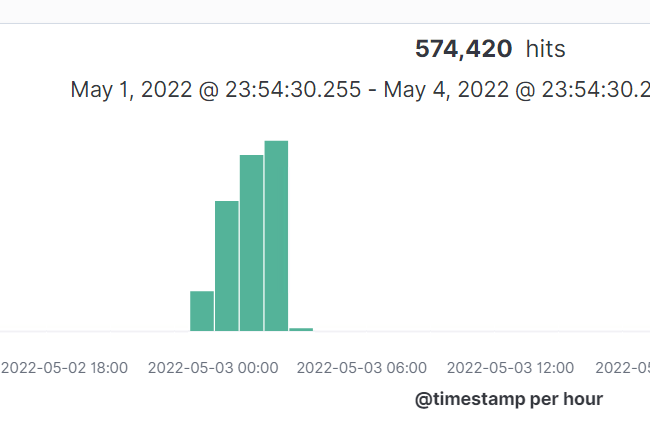
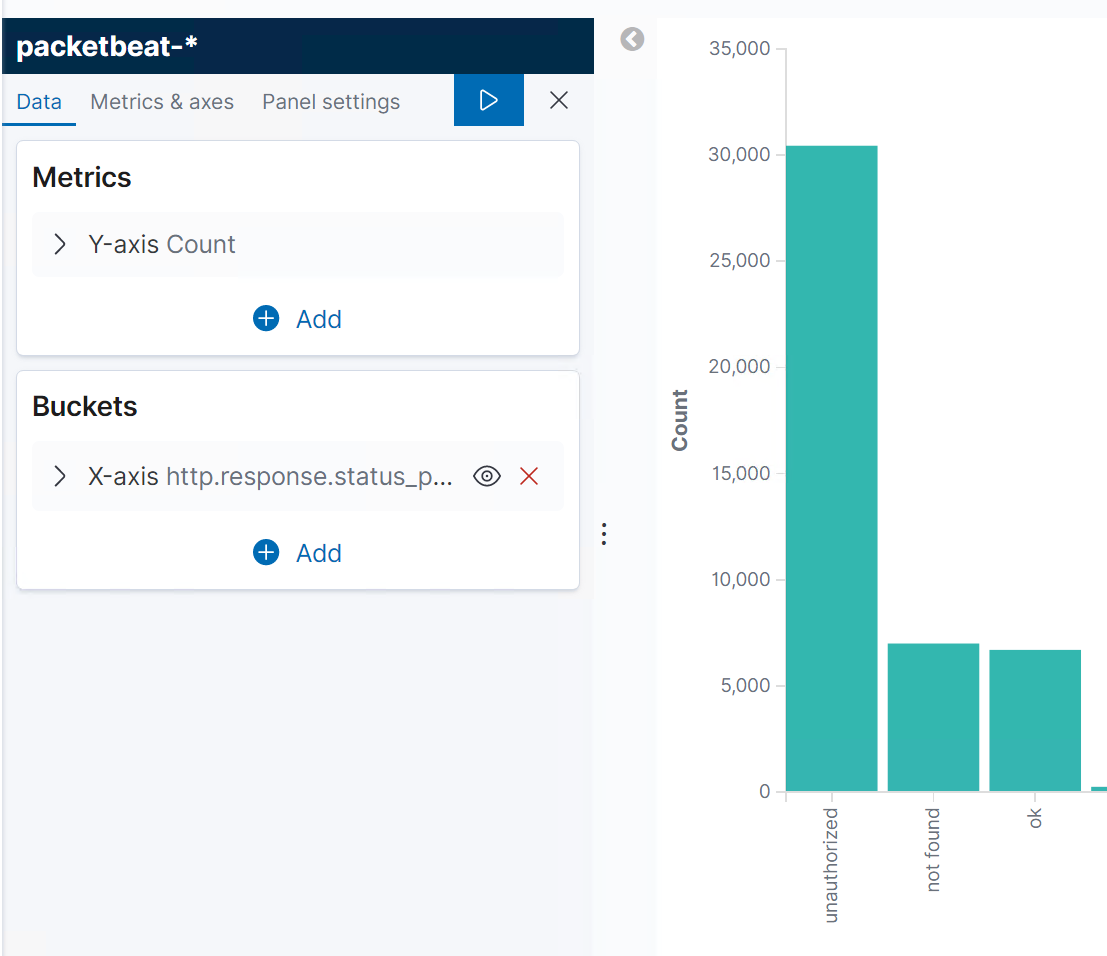
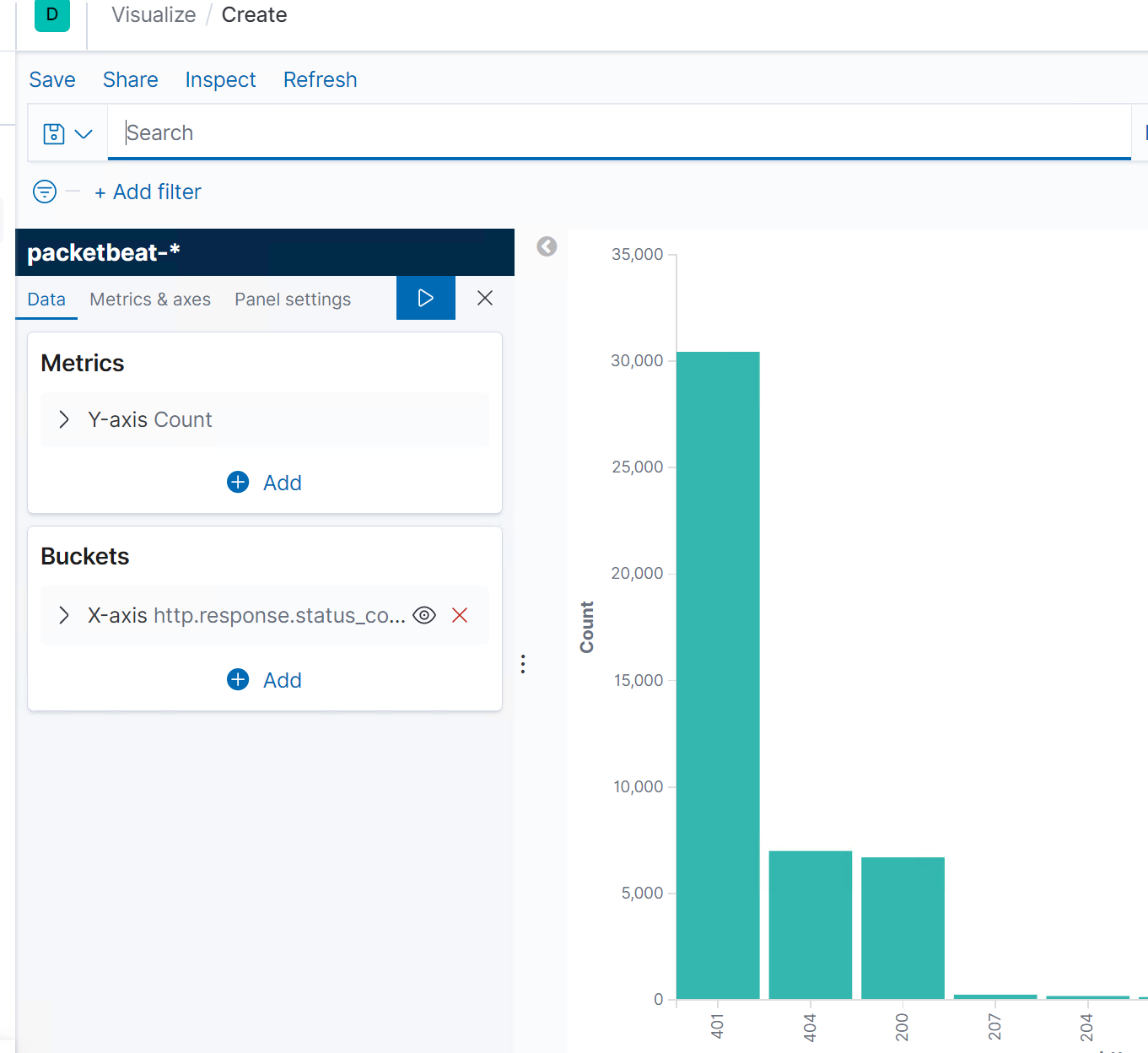
**Week 20 - Activity 2 - Incident Analysis with Kibana**

