SOC Analyst Case Study: Log Analysis of PowerShell Commands and Brute-Force Authentication Incidents

Project Description

In this project, I was tasked with assisting Wareville's Security Operations Center (SOC) team during one of their busiest times of the year to analyze alerts and differentiate between true positives and false positives. This involved leveraging tools like Elastic SIEM to investigate suspicious activities, such as encoded PowerShell commands and unusual login patterns, to identify potential threats. By correlating events and analyzing logs, I aimed to uncover the root cause of the anomalies while supporting the team in managing alert fatigue and ensuring the security of the town during the festive season.

Scenario

This story is based on my hands-on experience from a practical lab exercise on TryHackMe's *Advent of Cyber 2024*. It was an engaging opportunity to dive into real-world cybersecurity scenarios and apply my skills in a simulated yet challenging environment. The exercise placed me in the role of an analyst in a bustling Security Operations Center (SOC), where I was tasked with safeguarding a fictional town's digital infrastructure during the festive season—a time rife with cyber threats.

Wareville is a bustling town known for its vibrant celebrations during the festive season. As the holiday spirit filled the air, the town's Security Operations Center (SOC) prepared for their busiest time of the year. With increased online activities, shopping events, and digital interactions, cyber threats were at an all-time high. The SOC team faced the critical challenge of managing a surge in alerts, distinguishing real threats from false alarms, and safeguarding the town’s digital infrastructure. To support their mission, I joined the team as an analyst, bringing a fresh perspective and technical skills to help manage the overwhelming task of monitoring and responding to potential cyber incidents.

The SOC relied on advanced tools like Elastic SIEM to analyze and correlate event data from various sources. My role was to investigate suspicious patterns, such as encoded PowerShell commands and unexpected login behaviors, which could indicate malicious activities. By collaborating with the team, I aimed to identify true positives, eliminate false positives, and

ensure a smooth, secure environment for Wareville's festive celebrations. Together, we worked tirelessly to protect the town's critical systems and maintain the trust of its residents during the most wonderful—and challenging—time of the year.

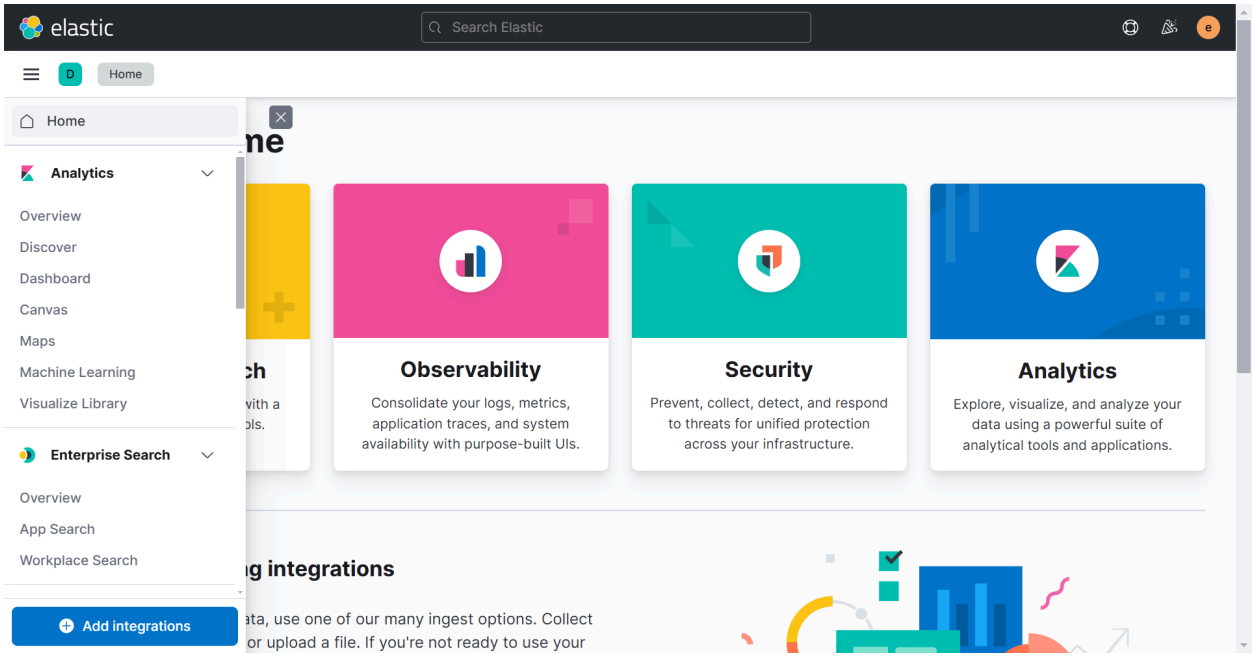
True positives or false positives?

