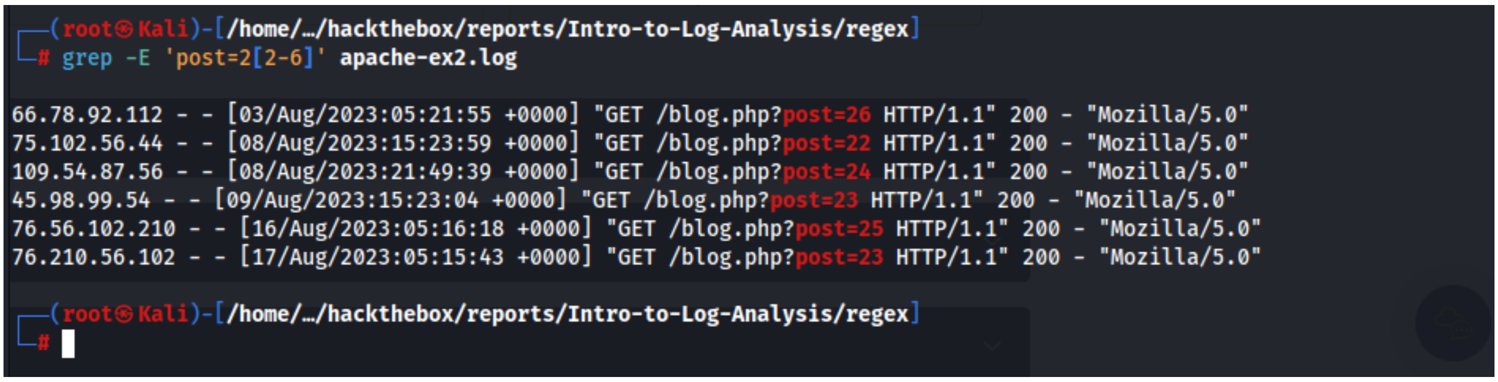
```
┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# grep '110.122.65.76' apache-1691435735822.log | grep '/login.php'

110.122.65.76 - - [31/Jul/2023:12:34:40 +0000] "GET /login.php HTTP/1.1" 200 9876 "Mozilla/5.0 (Windows NT 10.0; Win6
4; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.110 Safari/537.36"

┌──(root㉿Kali)-[/home/…/Documents/hackthebox/reports/Intro-to-Log-Analysis]
└─# 
```



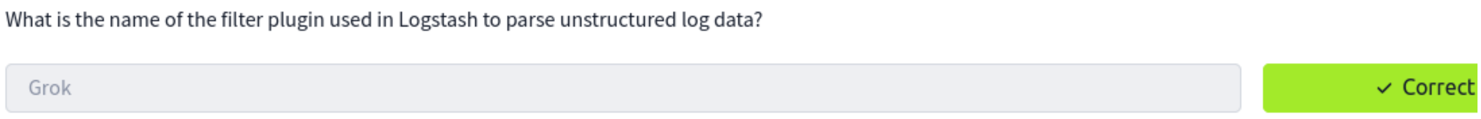
How would you modify the original `grep` pattern above to match blog posts with an ID between 22-26?

post=2[2-6]

From the image below, I modified the grep cmd with -E flag to capture the requests whose values at the post parameter would range between btwn 22-26.



```
┌──(root㉿Kali)-[/home/…/hackthebox/reports/Intro-to-Log-Analysis/regex]
└─# grep -E 'post=2[2-6]' apache-ex2.log

66.78.92.112 - - [03/Aug/2023:05:21:55 +0000] "GET /blog.php?post=26 HTTP/1.1" 200 - "Mozilla/5.0"
75.102.56.44 - - [08/Aug/2023:15:23:59 +0000] "GET /blog.php?post=22 HTTP/1.1" 200 - "Mozilla/5.0"
109.54.87.56 - - [08/Aug/2023:21:49:39 +0000] "GET /blog.php?post=24 HTTP/1.1" 200 - "Mozilla/5.0"
45.98.99.54 - - [09/Aug/2023:15:23:04 +0000] "GET /blog.php?post=23 HTTP/1.1" 200 - "Mozilla/5.0"
76.56.102.210 - - [16/Aug/2023:05:16:18 +0000] "GET /blog.php?post=25 HTTP/1.1" 200 - "Mozilla/5.0"
76.210.56.102 - - [17/Aug/2023:05:15:43 +0000] "GET /blog.php?post=23 HTTP/1.1" 200 - "Mozilla/5.0"

┌──(root㉿Kali)-[/home/…/hackthebox/reports/Intro-to-Log-Analysis/regex]
└─# 
```



What is the name of the filter plugin used in Logstash to parse unstructured log data?

