

FixIT Challenge – Splunk Lab Write-Up

Table of Contents

- Introduction
- Objectives
- Lab Environment
- Tools and Technologies Used
- Step-by-Step Process
- Results and Analysis
- Challenges Faced
- Conclusion
- References

Introduction

In this lab, I took on the role of John, a candidate applying for a SOC-L2 position at MSSP Cybertees Ltd. My final challenge was to resolve event parsing and data extraction issues within a Splunk instance. This test was designed to simulate real-world log ingestion and analysis problems and evaluate my ability to diagnose and fix configuration issues.

The hands-on experience focused on identifying improperly formatted events, adjusting Splunk configuration files, writing effective regular expressions, and extracting meaningful fields for analysis.

Objectives

- Understand and resolve event boundary issues in Splunk
- Configure props.conf and associated files to handle multi-line events
- Write and test regex patterns for field extraction
- Extract critical fields: Username, Department, Domain, Source_IP, Country
- Perform basic event analysis and answer targeted questions

- Gain familiarity with troubleshooting log ingestion problems in Splunk

Lab Environment.

- Virtual machine hosted by TryHackMe (accessible via split-screen view)
- Splunk Web Interface: http://MACHINE_IP:8000
- Splunk App: [Fixit](#)
- Splunk installation path: [/opt/splunk](#)
- Data source: Custom script outputting multi-line logs

Tools and Technologies Used

- **Splunk Enterprise** (Log ingestion, configuration tuning, field extraction)
- **props.conf, transforms.conf, fields.conf** (for parsing and transformation)
- **Regex** (Pattern-based field extraction)
- **Linux Terminal** (Navigating and editing configuration files)
- **ChatGPT** (Assistance with regex syntax and config structure)

Step-by-Step Process

The screenshot shows the Splunk Enterprise interface. At the top, there's a navigation bar with 'splunk-enterprise' and 'Apps'. Below it, a 'New Search' bar contains the query 'index=*'. The search results show 6,352 events. The interface has tabs for 'Events (6,352)', 'Patterns', 'Statistics', and 'Visualization'. The 'Events' tab is active, showing a list of events. The left sidebar has 'SELECTED FIELDS' (host, source, sourcetype) and 'INTERESTING FIELDS' (date_hour, date_mday, date_minute, date_month). The event list has columns for 'Time' and 'Event'. The events are multi-line, with some lines starting with country information (e.g., 'Brazil at: Thu Jun 26 15:01:36 2025') and others starting with network log details (e.g., '[Network-log]: User named David Lee from Marketing department accessed the resource Cybertees.THM/signup.html from the source IP 172.16.0.1 and country').

Taking a first look at our logs, we see that the events are improperly formatted. Two logs are jammed into one, and the event boundaries are not defined correctly. For example, some logs begin with country information when they should start with **[Network-log]**.

In order to properly research and analyze events, the logs need to be structured into single, complete records. To do this, we need to edit the **props.conf** file and define the correct event breaking rules.

Initial Diagnosis:

The logs are coming in as multi-line events, but Splunk is unable to determine the start of a new event.

Step 1: Review Default props.conf Suggestions

Using ChatGPT, I reviewed how to format the **props.conf** file.

i am creating a splunk project, i have been given some sample logs to investigate and my first task is to fix the logs so that a multi event is combined into one event, i would like you to guide me through the steps of fixing this multi event by editing the props.conf file, here are sample logs: 6/20/25

✓ Step-by-Step Guide

1. Identify the sourcetype

Your sample shows:

```
plaintext
```

Copy Edit

```
sourcetype = network_logs
```

We'll use this in the `props.conf` configuration.

2. Determine an appropriate event boundary

From your logs, each event seems to start with a timestamp in the format:

```
plaintext
```

Copy Edit

```
6/20/25
```

```
3:06:36.000 PM
```

Splunk breaks events using `LINE_BREAKER` and groups lines together based on `SHOULD_LINEMERGE` or `BREAK_ONLY_BEFORE` directives.

3. Create/Edit `props.conf`

Here's the configuration to group each pair of lines into one event:

```
ini
```

Copy Edit

```
[network_logs]
SHOULD_LINEMERGE = false
BREAK_ONLY_BEFORE = ^\d{1,2}/\d{1,2}/\d{2}\r?\n\d{1,2}:\d{2}:\d{2}\.\d{3} (AM|PM)
TIME_FORMAT = %m/%d/%y\n%I:%M:%S.%3N %p
MAX_EVENTS = 5
TRUNCATE = 10000
```

Here's what I learned about key parameters:

- `SHOULD_LINEMERGE = false` → Tells Splunk not to merge lines automatically
- `BREAK_ONLY_BEFORE` → Defines the regex pattern that indicates the start of a new event
- `TIME_FORMAT` → Helps Splunk extract timestamps correctly
- `MAX_EVENTS = 5` → Limits each event to 5 lines (adjustable)
- `TRUNCATE = 10000` → Prevents logs from being cut off if too long