In a SOC, events from various devices are aggregated into a SIEM, which serves as the central source of truth. Detection rules are defined to identify malicious or suspicious activities, triggering alerts when conditions are met. SOC analysts then evaluate these alerts to determine if they are True Positives (actual malicious activity) or False Positives (non-malicious activity). While this process seems straightforward in theory, distinguishing between attackers and legitimate actions, such as those by system administrators, can be challenging and time-consuming.

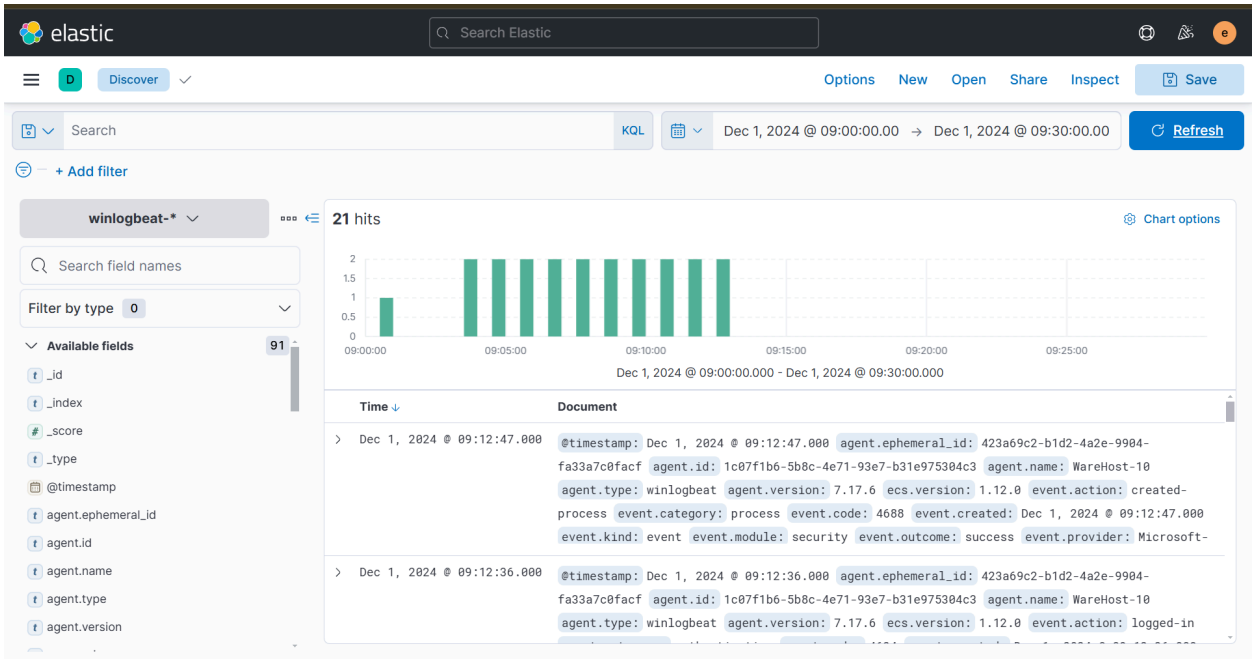
Differentiating between True Positives (TPs) and False Positives (FPs) is crucial. Misclassifying a TP as an FP could result in missing a cyber attack, while misclassifying an FP as a TP wastes time and diverts attention from real threats. To perform this task effectively, we can follow key guidelines to ensure accurate classification.

Accessing Elastic SIEM

I logged into the Elastic SIEM, then navigated to the menu on the left-hand side and clicked on the "Discover" tab to view the events.



Based on the alert from the Mayor's office, the activity took place on December 1st, 2024, between 09:00 and 09:30. To set this as our time window, we can adjust the timeframe settings in the upper-right corner. After applying the date, we see that there are 21 events within the specified timeframe.