Grok

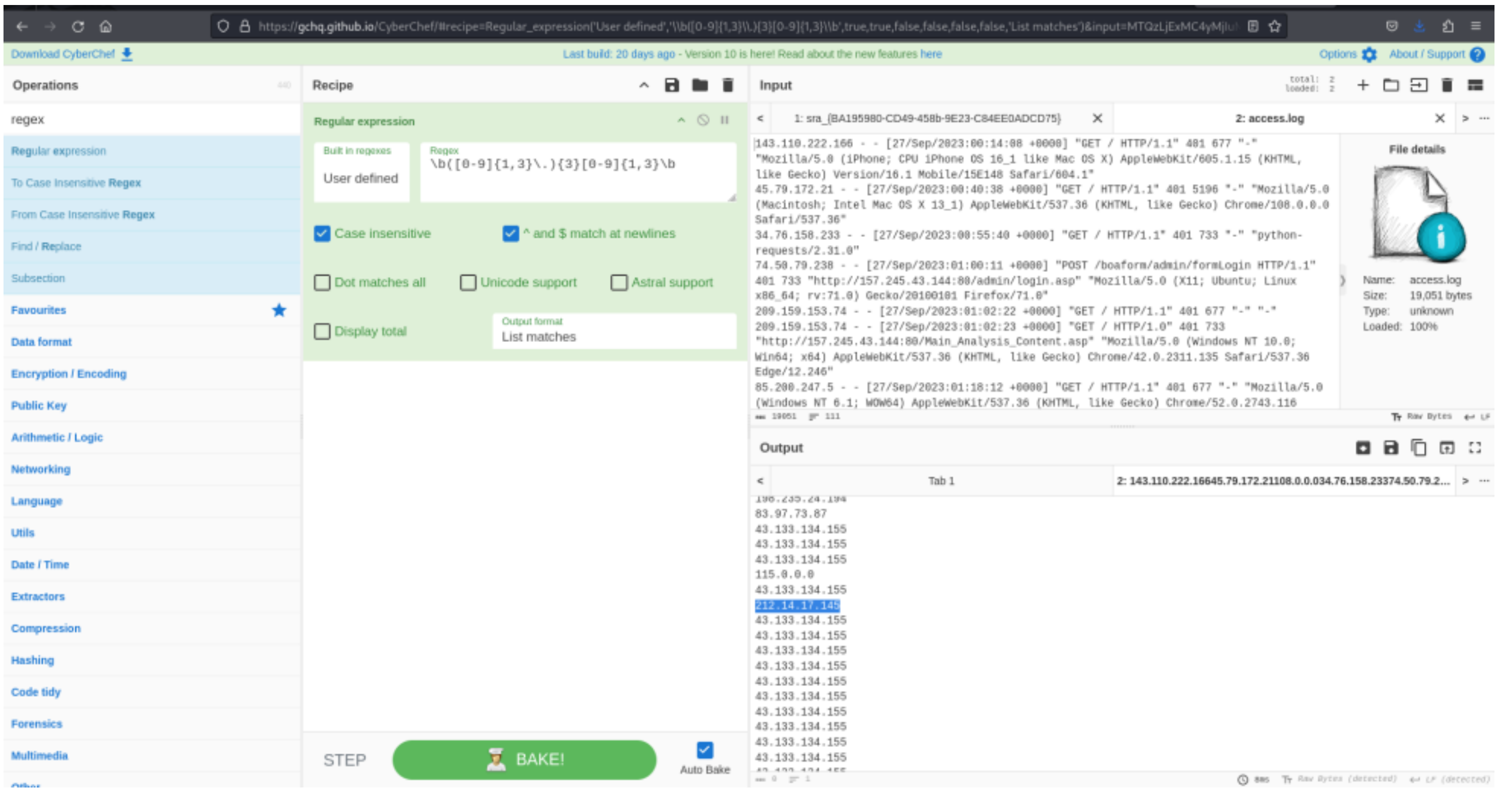
Reading through the notes in the room, I came along the text below.

Grok is a powerful Logstash plugin that enables you to parse unstructured log data into something structured and searchable. It's commonly used for any log format written for humans to read rather than for computer consumption. It works by combining text patterns with the `%{SYNTAX:SEMANTIC}` pattern syntax. However, sometimes, Logstash lacks the built-in pattern we need. In these cases, we can define custom patterns using the **Oniguruma syntax** and take advantage of regular expressions. More info on Grok and its use within the Elastic Stack can be found in the Elastic documentation here.

Upload the log file named "access.log" to CyberChef. Use regex to list all of the IP addresses. What is the full IP address beginning in 212?

212.14.17.145 ✓ Correct

First, I uploaded the file into CyberChef by following the instructions in the task. I Selected regular expression from the recipes and chose IPv4 addresses from the list of built-in regexes. Finally, I selected the option to list all matches as shown below.



```
https://gchq.github.io/CyberChef/#recipe=Regular_expression('User defined','\\b([0-9]{1,3}\\.){3}[0-9]{1,3}\\b',true,true,false,false,false,false,'List matches')&input=MTQzLjExMC4yMjlu
```

Regular expression
Built in regexes: User defined
Regex: `\b([0-9]{1,3}\.){3}[0-9]{1,3}\b`
[✓] Case insensitive [✓] ^ and $ match at newlines
[ ] Dot matches all [ ] Unicode support [ ] Astral support
[ ] Display total
Output format: List matches

Input (total: 2, loaded: 2)

```
143.110.222.166 - - [27/Sep/2023:00:14:08 +0000] "GET / HTTP/1.1" 401 677 "-"
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML,
like Gecko) Version/16.1 Mobile/15E148 Safari/604.1"
45.79.172.21 - - [27/Sep/2023:00:40:38 +0000] "GET / HTTP/1.1" 401 5196 "-" "Mozilla/5.0
(Macintosh; Intel Mac OS X 13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0
Safari/537.36"
34.76.158.233 - - [27/Sep/2023:00:55:40 +0000] "GET / HTTP/1.1" 401 733 "-" "python-
requests/2.31.0"
74.50.79.238 - - [27/Sep/2023:01:00:11 +0000] "POST /boaform/admin/formLogin HTTP/1.1"
401 733 "http://157.245.43.144:80/admin/login.asp" "Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:71.0) Gecko/20100101 Firefox/71.0"
209.159.153.74 - - [27/Sep/2023:01:02:22 +0000] "GET / HTTP/1.1" 401 677 "-" "-"
209.159.153.74 - - [27/Sep/2023:01:02:23 +0000] "GET / HTTP/1.0" 401 733
"http://157.245.43.144:80/Main_Analysis_Content.asp" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36
Edge/12.246"
85.200.247.5 - - [27/Sep/2023:01:18:12 +0000] "GET / HTTP/1.1" 401 677 "-" "Mozilla/5.0
(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116
```

