

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



TryHackMe: Intro to Log Analysis Walkthrough



Igor Aleksandrović · [Follow](#)

11 min read · Oct 4, 2023



Listen



Share



More



Let's dive into the TryHackMe “Intro to Log Analysis” room.

I highly recommend trying to solve the room yourself, and only using this resource if you are stuck and unable to proceed.

The first two tasks are without questions, but you should focus on reading them thoroughly since they lay the foundations for log analysis and will help you better understand the following tasks.

TASK 3 — Investigation Theory

This task explains the concepts of timelines, data visualisation and threat intel. At the end of the task, there are 2 questions:

Q1: What's the term for a consolidated chronological view of logged events from diverse sources, often used in log analysis and digital forensics?

Read the explanation of the different timelines listed in this task. We are 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.

A1: Super Timeline (or Consolidated Timeline)

Q2: Which threat intelligence indicator would `5b31f93c09ad1d065c0491b764d04933` and `763f8bdbbc98d105a8e82f36157e98bbe` be classified as?

Let's look at the provided indicators. They look similar. If you look closely, they both consist of 32 hexadecimal characters, which is characteristic of an output from the MD5 algorithm. Look under the "Threat Intel" paragraph to find the answer to the question.

A2: File Hashes

TASK 4 — Detection Engineering

This task encompasses common log file locations on Linux systems, common patterns for identifying suspicious behaviour, and common attack signatures. There are 2 questions at the end of the task:

Q1: What is the default file path to view logs regarding HTTP requests on an Nginx server?

Refer to the first paragraph of the task in order to find the correct path. The question is about HTTP requests, and to interact with a web server we need to access it.

A1: `/var/log/nginx/access.log`

Q2: A log entry containing `%2E%2E%2F%2E%2Fproc%2Fself%2Fenviron` was identified. What kind of attack might this infer?

This string contains multiple `%2E` and `%2F` characters, which indicates a URL encoding. We know from the explanation in the task that `%2E` refers to the URL encoding of the character `'.'` `%2F` refers to `'/'`. Decoding the string yields `"../proc/self/enviro`n". the `'../'` portion of the string indicates an attempt to back

out of the original directory and “**proc/self/enviro**n” indicates an attempt to access the environmental variables of a process. Refer to the common attack signatures section of the task to find the attack that shares these indicators.

A2: Path traversal

TASK 5— Automated vs. Manual Analysis

This short task explains the pros and cons of automated and manual analysis. The 2 questions at the end are quite trivial so I won't dwell on them much.

Q1: A log file is processed by a tool which returns an output. What form of analysis is this?

A1: Automated

Q2: An analyst opens a log file and searches for events. What form of analysis is this?

A2: Manual

TASK 6 — Log Analysis Tools: Command Line

Now we are getting into the meat of this room. At the moment I am writing this, there is no /root/Rooms/IntroToLogAnalysis folder on the AttackBox, so you will need to download the task files manually.

I suggest accessing the task files with the Linux terminal and following along with the task.

At the end of the task, we find some more questions.

Q1: Use a combination of the above commands on the apache.log file to return only the URLs. What is the flag that is returned in one of the unique entries?

To answer this question we need to use the cut command to return only URL files. Refer to the “cut” paragraph in the task.

We want to use a space character as a delimiter and we will accomplish that by specifying the command option -d which will specify the delimiter and the character “ ” that signifies a space. Next, we need to tell the command which field we need to return using the option -f. The field is simply a string bound by the

delimiter. Let's look at one line in the apache.log. For simplicity, we will stop at the HTML code portion of the log (9).

```
145.76.33.201 | - | - | [31/Jul/2023:12:34:20 +0000] | "GET | /login.php | HTTP/1.1" | 200
```

1 2 3 4 5 6 7 8 9

apache.log structure

As you can see, if we wanted to return the IP address we would use -f 1, if we wanted the timestamp we would use -f 4, if we wanted the request method we would use -f 6, etc...

Now that we know all that, we can combine the options and form a command:

```
cut -d ' ' -f 7 apache.log
```

We used the -f 7 option because we wanted to get the URL from the HTTP request. Scroll through the returns and you should be able to see the flag.

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

URL flag

A1: c701d43cc5a3acb9b5b04db7f1be94f6

Q2: In the apache.log file, how many total HTTP 200 responses were logged?