Step 2: Create a Custom props.conf File

For this lab, I simplified the props.conf setup. I navigated to `/opt/splunk/etc/apps/fixit/default/` and created a new configuration for the source type.

```
root@tryhackme: /opt/splunk
File Edit View Search Terminal Help
[network_logs]
SHOULD_LINEMERGE = true
BREAK_ONLY_BEFORE = ^\d{1,2}/\d{1,2}/\d{2}\r?\n\d{1,2}:\d{2}:\d{2}\.\d{3} (AM|PM)
~
~
~
```

- I used `SHOULD_LINEMERGE = true` so Splunk could join lines unless instructed otherwise.
- The `BREAK_ONLY_BEFORE` stanza uses a custom regex pattern to mark the beginning of each log: `\[Network-log\]`.

Step 3: Restart Splunk

After saving the file, I restarted Splunk to apply the configuration changes.

```
root@tryhackme:/opt/splunk# vi etc/apps/fixit/default/props.conf
root@tryhackme:/opt/splunk# splunk restart
```

As shown below, the logs now appear correctly grouped, each event starting with **[Network-log]** as expected.

The screenshot shows the Splunk Enterprise interface with a search results table. The table has two columns: 'Time' and 'Event'. There are three log entries displayed, each starting with '[Network-log]'. The first entry is from 6/27/25 1:53:46.000 PM, the second is from 6/27/25 1:53:45.000 PM, and the third is from 6/27/25 1:53:44.000 PM. The interface also shows a search bar with 'index=*' and a table of selected fields on the left.

Time	Event
6/27/25 1:53:46.000 PM	[Network-log]: User named Robert Wilson from IT department accessed the resource Cybertees.THM/signup.html from the source IP 192.168.1.7 and country United States at: Fri Jun 27 13:53:46 2025 host = tryhackme : source = networks : sourcetype = network_logs
6/27/25 1:53:45.000 PM	[Network-log]: User named Robert Wilson from IT department accessed the resource Cybertees.THM/products/product1.html from the source IP 192.168.0.7 and country United States at: Fri Jun 27 13:53:45 2025 host = tryhackme : source = networks : sourcetype = network_logs
6/27/25 1:53:44.000 PM	[Network-log]: User named Bob Johnson from Finance department accessed the resource Cybertees.THM/checkout.html from the source IP 192.168.0.5 and country Mexico at: Fri Jun 27 13:53:44 2025 host = tryhackme : source = networks : sourcetype = network_logs

Step 4: Field Extraction via Regex

With events fixed, I moved on to field extraction to make analysis easier.

I launched the **Field Extractor** in Splunk and selected a sample log entry. I then used a regex pattern to pull the following fields:

- Username
- Department
- Domain
- Source_IP
- Country

List ▾ Format

< Hide Fields ≡ All Fields

SELECTED FIELDS


a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS

date_hour 2
date_mday 2
date_minute 25
a date_month 2
date_second 60
a date_wday 2
date_year 2
a date_zone 1
a index 1
linecount 2
a punct 14
a splunk_server 1
timeendpos 7
a timestamp 1
timestartpos 7

+ Extract New Fields

i	Time
>	6/27/25 2:06:19.000 PM
>	6/27/25 2:06:18.000 PM
>	6/27/25 2:06:17.000 PM
>	6/27/25 2:06:16.000 PM
>	6/27/25 2:06:15.000 PM
>	6/27/25 2:06:14.000 PM
>	6/27/25 2:06:13.000 PM



Starting by selecting a sample log, anyone will do, and clicking next.

splunk>enterprise Apps ▾

Extract Fields

Select Sample Select Method Select Fields Save **Next >**

Select Sample Event

Choose a source or source type, select a sample event, and click Next to go to the next step. The field extractor will use the event to extract fields. [Learn more](#)

I prefer to write the regular expression myself >



Source type
network_logs

Time Range
Last 90 days ▾

[Network-log]: User named Margaret King from HR department accessed the resource Cybertees.THM/checkout.html from the source IP 172.16.0.5 and country Australia at: Fri Jun 27 14:06:19 2025

Events

✓ 1,000 events (3/29/25 12:00:00.000 AM to 6/27/25 2:22:38.000 PM)



splunk>enterprise Apps ▾

Extract Fields

Select Sample Select Method Select Fields Save < Back Next

Select Method

Indicate the method you want to use to extract your field(s). [Learn more](#) [I prefer to write the regular expression myself >](#)

Source type
network_logs

```
[Network-log]: User named Margaret King from HR department accessed the resource Cybertees.THM/checkout.html from the source IP 172.16.0.5 and country Australia at: Fri Jun 27 14:06:19 2025
```

(.*?)