Name: access.log
Size: 19,051 bytes
Type: unknown
Loaded: 100%

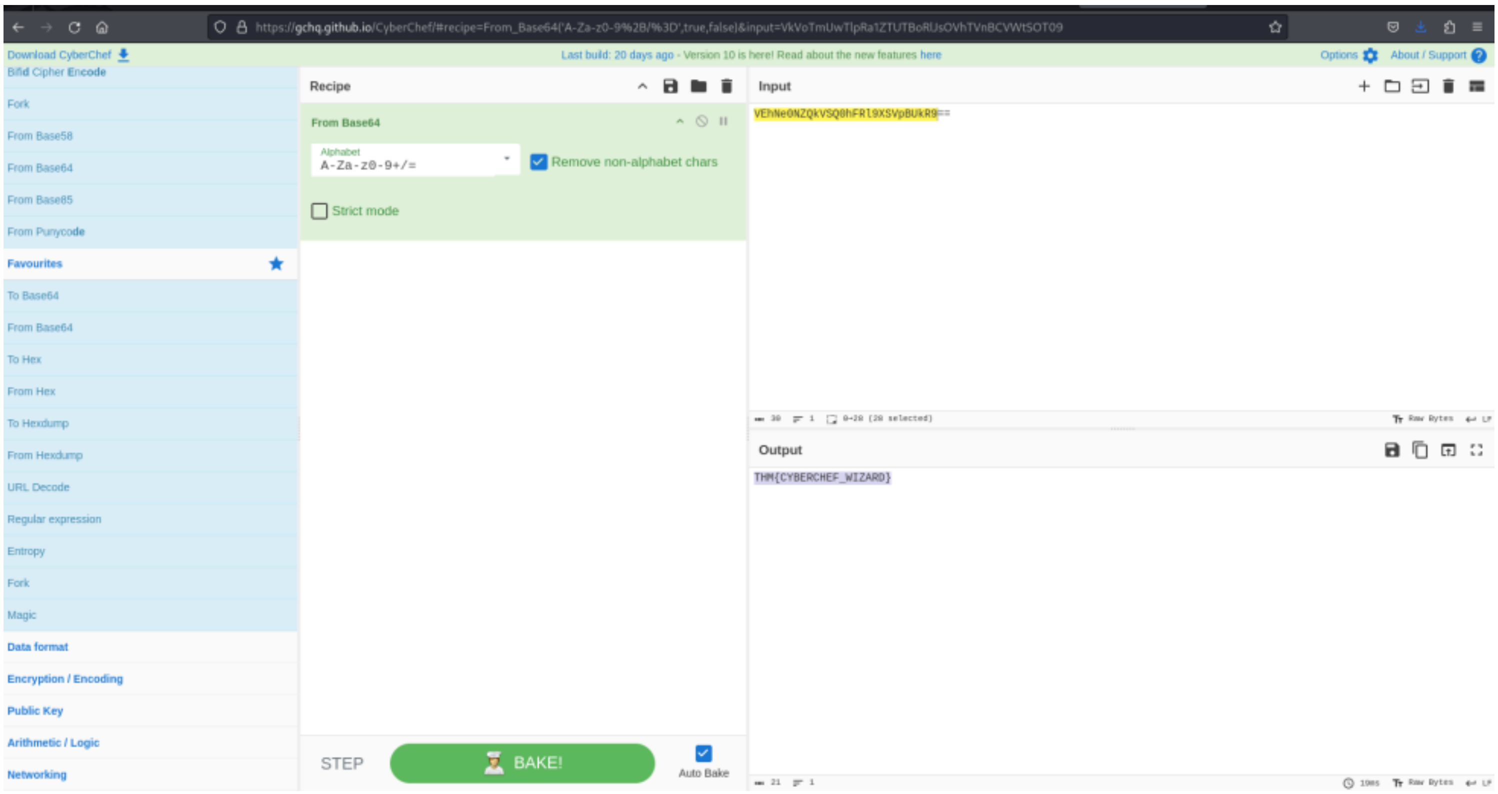
2: 143.110.222.16645.79.172.21108.0.0.034.76.158.23374.50.79.2...

```
83.97.73.87
43.133.134.155
43.133.134.155
43.133.134.155
115.0.0.0
43.133.134.155
212.14.17.145
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
43.133.134.155
```

Using the same log file from Question #2, a request was made that is encoded in base64. What is the decoded value?

THM{CYBERCHEF_WIZARD} ✓ Correct

I had cat the content of this file from my terminal, and having the knowledge of how a base64 looks like, I was able to locate it. I then uploaded it to the cyberchef and there I retrieved the flag as seen below.



```
https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,false)&input=VkVoTmUwTlpRa1ZTUTBoRlJsOVhTVnBCVWtSOT09
```

Recipe: From Base64 — Alphabet: A-Za-z0-9+/= — ✓ Remove non-alphabet chars — Strict mode: ☐