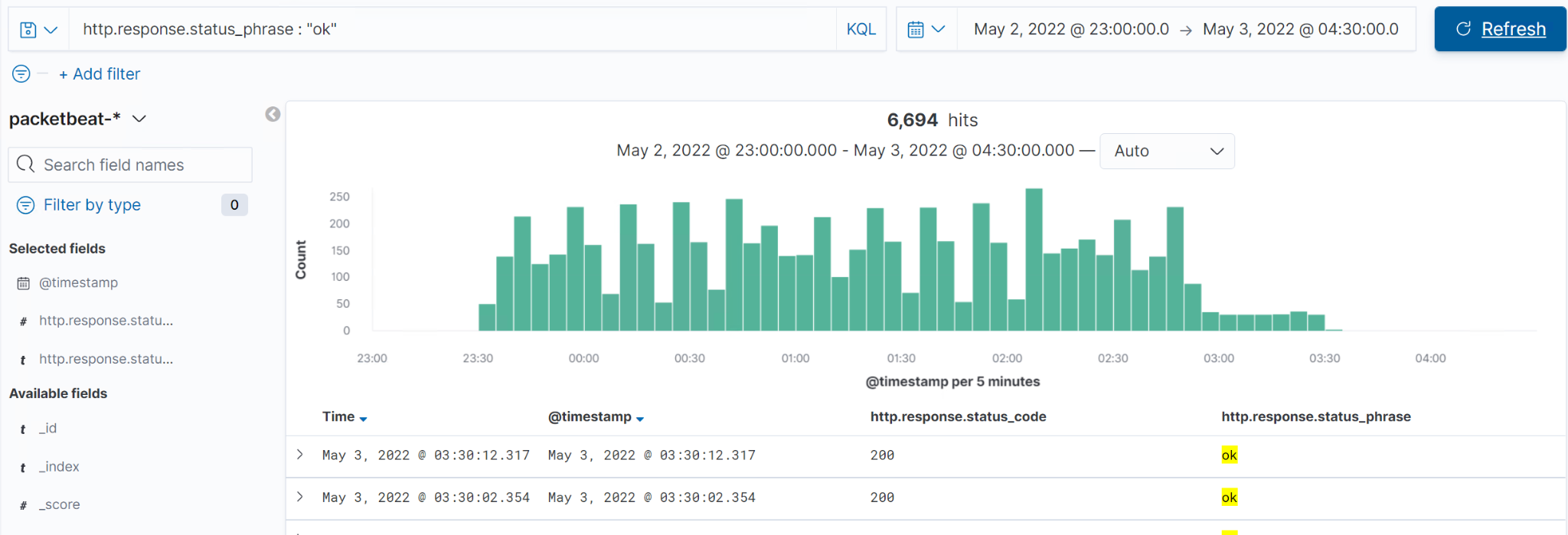
1. Identify the offensive traffic.
   1. Identify the traffic between your machine and the web machine:
      1. When did the interaction occur?
         1. Using Discover -Packetbeat - it clearly shows the attack started 2300 on 2 May, and effectively ended 0300 on May 3.