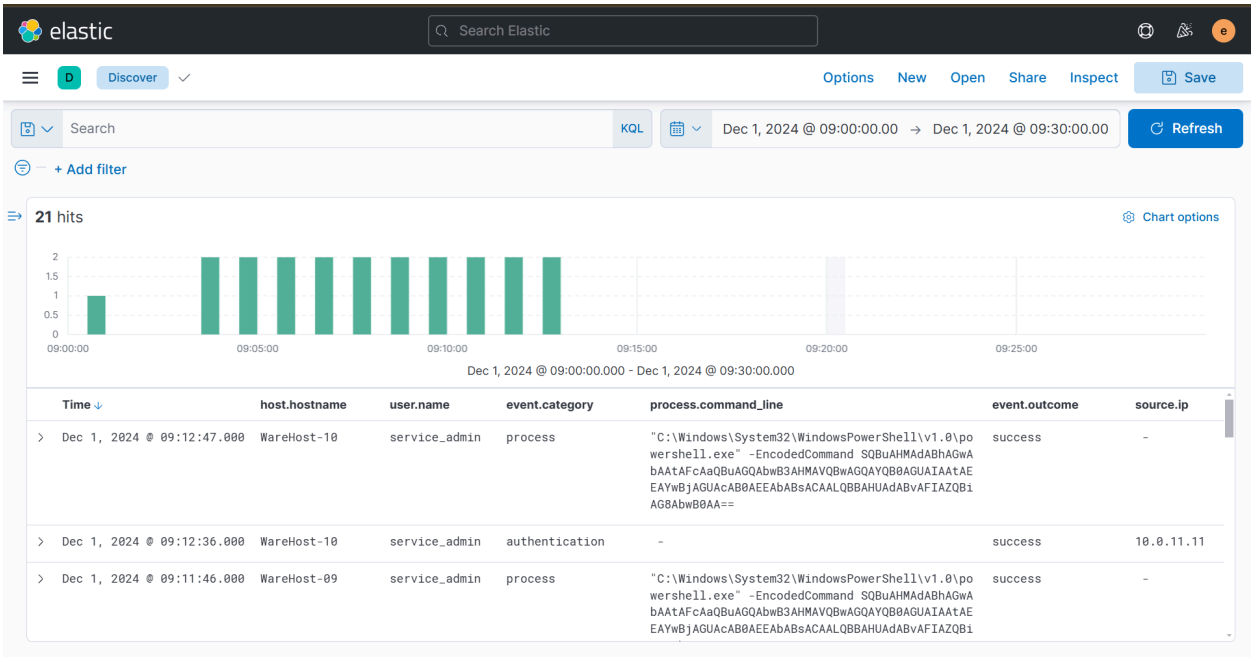
In their current form, these events don't look very easily readable. We can use the fields in the left pane to add columns to the results and make them more readable. Hovering on the field name in the left pane will allow adding that field as a column, as the fields can be seen in the picture above.

Since we are focusing on events related to PowerShell, I wanted to gather specific details from the logs.

- First, I added the **host.hostname** field as a column to identify the hostname where the command was executed.
- To find out which user performed the activity, I included the **user.name** field as a column.
- I then added the **event.category** field to ensure we are reviewing the correct event category.
- To view the actual PowerShell commands that were executed, I added the **process.command_line** field.

- To determine whether the activity succeeded, I included the **event.outcome** field.
- Lastly, to see who ran the powershell commands , I included **source.ip.**

After adding these fields as columns, the results were displayed in a more structured format, like this.