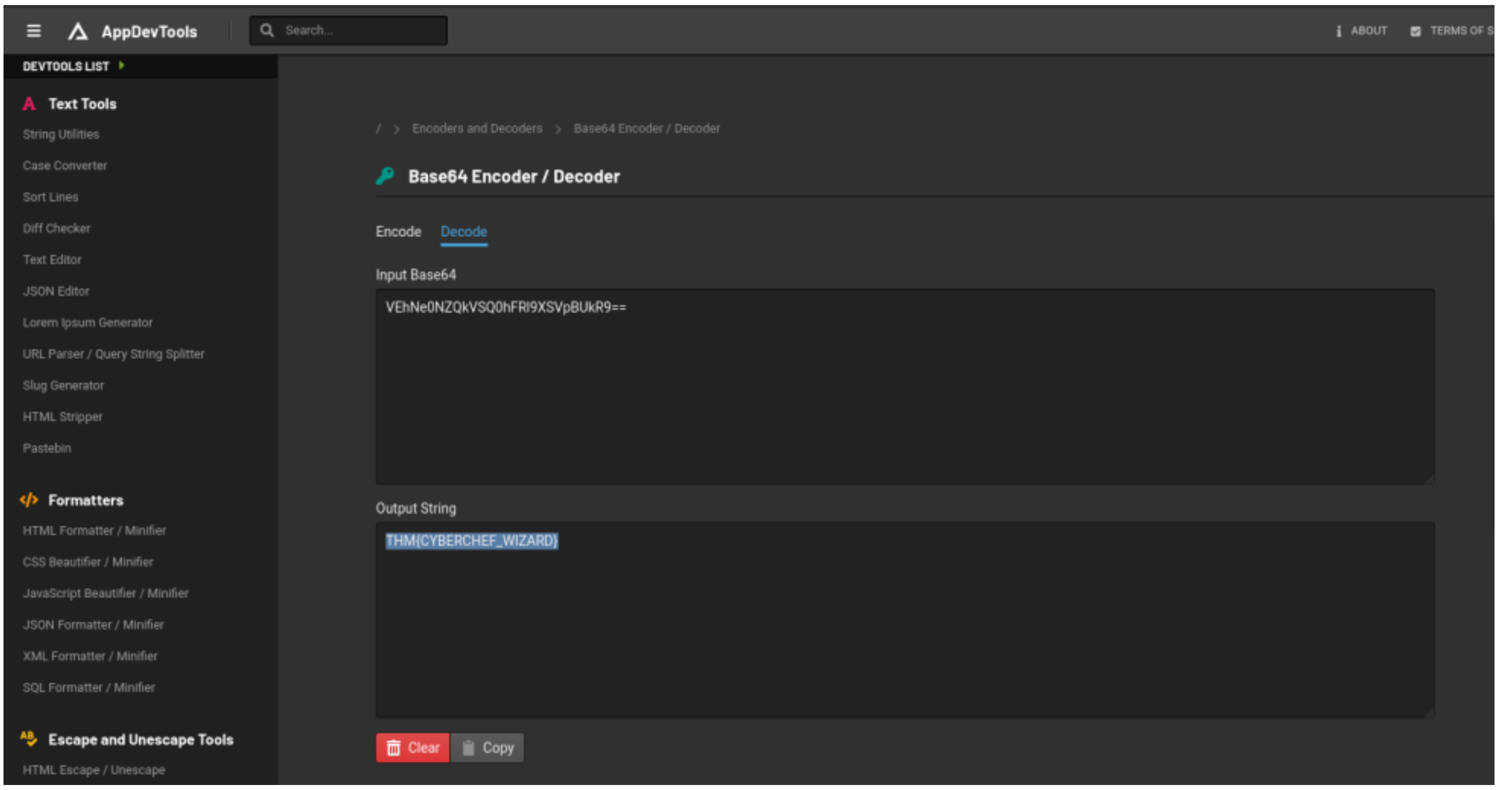
Input:

```
VEhNe0NZQkVSQ0hFRl9XSVpBUkR9==
```

Output:

```
THM{CYBERCHEF_WIZARD}
```

Though taking this base64 text to the site shown below, I was able to decode the base64 text.