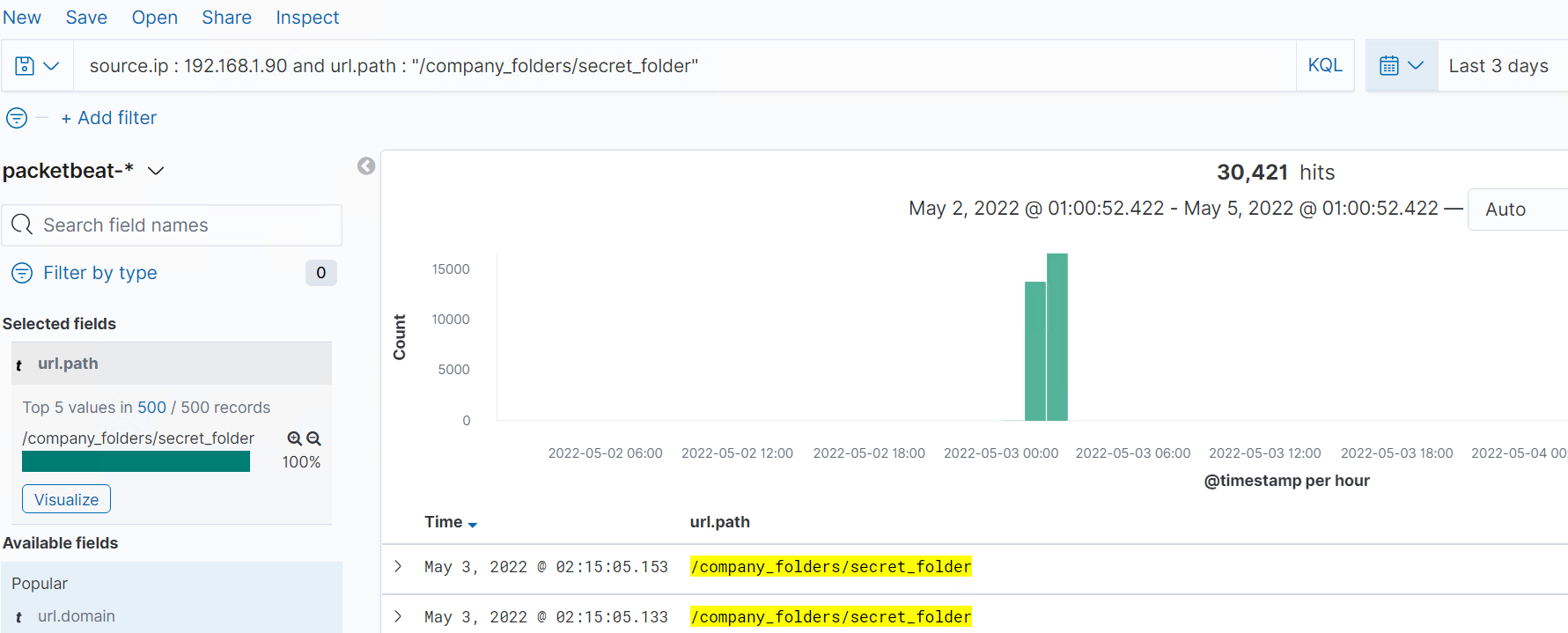
* + 1. What responses did the victim send back?
       1. Using both fields of http **http.response.status\_code & http.response.status\_phrase** we saw that there was a majority of unauthorized responses being sent back