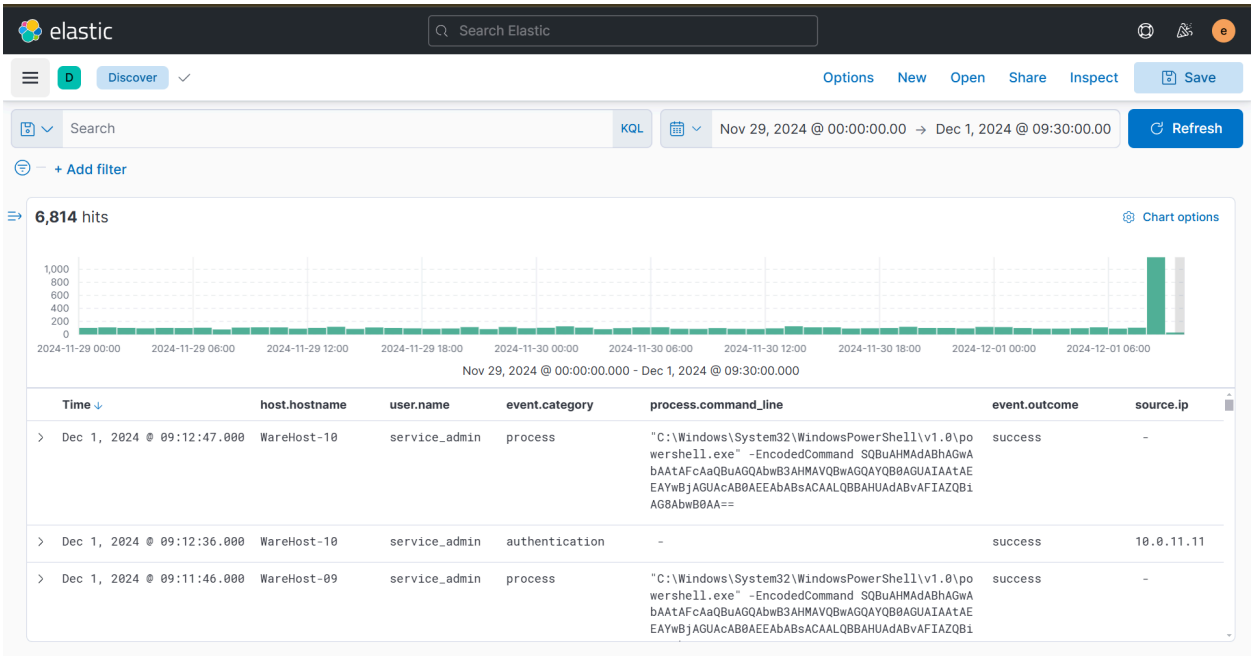


That's interesting! It appears that the same encoded PowerShell command was executed on multiple machines. Additionally, it's worth noting that before each execution of the PowerShell command, there is a successful authentication event as it can be seen by the time.

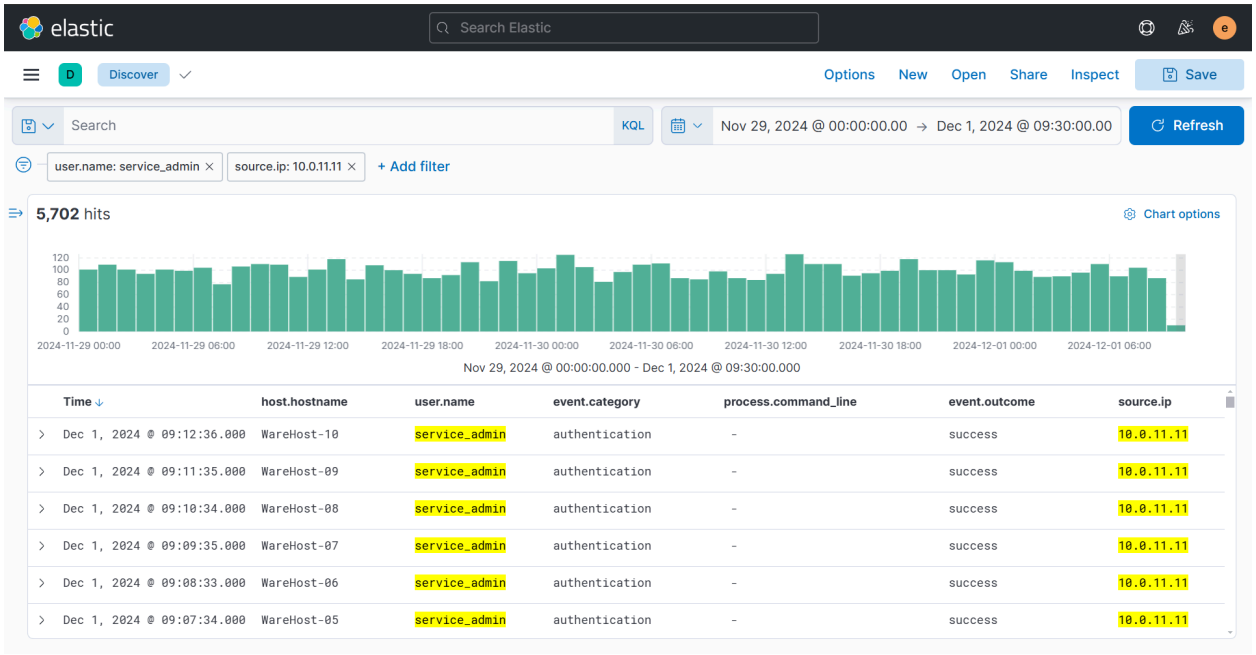
I noticed that the activity occurred individually on each machine, with a precise time gap between the login and PowerShell commands. Best practices recommend using named accounts for administrative tasks to ensure accountability, so the use of a generic admin account here raised suspicion. When I asked, the analysts informed me that this account is typically used by two administrators who weren't in the office when the activity took place. Something doesn't seem right, and I need to find out who ran these commands.