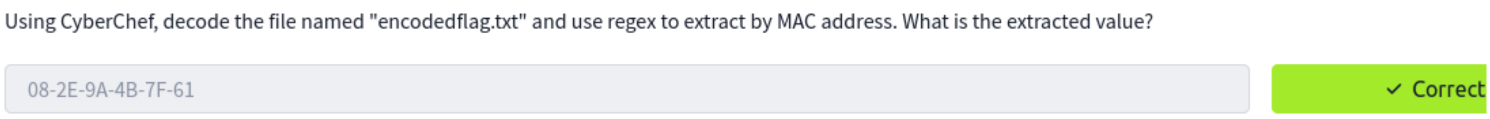
Base64 Encoder / Decoder

Input Base64

```
VEhNe0NZQkVSQ0hFRl9XSVpBUkR9==
```

Output String

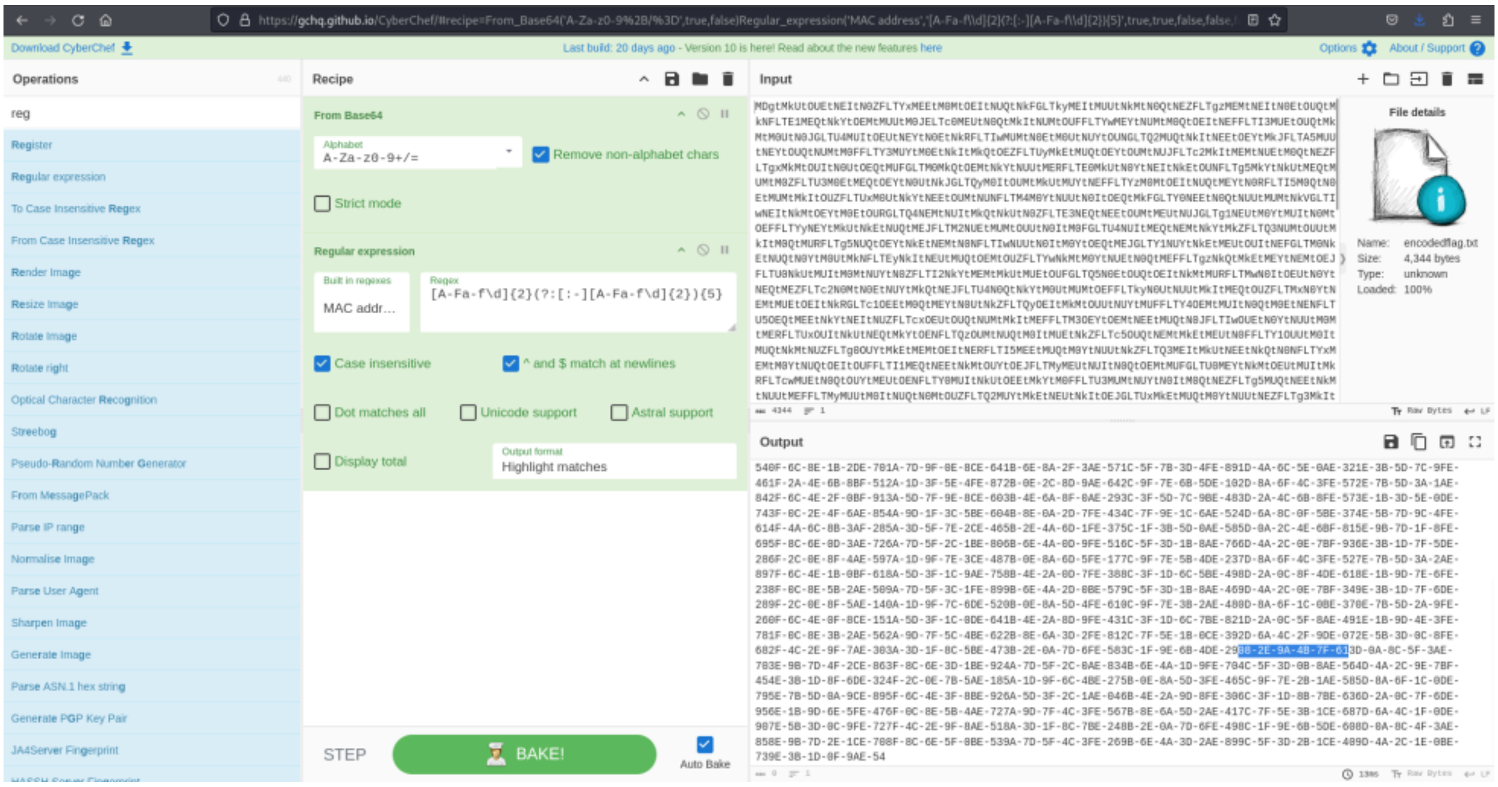
```
THM{CYBERCHEF_WIZARD}
```



Using CyberChef, decode the file named "encodedflag.txt" and use regex to extract by MAC address. What is the extracted value?

08-2E-9A-4B-7F-61 ✓ Correct

I loaded the file "encodedflag.txt" into our skilled Chef and ask him to decode it from Base64. In the output, I have what is seemingly a list of MAC addresses. I Just add another operation that uses regular expressions to find a MAC address (it is one of the pre-defined expressions) since looking for a valid MAC address here would be like looking for a needle in a haystack.


https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,false)Regular_expression('MAC address','[A-Fa-f\\d]{2}(?:[:-][A-Fa-f\\d]{2}){5}',true,true,false,false,

Recipe

From Base64
Alphabet: A-Za-z0-9+/=
☑ Remove non-alphabet chars
☐ Strict mode

Regular expression
Built in regexes: MAC addr...
Regex: [A-Fa-f\d]{2}(?:[:-][A-Fa-f\d]{2}){5}
☑ Case insensitive
☑ ^ and $ match at newlines
☐ Dot matches all
☐ Unicode support
☐ Astral support
☐ Display total
Output format: Highlight matches

File details
Name: encodedflag.txt
Size: 4,344 bytes
Type: unknown
Loaded: 100%

Input

```
MDgtMkUtOUEtNEItN0ZFLTYxMEEtM0MtOEItNUQtNkFGLTkyMEItMUUtNkMtN0QtNEZFLTgzMEMtNEItN0EtOUQtM
kNFLTE1MEQtNkYtOEMtMUUtM0JELTc0MEUtN0QtMkItNUMtOUFFLTYwMEYtNUMtM0QtOEItNEFFLTI3MUEtOUQtMk
MtM0UtN0JGLTU4MUItOEUtNEYtN0EtNkRFLTIwMUMtN0EtM0UtNUYtOUNGLTQ2MUQtNkItNEEtOEYtMkJFLTA5MUU
tNEYtOUQtNUMtM0FFLTY3MUYtM0EtNkItMkQtOEZFLTUyMkEtMUQtOEYtOUMtNUJFLTc2MkItMEMtNUEtM0QtNEZF
LTgxMkMtOUItN0UtOEQtMUFGLTM0MkQtOEMtNkYtNUUtMERFLTE0MkUtN0YtNEItNkEtOUNFLTg5MkYtNkUtMEQtM
UMtM0ZFLTU3M0EtMEQtOEYtN0UtNkJGLTQyM0ItOUMtMkUtMUYtNEFFLTYzM0MtOEItNUQtMEYtN0RFLTI5M0QtN0
EtMUMtMkItOUZFLTUxM0UtNkYtNEEtOUMtNUNFLTM4M0YtNUUtN0ItOEQtMkFGLTY0NEEtN0QtNUUtMUMtNkVGLTI
wNEItNkMtOEYtM0EtOURGLTQ4NEMtNUItMkQtNkUtN0ZFLTE3NEQtNEEtOUMtMEUtNUJGLTg1NEUtM0YtMUItN0Mt
OEFFLTYyNEYtMkUtNkEtNUQtMEJFLTM2NUEtMUMtOUUtN0ItM0FGLTU4NUItMEQtNEMtNkYtMkZFLTQ3NUMtOUUtM
kItM0QtMURFLTg5NUQtOEYtNkEtNEMtN0NFLTIwNUUtN0ItM0YtOEQtMEJGLTY1NUYtNkEtMEUtOUItNEFGLTM0Nk
EtNUQtN0YtM0UtMkNFLTEyNkItNEUtMUQtOEMtOUZFLTYwNkMtM0YtNUEtN0QtMEFFLTgzNkQtMkEtMEYtNEMtOEJ
FLTU0NkUtMUItM0MtNUYtN0ZFLTI2NkYtMEMtMkUtMUEtOUFGLTQ5N0EtOUQtOEItNkMtMURFLTMwN0ItOEUtN0Yt
NEQtMEZFLTc2N0MtN0EtNUYtMkQtNEJFLTU4N0QtNkYtM0UtMUMtOEFFLTkyN0UtNUUtNUItMkItMEQtOUZFLTMxN0Yt
U5OEQtMEEtNkYtNEItNUZFLTcxOEUtOUQtNUMtMkItMEFFLTM3OEYtOEMtNEEtMUQtN0JFLTIwOUEtN0YtNUUtM0M
tMERFLTUxOUItNkUtNEQtMkYtOENFLTQzOUMtNUQtM0ItMUEtNkZFLTc5OUQtNEMtMkEtMEUtN0FFLTY1OUUtM0It
MUQtNkMtNUZFLTg0OUYtMkEtMEMtOEItNERFLTI5MEEtMUQtM0YtNUUtNkZFLTQ3MEItMkUtNEEtNkQtN0NFLTYxM
EMtM0YtNUQtOEItOUFFLTI1MEQtNEEtNkMtOUYtOEJFLTMyMEUtNUItM0QtOEMtMUFGLTU0MEYtNkMtOEUtMUItMk
RFLTcwMUEtN0QtOUYtMEEtMkYtOENFLTY4MUItMEUtNkYtM0UtMkFFLTU3MUMtNUYtN0ItM0QtNEZFLTg5MUQtNEEtNkM
```