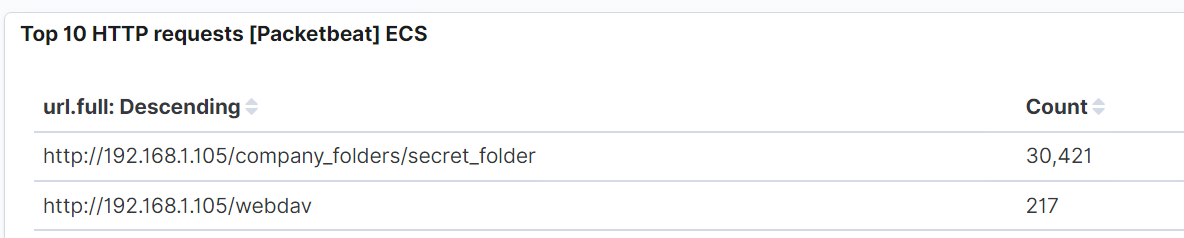


* + 1. What data is concerning from the Blue Team perspective?
       1. In short, there were a small but significant number of ok (successful logins)( 200) vs the unauthorized one (401). This indicates that the attacker was able to target the “/webdav” URL, which contains secret documents (and the reverse shell exploit), and was able to, at times, successfully access the webpage.