Since the timeframe we previously used was focused on the PowerShell events, the authentication events might have occurred before that. To better understand the context and historical events for this user, I will expand the search. Let's check if there

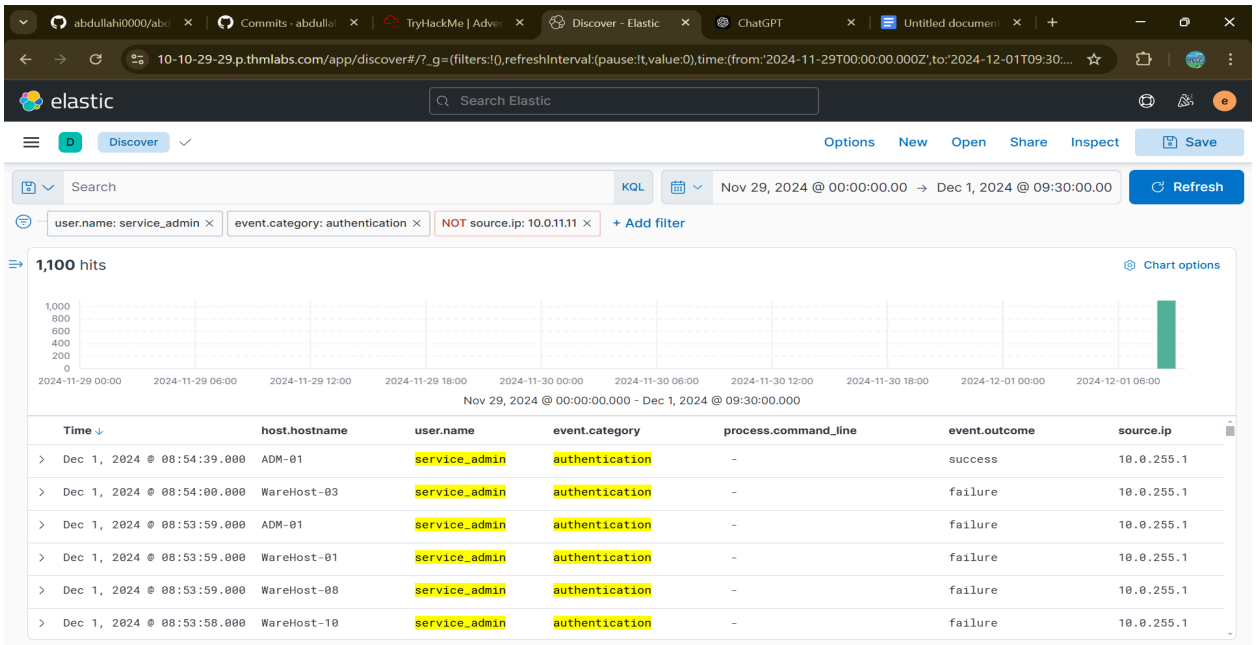
are any events for the user between November 29th and December 1st. After updating the time filter for these dates, the results appear as follows.



There were over 6,800 events in these three days, with a spike at the end of the logs. However, despite filtering for the end of December 1st, no events appeared after the successful PowerShell execution. Additionally, there were more authentication events in the days leading up to December 1st than on that day. To narrow down the search, I will filter for the user "service_admin" and the source IP "10.0.11.11".