4344 1

Output

```
540F-6C-8E-1B-2DE-701A-7D-9F-8E-8CE-641B-6E-8A-2F-3AE-571C-5F-7B-3D-4FE-891D-4A-6C-5E-0AE-321E-3B-5D-7C-9FE-
461F-2A-4E-6B-8BF-512A-1D-3F-5E-4FE-872B-0E-2C-8D-9AE-642C-9F-7E-6B-5DE-102D-8A-6F-4C-3FE-572E-7B-5D-3A-1AE-
842F-6C-4E-2F-0BF-913A-5D-7F-9E-8CE-603B-4E-6A-8F-0AE-293C-3F-5D-7C-9BE-483D-2A-4C-6B-8FE-573E-1B-3D-5E-0DE-
743F-8C-2E-4F-6AE-854A-9D-1F-3C-5BE-604B-8E-0A-2D-7FE-434C-7F-9E-1C-6AE-524D-6A-8C-0F-5BE-374E-5B-7D-9C-4FE-
614F-4A-6C-8B-3AF-285A-3D-5F-7E-2CE-465B-2E-4A-6D-1FE-375C-1F-3B-5D-0AE-585D-0A-2C-4E-6BF-815E-9B-7D-1F-8FE-
695F-8C-6E-8D-3AE-726A-7D-5F-2C-1BE-806B-6E-4A-0D-9FE-516C-5F-3D-1B-8AE-766D-4A-2C-0E-7BF-936E-3B-1D-7F-5DE-
286F-2C-8E-8F-4AE-597A-1D-9F-7E-3CE-487B-0E-8A-6D-5FE-177C-9F-7E-5B-4DE-237D-8A-6F-4C-3FE-527E-7B-5D-3A-2AE-
897F-6C-4E-1B-0BF-618A-5D-3F-1C-9AE-758B-4E-2A-0D-7FE-388C-3F-1D-6C-5BE-498D-2A-0C-8F-4DE-618E-1B-9D-7E-6FE-
238F-8C-8E-5B-2AE-509A-7D-5F-3C-1FE-899B-6E-4A-2D-0BE-579C-5F-3D-1B-8AE-469D-4A-2C-0E-7BF-349E-3B-1D-7F-6DE-
289F-2C-0E-8F-5AE-140A-1D-9F-7C-6DE-520B-0E-8A-5D-4FE-610C-9F-7E-3B-2AE-480D-8A-6F-1C-0BE-370E-7B-5D-2A-9FE-
260F-6C-4E-8F-8CE-151A-5D-3F-1C-0DE-641B-4E-2A-8D-9FE-431C-3F-1D-6C-7BE-821D-2A-0C-5F-8AE-491E-1B-9D-4E-3FE-
781F-8C-8E-3B-2AE-562A-9D-7F-5C-4BE-622B-8E-6A-3D-2FE-812C-7F-5E-1B-0CE-392D-6A-4C-2F-9DE-072E-5B-3D-0C-8FE-
682F-4C-2E-9F-7AE-303A-3D-1F-8C-5BE-473B-2E-0A-7D-6FE-583C-1F-9E-6B-4DE-2908-2E-9A-4B-7F-613D-0A-8C-5F-3AE-
703E-9B-7D-4F-2CE-863F-8C-6E-3D-1BE-924A-7D-5F-2C-0AE-834B-6E-4A-1D-9FE-704C-5F-3D-0B-8AE-564D-4A-2C-9E-7BF-
454E-3B-1D-8F-6DE-324F-2C-0E-7B-5AE-185A-1D-9F-6C-4BE-275B-0E-8A-5D-3FE-465C-9F-7E-2B-1AE-585D-0A-6F-1C-0DE-
795E-7B-5D-0A-9CE-895F-6C-4E-3F-8BE-926A-5D-3F-2C-1AE-046B-4E-2A-9D-8FE-306C-3F-1D-8B-7BE-636D-2A-0C-7F-6DE-
956E-1B-9D-6E-5FE-476F-0C-8E-5B-4AE-727A-9D-7F-4C-3FE-567B-8E-6A-5D-2AE-417C-7F-5E-3B-1CE-687D-6A-4C-1F-0DE-
907E-5B-3D-0C-9FE-727F-4C-2E-9F-8AE-518A-3D-1F-8C-7BE-248B-2E-0A-7D-6FE-498C-1F-9E-6B-5DE-608D-0A-8C-4F-3AE-
858E-9B-7D-2E-1CE-708F-8C-6E-5F-8BE-539A-7D-5F-4C-3FE-269B-6E-4A-3D-2AE-899C-5F-3D-2B-1CE-489D-4A-2C-1E-0BE-
739E-3B-1D-0F-9AE-54
```

What languages does Sigma use?

YAML ✓ Correct

I looked at the Sigma paragraph in the task and read what syntax Sigma uses.