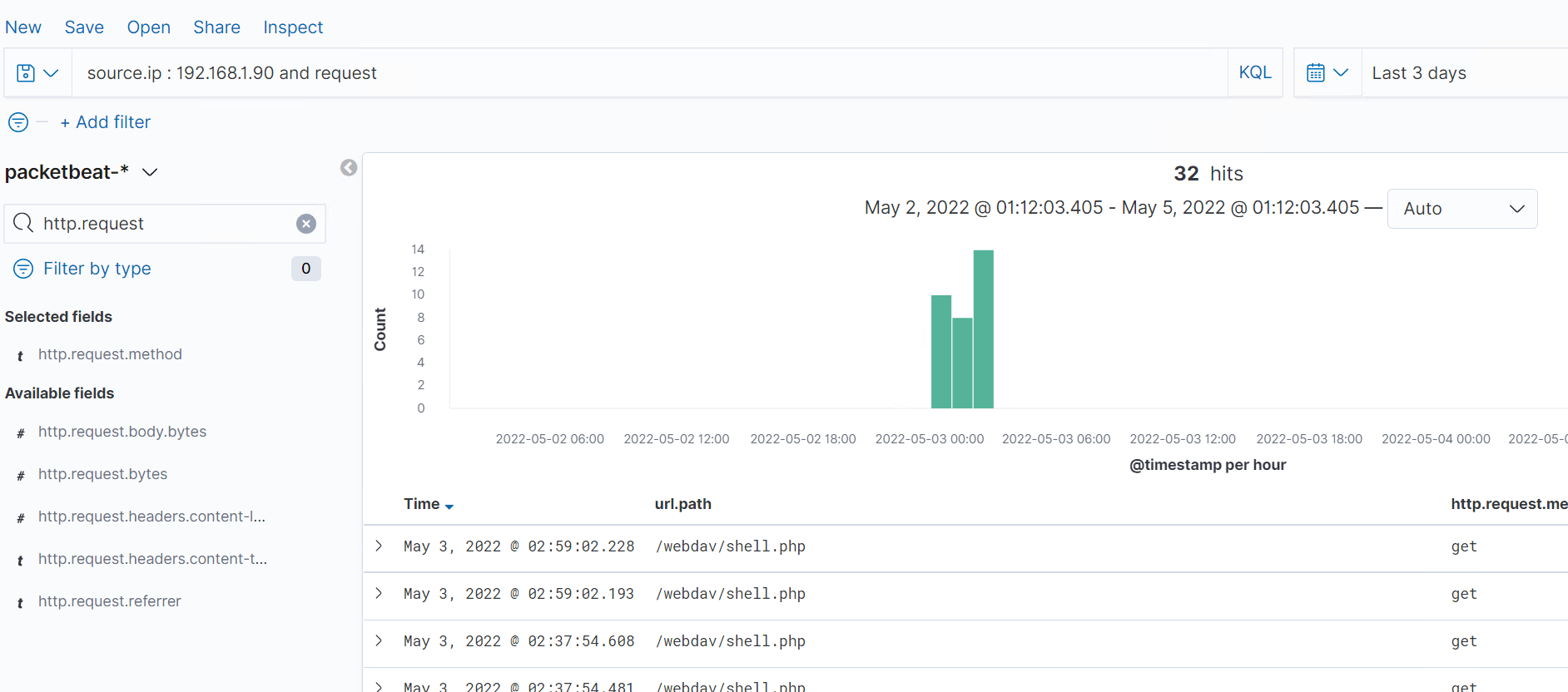


1. Find the request for the hidden directory.
   1. In your attack, you found a secret folder. Let's look at that interaction between these two machines.
      1. How many requests were made to this directory? At what time and from which IP address(es)?
         1. 30,421 from IP address 192.168.1.90 from 0100-0200 on May 3.