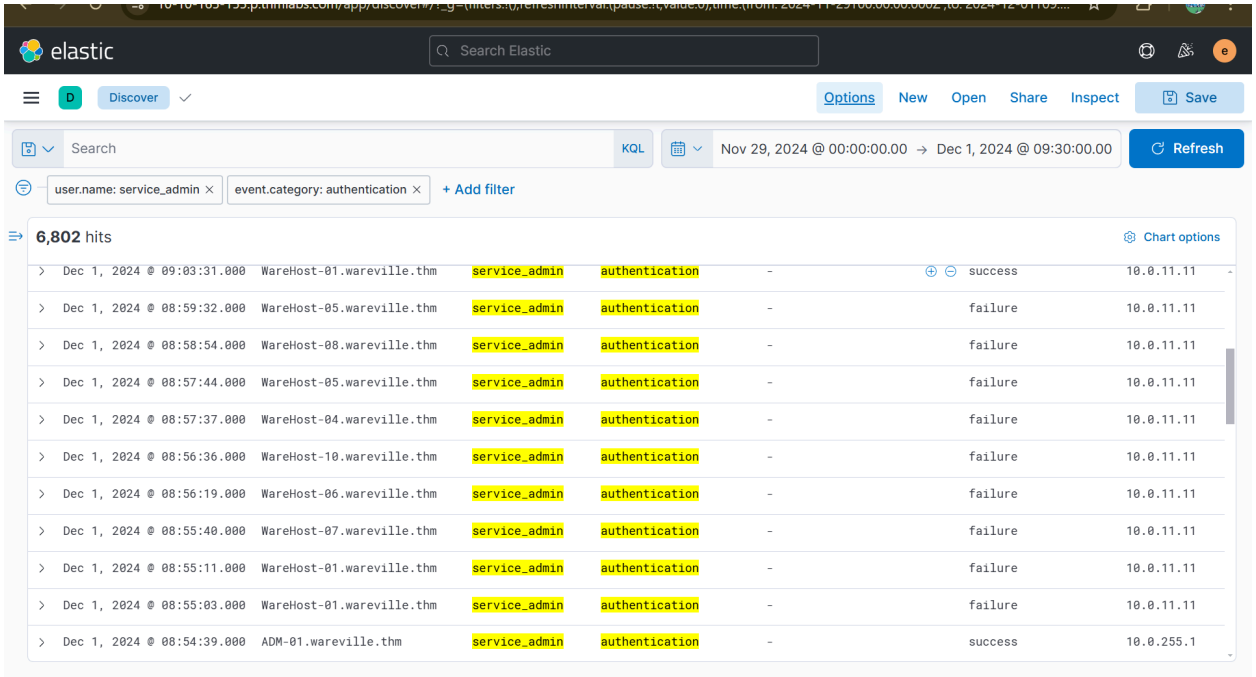


It looks like all these events have been coming from the same user and the same IP address. I definitely need to investigate further. This also does not explain the spike. I'll filter for authentication events first and then filter out the Source IP here to see if I can find the IP address that caused the spike.