Sigma

Sigma is a highly flexible open-source tool that describes log events in a structured format. Sigma can be used to find entries in log files using pattern matching. Sigma is used to:

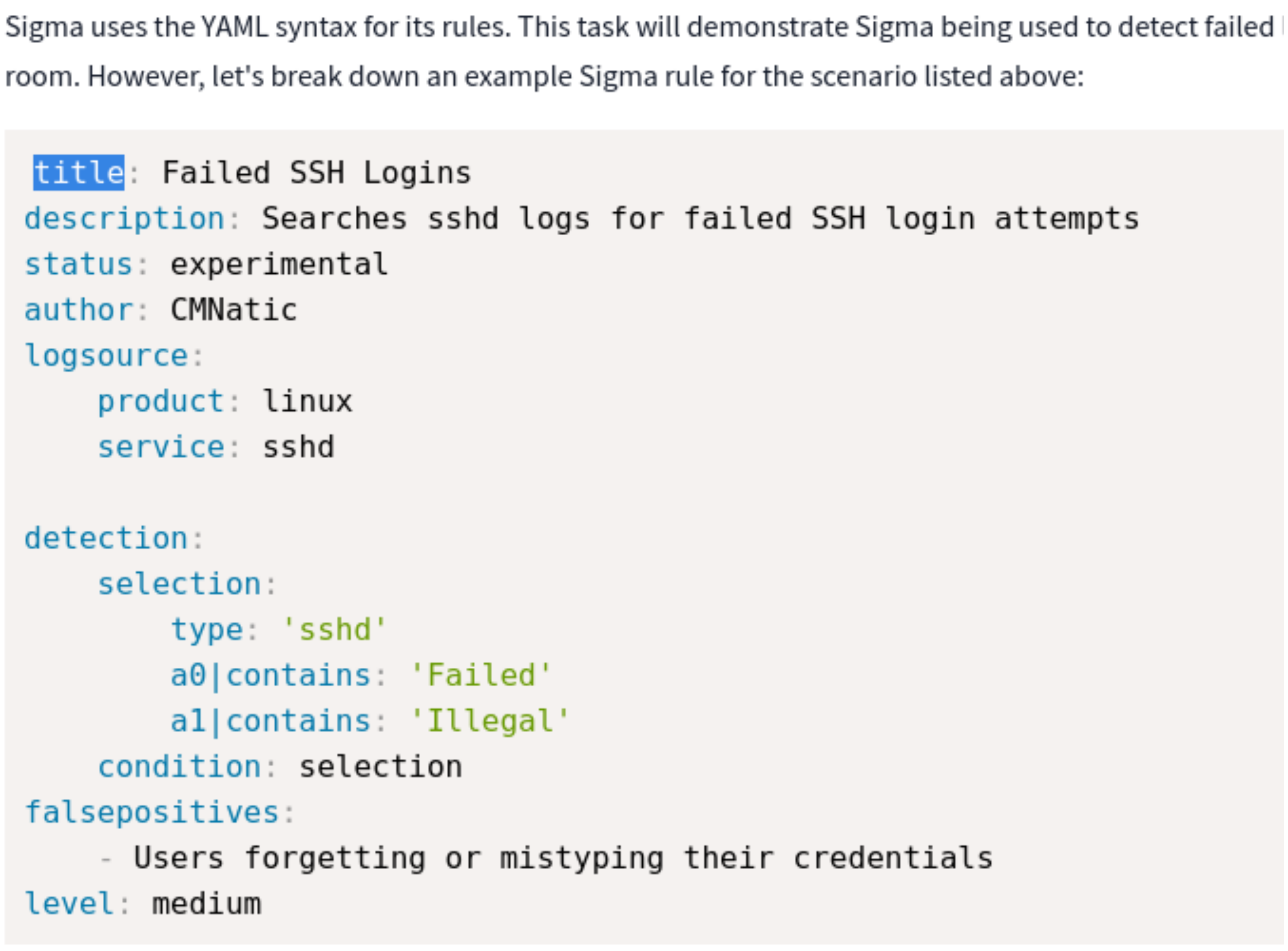
1. Detect events in log files
2. Create SIEM searches
3. Identify threats

Sigma uses the YAML syntax for its rules. This task will demonstrate Sigma being used to detect failed login events in SSH. Please note that writing a Sigma rule is out-of-scope for this room. However, let's break down an example Sigma rule for the scenario listed above:

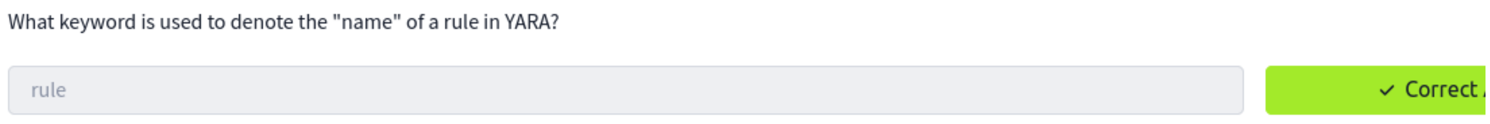
What keyword is used to denote the "title" of a Sigma rule?

title ✓ Correct

Looking at the Sigma syntax, we can pretty easily spot the required keyword which was “title”