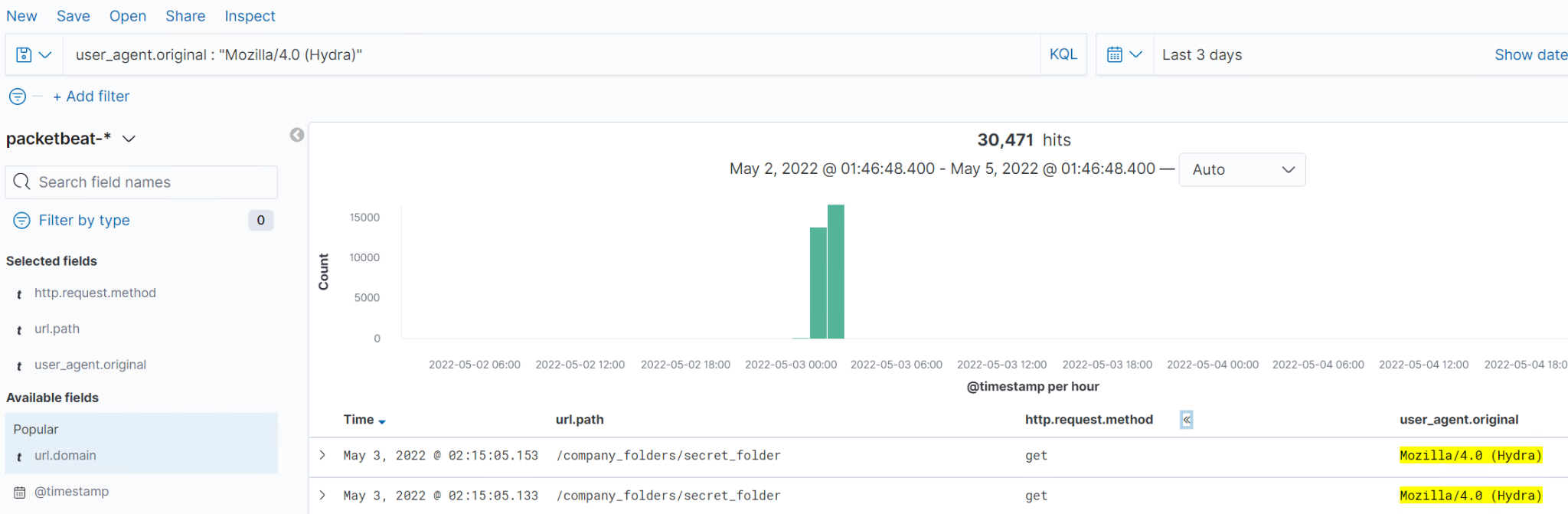


* + 1. Which files were requested? What information did they contain?
       1. Using the search - source.ip : 192.168.1.90 and request - it was the shell.php file they were after. That php file contained a reverse TCP exploit.