Regular Expression

Splunk Enterprise will extract fields using a Regular Expression.

x|y|z

Delimiters

Splunk Enterprise will extract fields using a delimiter (such as commas, spaces, or characters). Use this method for delimited data like comma separated values (CSV files).

I created the regex with help from ChatGPT and tested it in Splunk.

```
now i have the logs formatted more percisely, i want to extract the
following fields from it, Username, Department, Domain, Source_IP,
and country. please provide me with a regex pattern to use for the
field extraction function. Here are some sample logs after the tuning:
6/27/25
2:06:19.000 PM
[Network-log]: User named Margaret King from HR department
accessed the resource Cybertees.THM/checkout.html from the source
IP 172.16.0.5 and country
Australia at: Fri Jun 27 14:06:19 2025
```


Thanks, Lucas — now that your logs are consistently formatted, here's the refined **regex pattern** you can use in Splunk's **Field Extraction** function to extract the five fields you mentioned.

✓ Regular Expression for Field Extraction

regex

Copy Edit

```
\[Network-log\]: User named (?<Username>[\w\s]+) from (?<Department>\w+) department accessed the r
```

splunk-enterprise Apps

Messages Settings Activity Help Find

Extract Fields < Back Existing fields >

⚠ If you manually edit and then preview the regular expression below, you cannot return to the automatic field extraction workflow.

Use the event listing below to validate the field extractions produced by your regular expression.

Regular Expression Regular Expression Reference View in Search

\[Network-log\]: User named (?<Username>[\w\s]+) from (?<Department>\w+) department accessed the resource (?<Domain>[\w\.\-]+) from the source IP (?<Source_IP>[0-9]{1,3}[.](0|[1-9])?) and country's (?<Country>[\w\s]+) at:

Preview Save

The pattern successfully captured each field, allowing for quick searches and efficient event analysis.

List Format 20 Per Page

	Time	Event
>	6/27/25 2:43:39.000 PM	[Network-log]: User named Nancy Lewis from IT department accessed Germany at: Fri Jun 27 14:43:39 2025 host = tryhackme source = networks sourcetype = network_log
>	6/27/25 2:43:39.000 PM	[Network-log]: User named Daniel Martin from Finance department accessed Russia at: Fri Jun 27 14:43:39 2025

Hide Fields All Fields

ELECTED FIELDS

- host 1
- source 1
- sourcetype 1

INTERESTING FIELDS

- Country 9
- date_hour 1
- date_mday 1
- date_minute 22
- date_month 1
- date_second 60
- date_wday 1
- date_year 1
- date_zone 1
- Department 6
- Domain 11
- index 1
- linecount 2
- punct 14
- Source_IP 52
- splunk_server 1
- timeendpos 7
- timestamp 1
- timestartpos 7
- Username 27

Extract New Fields

Username

27 Values, 39.013% of events

Selected Yes No

Reports

Top values Top values by time Rare values

Events with this field

Top 10 Values	Count	%
Robert Wilson	338	15.655%
Alice Smith	101	4.678%
Christopher Turner	85	3.937%
Joseph Evans	85	3.937%
Patricia Allen	83	3.844%
Nancy Lewis	81	3.752%
David Lee	73	3.381%
Johny Bil	73	3.381%
Kevin Jackson	73	3.381%
Susan Walker	73	3.381%

Results and Analysis

- Splunk now ingests and displays each log entry as a single, unified event.
- All five fields (Username, Department, Domain, Source_IP, Country) were accurately extracted using the custom regex.
- I was able to answer all the challenge questions using filtered search queries.

Example Insights:

- Number of unique usernames: **28**
- Top two countries user Robert accessed the domain from: **Canada, United States**
- File accessed by user: **secret-document.pdf**
- Number of unique departments: **6**
- Domain captured: **Cybertees.THM**

Challenges Faced

Identifying the correct event boundary for multi-line logs required experimentation with **BREAK_ONLY_BEFORE**.

Regex pattern construction was initially tricky due to the inconsistent formatting of logs, but ChatGPT helped fine-tune it.

Restarting Splunk to test changes was time-consuming and required patience for verification.

Maintaining accuracy while extracting fields, especially when fields like usernames and departments varied in length and content.

Conclusion

This lab taught me how to manipulate Splunk configuration files to fix log ingestion issues and apply regular expressions to extract meaningful fields from unstructured data.

Key takeaways include:

- Understanding and configuring `props.conf`, `fields.conf`, and `transforms.conf`
- Writing and testing regex patterns for log parsing
- The importance of structured data for effective log analysis in a SOC environment

This exercise strengthened my ability to troubleshoot real-world log issues and showcased practical Splunk skills that are critical in cybersecurity operations.

References

TryHackMe – FIXIT Challenge Room

Splunk Documentation: `props.conf`, Regex in Field Extraction

ChatGPT assistance with syntax guidance and regex optimization