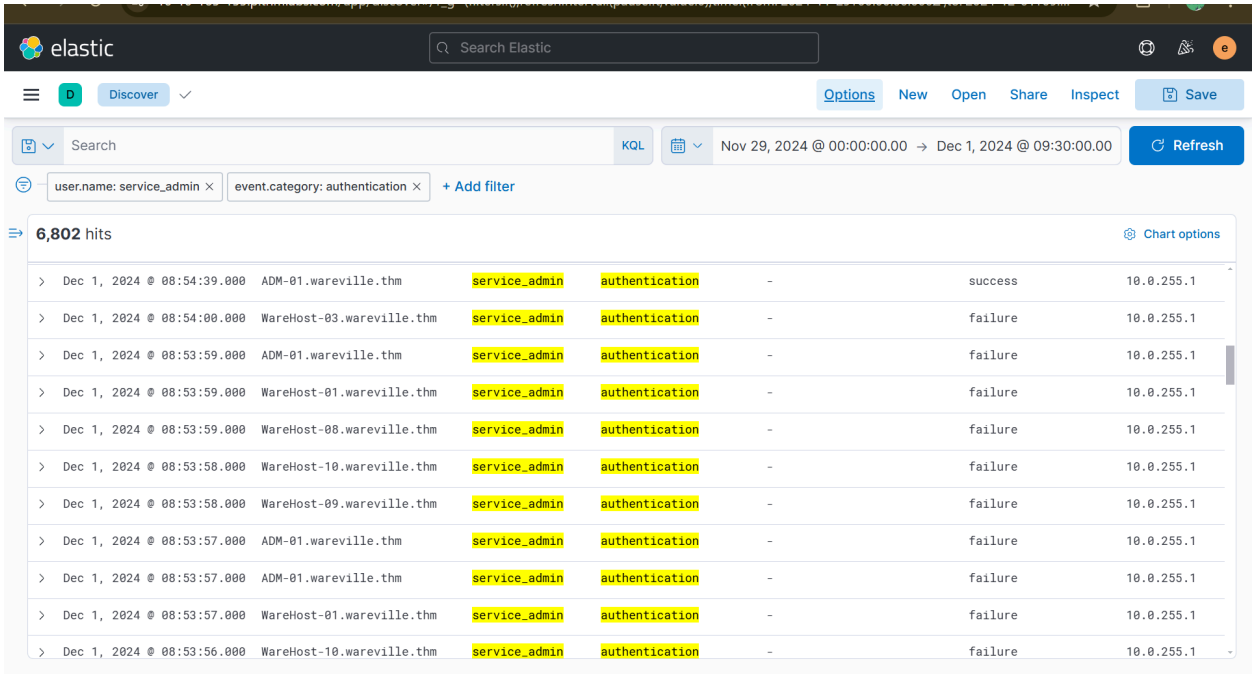


As I scroll down, I notice numerous failed login events. Additionally, the IP address for the spike (10.0.255.1) differs from the one seen in the previous days' events (10.0.11.11). The analysts previously investigated this and found that a script with expired credentials was causing the issue, but it was later updated with new credentials. This could be another script causing the issue, so let's investigate further.

I'll remove the source IP filter to focus on authentication events near the spike. After applying the new filter, we can see that the failed logins stopped shortly after the successful login from the new IP.

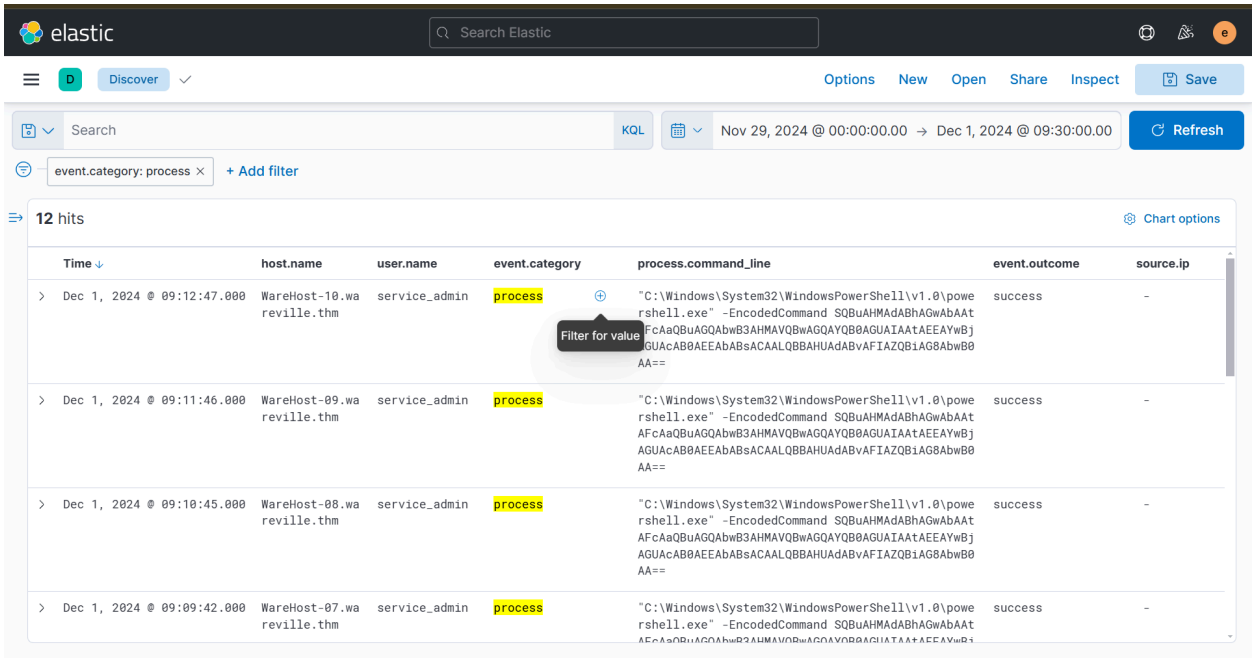


The suspicions are rising and it seems to me that someone tried a brute force attack on December 1st



The results also showed that they succeeded with the brute-force attempt because of the successful authentication attempt and quickly ran some PowerShell commands on the affected machines. Once the PowerShell commands were run, we didn't see any further login attempts. This looks like a TP, and there needs to be an escalation.