To answer this question we need to filter all lines with the HTTP 200 (OK) response, and then count the number of the responses. To filter the lines we will use the awk command that can filter the file based on certain conditions.

awk command automatically splits the log lines into fields, so we need to specify what are we looking for and in which field. We need the status code field, so we can reference the earlier apache.log structure and see that the status codes are in the 9th field of the log. In was equality is specified by the operator "==". The code should look like this:

```
awk '$9 == 200' apache.log
```

Now we have all entries with the HTML status code 200, but how do we count them? There are multiple ways to do the counting, using grep, sed and awk, but we will focus on the easiest one.

Since we know that each line in the apache.log is one log entry and that the command wc counts lines, words, and characters in a specified file, we will simply pipe the output of the awk command into the wc command. Since we don't need words or characters we will use option -l to tell the wc command only to return lines.

```
awk '$9 == 200' apache.log | wc -l
```

And just like that, we have the number of lines that represent the number of entries in the log file.

A2: 52

Q3: In the apache.log file, which IP address generated the most traffic?

To answer this question we first need to select only the IP addresses from the logs. We can do that with the cut command. Since we need the first string, the command will look something like:

```
cut -d ' ' -f 1 apache.log
```

We now have a list of IP addresses taken from the logs. Note that we have not selected unique addresses yet, we just took the IP address from each line in the log file.

```
user@tryhackme$ cut -d ' ' -f 1 apache.log
203.0.113.42
120.54.86.23
185.76.230.45
201.39.104.77
112.76.89.56
211.87.186.35
156.98.34.12
```

listing IP addresses with cut command

After that, we should remove duplicates so that only unique addresses remain. Since the `uniq` command identifies and removes adjacent duplicate lines from sorted input, we need to sort all IP addresses in order. We can do that with the `sort` command using the numerical `-n` option. We just need to pipe the output of the `cut` command into `sort`.

```
cut -d ' ' -f 1 apache.log | sort -n
```

```
user@tryhackme$ cut -d ' ' -f 1 apache.log | sort -n
76.89.54.221
76.89.54.221
76.89.54.221
76.89.54.221
76.89.54.221
76.89.54.221
77.188.103.244
99.76.122.65
104.76.29.88
```

Sorted IP addresses

We now have the IP addresses sorted, so we can easily remove duplicates with the `uniq` command. We also want to count how many times each address appears, so we

will use the option -c along with the uniq command.

```
cut -d ' ' -f 1 apache.log | sort -n -r | uniq -c
```

We now have a list of addresses and the number of times each address appears in the logs. Scan through the list and find the IP address with the highest count (HINT: it is 8).

A3: 145.76.33.201

Q4: What is the complete timestamp of the entry where 110.122.65.76 accessed /login.php?

To answer this question we need to find the log that contains both the correct IP and the URL /login.php. To accomplish that we will use the grep command. Since we need to find 2 strings and since both conditions need to be met, we need to introduce the “AND” operator into grep. Since there is no native AND operator in grep, we need to simulate it, and we will do so using -E option. This option enables grep to interpret patterns as extended regular expressions.

```
grep -E 'pattern1.*pattern2' filename
```

The .* operator indicates the end of the pattern and the beginning of pattern2. So our command would look like this:

```
grep -E '110.122.65.76.*login.php' apache.log
```

Our query returned only 1 result in the log file.

Now we only need to take the timestamp as the answer to the question.

A4: 31/Jul/2023:12:34:40 +0000

TASK 7 — Log Analysis Tools: Regular Expressions

After task 6, task 7 looks like a breeze. Read carefully the explanations for the regular expressions (RegEx) since we will need them in task 8.

At the end of the task there are 2 questions.

Q1: How would you modify the original grep pattern above to match blog posts with an ID between 22–26?

Now let's look into the original command.

```
grep -E 'post=1[0-9]' apache-ex2.log
```

The grep RegEx patterns are clearly explained in this task so this should be easy. We know that `post=1` represents the exact match, and `[]` marks the group of characters that can be matched, meaning that the character after 1 can be any number between 0 and 9. Our command to find ID-s between 22–26 would look like this:

```
grep -E 'post=2[2-6]' apache-ex2.log
```

A1: `post=2[2-6]`

Q2: What is the name of the filter plugin used in Logstash to parse unstructured log data?