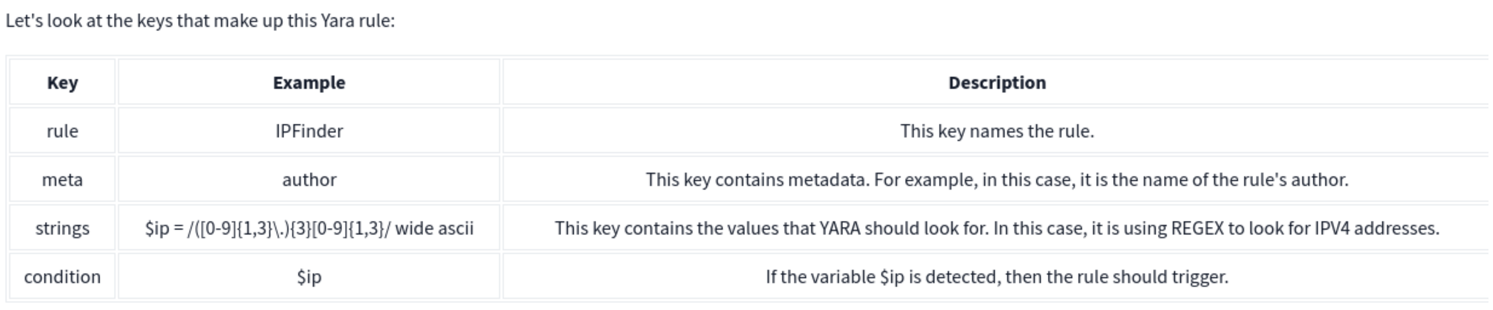

Sigma uses the YAML syntax for its rules. This task will demonstrate Sigma being used to detect failed
room. However, let's break down an example Sigma rule for the scenario listed above:

```yaml
title: Failed SSH Logins
description: Searches sshd logs for failed SSH login attempts
status: experimental
author: CMNatic
logsource:
    product: linux
    service: sshd

detection:
    selection:
        type: 'sshd'
        a0|contains: 'Failed'
        a1|contains: 'Illegal'
    condition: selection
falsepositives:
    - Users forgetting or mistyping their credentials
level: medium
```

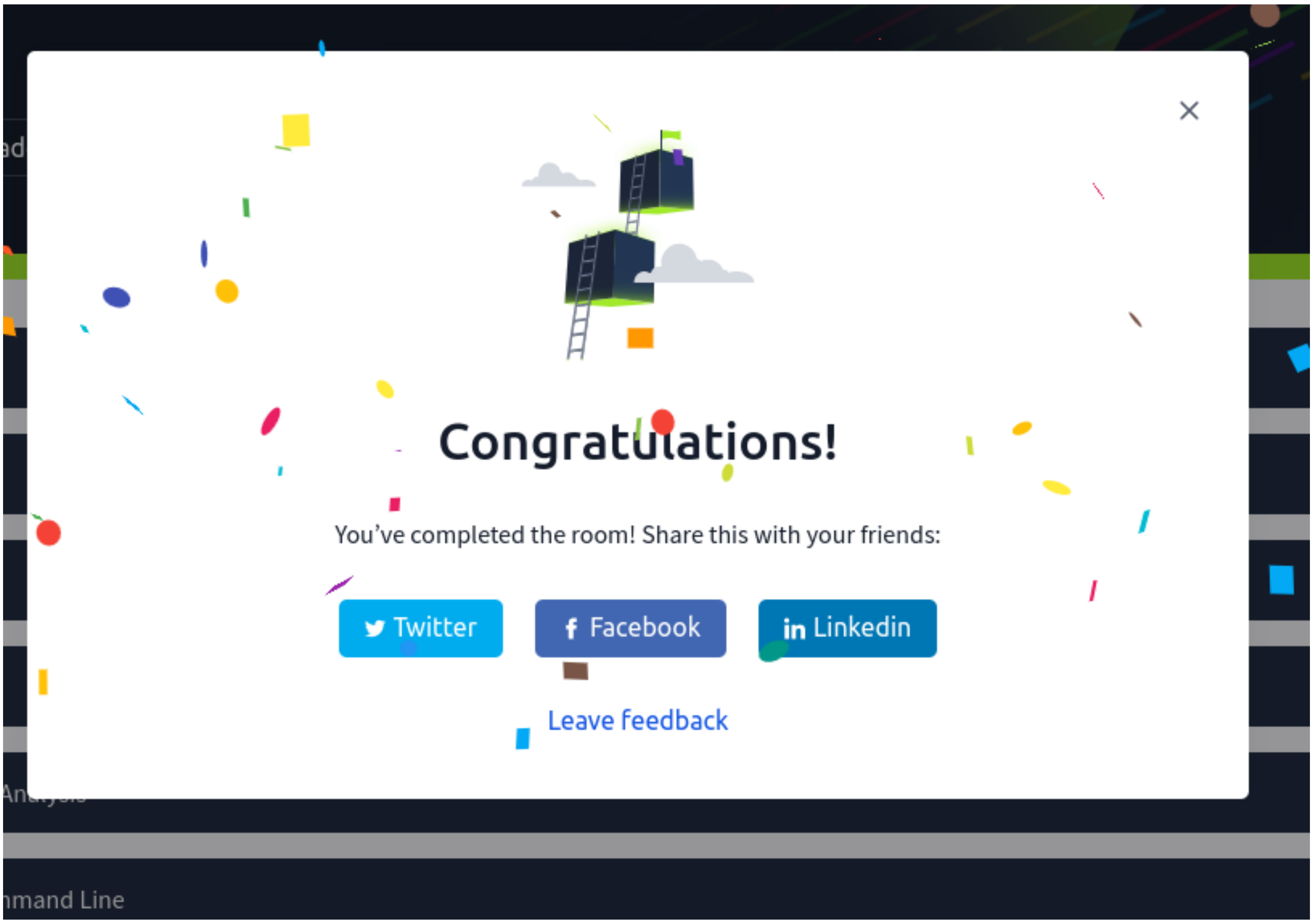

What keyword is used to denote the "name" of a rule in YARA?

rule

I looked at the YARA paragraph in the task and read what syntax YARA uses as seen in the image below.


Let's look at the keys that make up this Yara rule:

Key	Example	Description
rule	IPFinder	This key names the rule.
meta	author	This key contains metadata. For example, in this case, it is the name of the rule's author.
strings	$ip = /([0-9]{1,3}\.){3}[0-9]{1,3}/ wide ascii	This key contains the values that YARA should look for. In this case, it is using REGEX to look for IPV4 addresses.
condition	$ip	If the variable $ip is detected, then the rule should trigger.



https://tryhackme.com/r/room/introtologanalysis

CONCLUSION
In this room, we covered the basic methodology behind adopting an effective log analysis strategy. We explored the importance of log data collection, common attack patterns, and useful tools for the investigation and response processes.
A couple of tools I have interacted with in this room and ouside this room majorly are used by blueteamers.

https://threatfox.abuse.ch/
https://tryhackme.com/jr/splunkdashboardsandreports
https://github.com/log2timeline/plaso
https://plaso.readthedocs.io/en/latest/