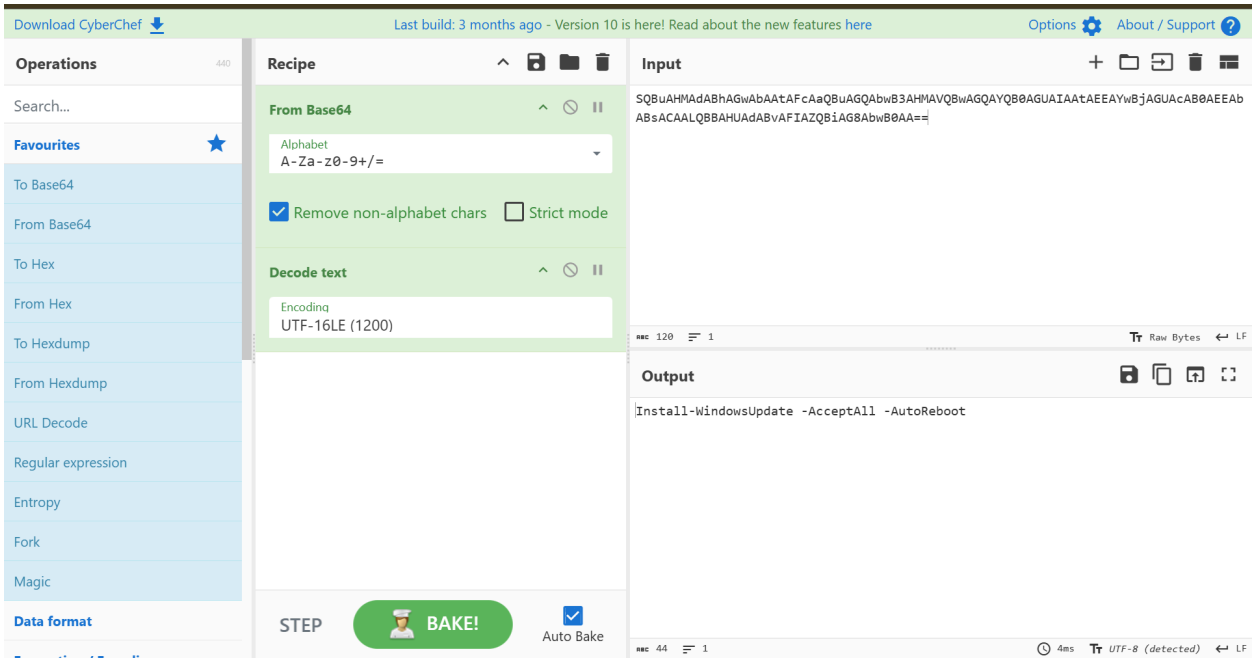
Since the PowerShell command was encoded and its contents were unknown, I needed to decode it. To investigate further, I adjusted the filters to include **event.category: process** for a closer analysis of the PowerShell commands.



We can see the PowerShell command in the **process.command_line** field.

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -EncodedCommand SQBuAHMAdABhAGwAbAAtAFcAaQBuAGQAbwB3AHMAVQBwAGQAYQB0AGUAIAAtAEEAYwBjAGUAcAB0AEEAbABsACAALQBBAHUAdABvAFIAZQBiAG8AbwB0AA==
```

PowerShell commands are typically Base64 encoded and can be decoded using tools like CyberChef, which is the tool I'll be using. Since the command is Base64 encoded, I applied two recipes from the left pane: **FromBase64** and **Decode Text**. I configured the **Decode Text** recipe to use UTF-16LE (1200), as this is the encoding PowerShell uses for Base64.



It looks like Someone had stepped in to help us strengthen our defenses, but who could it be?

As I dug deeper into uncovering the identity of our mysterious hero, I made a startling discovery. The credentials for the scripts managing Windows updates on the machines were outdated. Someone had brute-forced the systems, logged in successfully, and then fixed the credentials. This became clear when I noticed that each executed PowerShell command was preceded by a successful login from the same source IP—a source that had been causing failed login attempts over the past few days. But the biggest surprise? It was Glitch who accessed ADM-01 and fixed the credentials, a fact I confirmed after identifying the owner of the IP address.

Conclusion

In this log analysis project, I had the opportunity to work alongside Wareville's Security Operations Center (SOC) team to investigate alerts and distinguish between true positives and false positives. By leveraging tools like Elastic SIEM, I analyzed encoded PowerShell commands, identified brute force attempts, and uncovered suspicious login behaviors. My work involved correlating events, decoding malicious commands, and identifying critical patterns, ensuring the security of the town's digital infrastructure during a high-risk period.

Through this experience, I honed my skills in log analysis, event correlation, and the use of advanced SIEM tools, gaining a deep understanding of cybersecurity operations and incident

response. This project has prepared me for roles in log analysis and SOC environments, showcasing my ability to identify, analyze, and mitigate threats effectively.

Abdullahi-cybersecurity-portfolio