Just read the last paragraph of the task.


A2: Grok

TASK 8 — Log Analysis Tools: CyberChef

In this task, we will be introduced to a powerful tool called CyberChef. You would do well to familiarize yourself with this tool because it enables a security analyst to do a great many things with ease.

Q1: 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?

First, upload the file into CyberChef by following the instructions in the task. Select regular expression from the recipes and choose IPv4 addresses from the list of built-in regexes. Finally, select the option to list all matches.



Regex IPv4

The output should look something like this:

Output

```
143.110.222.166
45.79.172.21
108.0.0.0
34.76.158.233
74.50.79.238
157.245.43.144
209.159.153.74
209.159.153.74
157.245.43.144
85.200.247.5
134.122.118.79
134.122.118.79
20.55.53.144
143.110.222.166
198.235.24.195
54 36 115 221
```

Now, we could manually scroll through all addresses until we find what we are looking for, but that would be inefficient and boring. Instead, we will add another operation to our recipe. This operation will further filter the IP output (IP addresses) of our first operation.

Let's try to adjust the simple IP regex from the task to suit our purpose.

```
\b([0-9]{1,3}\.){3}[0-9]{1,3}\b
```

We only need to change the first octet to a specific value and reduce the number of repetitions from 3 to 2. The code should look something like this:

```
\b212\.([0-9]{1,3}\.){2}[0-9]{1,3}\b
```

The screenshot shows the CyberChef 'Recipe' interface with two operations. The first operation is a 'Regular expression' filter with the built-in regex 'IPv4 address'. The second operation is also a 'Regular expression' filter, but with a 'User defined' regex. The regex for the second operation is `\b212\.([0-9]{1,3}\.){2}[0-9]{1,3}\b`. Both operations have the 'Case insensitive' checkbox checked. The 'Output format' for both is set to 'List matches'.

Recipe

Regular expression

Built in regexes
IPv4 address

Regex
(?: (?: \d | [01]? \d \d | 2[0-4] \d | 25[0-5]) \.) {3} (?: 25[0-5] | 2[0-4] \d | [01]? \d \d | \d) (?: \ / \d {1,2}) ?

☒ Case insensitive

☒ ^ and \$ match at newlines ☐ Dot matches all ☐ Unicode support ☐ Astral support ☐ Display total

Output format
List matches

Regular expression

Built in regexes
User defined

Regex
\b212\.([0-9]{1,3}\.){2}[0-9]{1,3}\b

☒ Case insensitive

☒ ^ and \$ match at newlines ☐ Dot matches all ☐ Unicode support ☐ Astral support ☐ Display total

Output format
List matches

Adding another layer to CyberChef recipe

The output should be the address that we are looking for.

A1: 212.14.17.145

Q2: Using the same log file from Question #2, a request was made that is encoded in Base64. What is the decoded value?

First, clear the recipe so that we are working with a blank slate. How can we find a Base64 string? One of the characteristics of Base64 is that it is a multiple of 3. If the Base64 string doesn't have a multiple of 3 characters (3,6,9,12, etc.) then the padding at the end will be displayed as "=" signs. For example, a Base64 string with 4 characters would be "dXN==". We can use that knowledge to help us find the string.

We need to find a request in the log. HTTP requests in this log are either GET or POST requests. So we need the RegEx that will find any string that starts with either GET or POST and ends with an = sign, with however many characters in the middle. We can now build our expression:

```
(POST|GET) /.*=
```

the ./ sign matches 0 any character zero or more times. When we apply the expression to CyberChef we get the following output:

Output

```
GET /dns-query?name=dnsscan.shadowserver.org&type=  
GET /?XDEBUG_SESSION_START=  
GET /VEhNe0NZQkVSQ0hFRl9XSVPBUkR9==  
GET /viwwwsogou?op=8&query=
```

Now, one of those lines looks suspiciously like a Base64. Can you guess which?

Yes, it is VEhNe0NZQkVSQ0hFRl9XSVPBUkR9.

We take the string and feed it again to the CyberChef, and select the operation to decode from Base64. Voila, we have our flag!

Q3: Using CyberChef, decode the file named “encodedflag.txt” and use regex to extract by MAC address. What is the extracted value?

To answer this question load the file “encodedflag.txt” into our skilled Chef and ask him to decode it from Base64. In the output, we will have what is seemingly a list of MAC addresses.

Fake MAC adresses

Open in app



Fortunately, we have our trusted Chef. Just add another operation that uses regular expressions to find a MAC address (it is one of the pre-defined expressions).

Recipe

From Base64

Alphabet
A-Za-z0-9+/,=

☒ Remove non-alphabet chars ☐ Strict mode

Regular expression

Built in regexes
MAC address

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

Finding the MAC

And now, in the middle of the hexadecimal field, we have our MAC address highlighted.

```
BF-512A-1D-3F-5E-4FE-872B-0E-2C-8D-9AE-642C-9F-7E-6B-5DE-102D-8A
C-6B-8FE-573E-1B-3D-5E-0DE-743F-0C-2E-4F-6AE-854A-9D-1F-3C-5BE-6
65B-2E-4A-6D-1FE-375C-1F-3B-5D-0AE-585D-0A-2C-4E-6BF-815E-9B-7D-
DE-286F-2C-0E-8F-4AE-597A-1D-9F-7E-3CE-487B-0E-8A-6D-5FE-177C-9F
D-6C-5BE-498D-2A-0C-8F-4DE-618E-1B-9D-7E-6FE-238F-0C-8E-5B-2AE-5
40A-1D-9F-7C-6DE-520B-0E-8A-5D-4FE-610C-9F-7E-3B-2AE-480D-8A-6F-
AE-491E-1B-9D-4E-3FE-781F-0C-8E-3B-2AE-562A-9D-7F-5C-4BE-622B-8E
A-7D-6FE-583C-1F-9E-6B-4DE-2908-2E-9A-4B-7F-613D-0A-8C-5F-3AE-70
B-1D-8F-6DE-324F-2C-0E-7B-5AE-185A-1D-9F-6C-4BE-275B-0E-8A-5D-3F
06C-3F-1D-8B-7BE-636D-2A-0C-7F-6DE-956E-1B-9D-6E-5FE-476F-0C-8E-
AE-518A-3D-1F-8C-7BE-248B-2E-0A-7D-6FE-498C-1F-9E-6B-5DE-608D-0A
```

Found you!

A3: 08-2E-9A-4B-7F-61

TASK 9 —Log Analysis Tools: Yara and Sigma

The last task is pretty straightforward. Read the explanations and let's look at the questions in the end.

Q1: What languages does Sigma use?

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

A1: YAML

Q2: What keyword is used to denote the “title” of a Sigma rule?

If we look at the Sigma syntax, we can pretty easily spot the required keyword.

A2: title

Q3: What keyword is used to denote the “name” of a rule in YARA?

As with the previous question, look at the syntax for YARA and spot where the name of the rule is defined.

A3: rule

CONCLUSION

I hope that you answered as many questions as you could on your own and that I provided satisfactory explanations for the questions that you were stuck on.

Keep going, keep learning and you have a bright future ahead of you. :)

[Cybersecurity](#)[Tryhackme Walkthrough](#)[Tryhackme Writeup](#)[Tryhackme](#)[Follow](#)

Written by Igor Aleksandrović

8 Followers · 1 Following

No responses yet



What are your thoughts?

Respond

Recommended from Medium



In System Weakness by Joseph Alan

TryHackMe Critical Write-Up: Using Volatility For Windows Memory Forensics

This challenge focuses on memory forensics, which involves understanding its concepts, accessing and setting up the environment using tools...

Jul 18, 2024

👏 46

💬 1





In T3CH by Axoloth

TryHackMe | FlareVM: Arsenal of Tools| WriteUp

Learn the arsenal of investigative tools in FlareVM



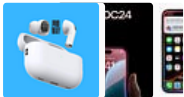
Nov 28, 2024



50

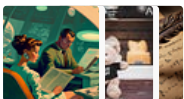


Lists



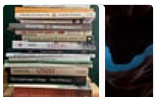
Tech & Tools

22 stories · 377 saves



Medium's Huge List of Publications Accepting Submissions

377 stories · 4319 saves



Staff picks

793 stories · 1549 saves



Natural Language Processing

1883 stories · 1522 saves

Learn > PaperCut: CVE-2023-27350

PaperCut: CVE-2023-27350

Authorisation bypass (CVE-2023-27350) in PaperCut Print Management software leading to remote code execution.

Info 30 min

Start AttackBox Help Save Room 35 Options

Room completed (100%)

- Task 1 Introduction
- Task 2 Understanding PaperCut and CVE-2023-27350
- Task 3 Exploiting CVE-2023-27350
- Task 4 Detection and Mitigation
- Task 5 Conclusion

 Reju Kole

PaperCut: CVE-2023-27350-THM-Walkthrough-By-Reju-Kole

Category—Info

Aug 23, 2024 107



 Sunny Singh Verma [SuNnY]

Linux Incident Surface TryHackMe Writeup | THM Detailed Walkthrough | SuNnY

The Linux Incident Surface refers to all potential points within a Linux system where incidents, such as security breaches or malicious...

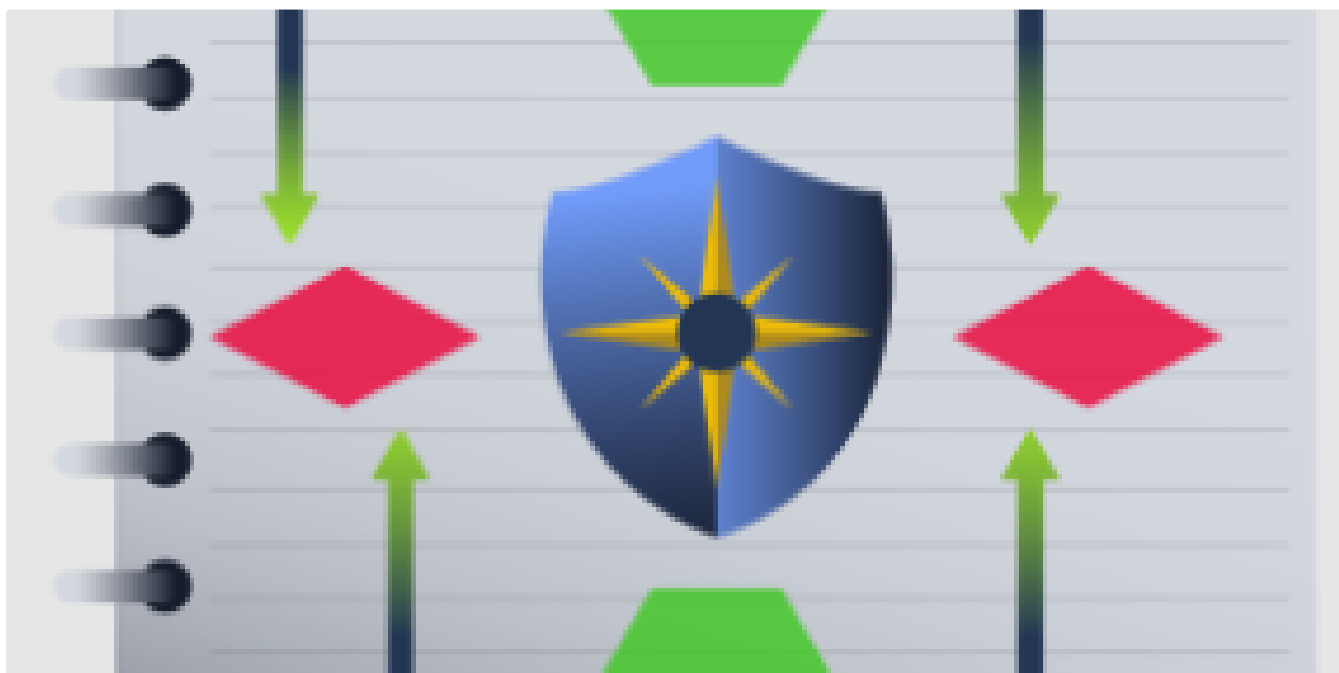
Sep 23, 2024 🖱 101

 In InfoSec Write-ups by Visir

Linux Incident Surface: The Cyber Security Challenge |Tryhackme Walkthrough

In this TryHackMe room, “Linux Incident Surface: The Cyber Security Challenge,” we dive into a scenario featuring an attacker and defender...

Sep 21, 2024 🖱 153





MAGESH

IR Playbooks-Tryhackme Writeup

Learn the basics of creating and using IR playbooks.

Sep 16, 2024



See more recommendations