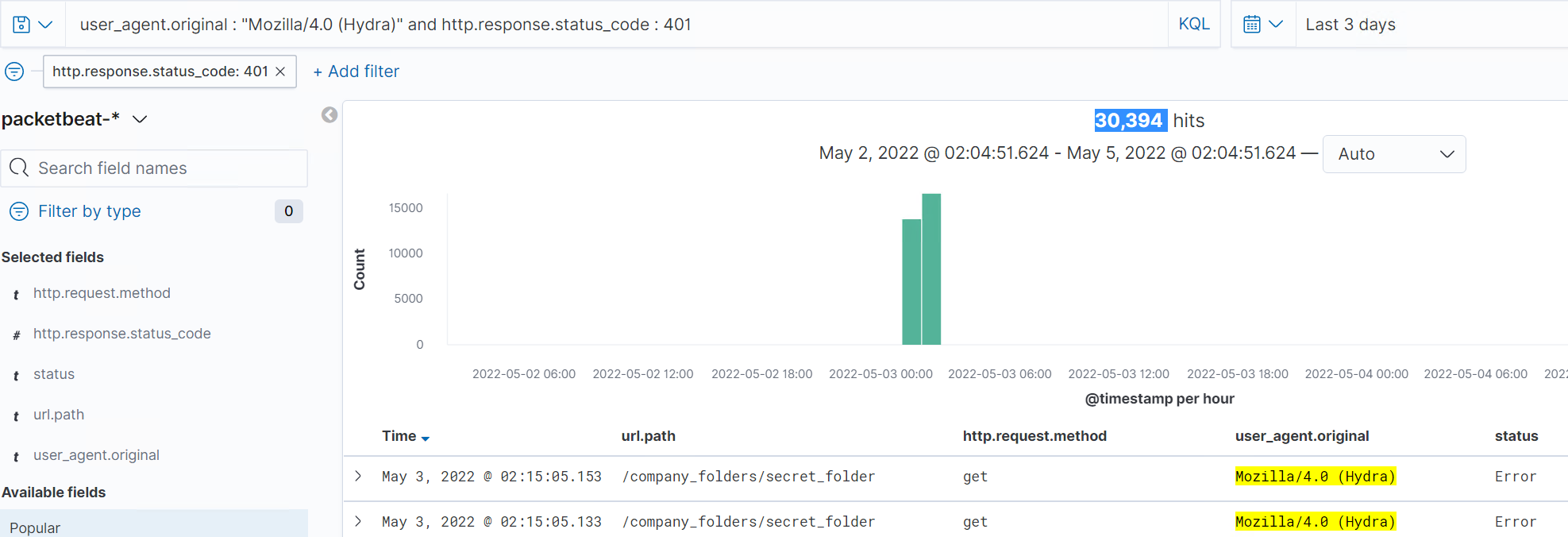


* + 1. What kind of alarm would you set to detect this behavior in the future?
       1. It would be beneficial to set an alert to monitor surging unauthorized requests. This alert should send notifications once a threshold of unsuccessful requests or logins occurs.
    2. Identify at least one way to harden the vulnerable machine that would mitigate this attack.
       1. Account Lockouts – lock out an account after x number of unsuccessful attempts. I would suggest that we put that number at 5 and after that the user would have to call some sort of support number to verify themselves and get the account unlocked again
       2. Login Captcha – this would easy stop a botnet, and or brute force attack for the most part as this would require human interaction to correctly answer the right question to continue logging on
       3. Block traffic to port 80 – which was the port that the attacker accessed and carried out the attack with.
       4. Enable / Force 2FA – Basically force all your users to use their phones as a means of getting into their account as the second half of 2FA
       5. Stringent Password Requirements – force at least a 12-character password, with at least one upper case, one number, one special character, be unable to use the last 10 passwords (previous use history)

1. Identify the brute force attack.
   1. After identifying the hidden directory, you used Hydra to brute-force the target server. Answer the following questions:
      1. Can you identify packets specifically from Hydra?
         1. user\_agent.original:”Mozilla/4.0 (Hydra)”

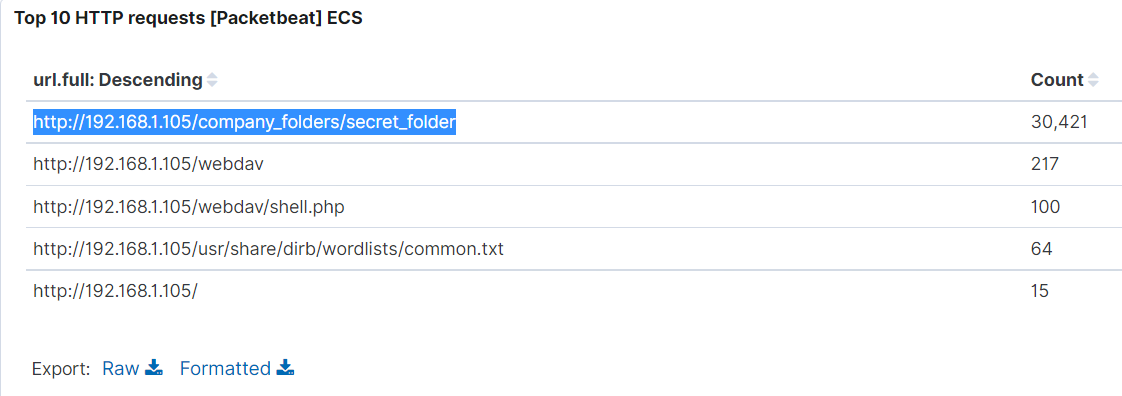


* + 1. How many requests were made in the brute-force attack?
       1. This number depends on how many times you ran the hydra attack on your Kali VM. From the graph we see 30,394.



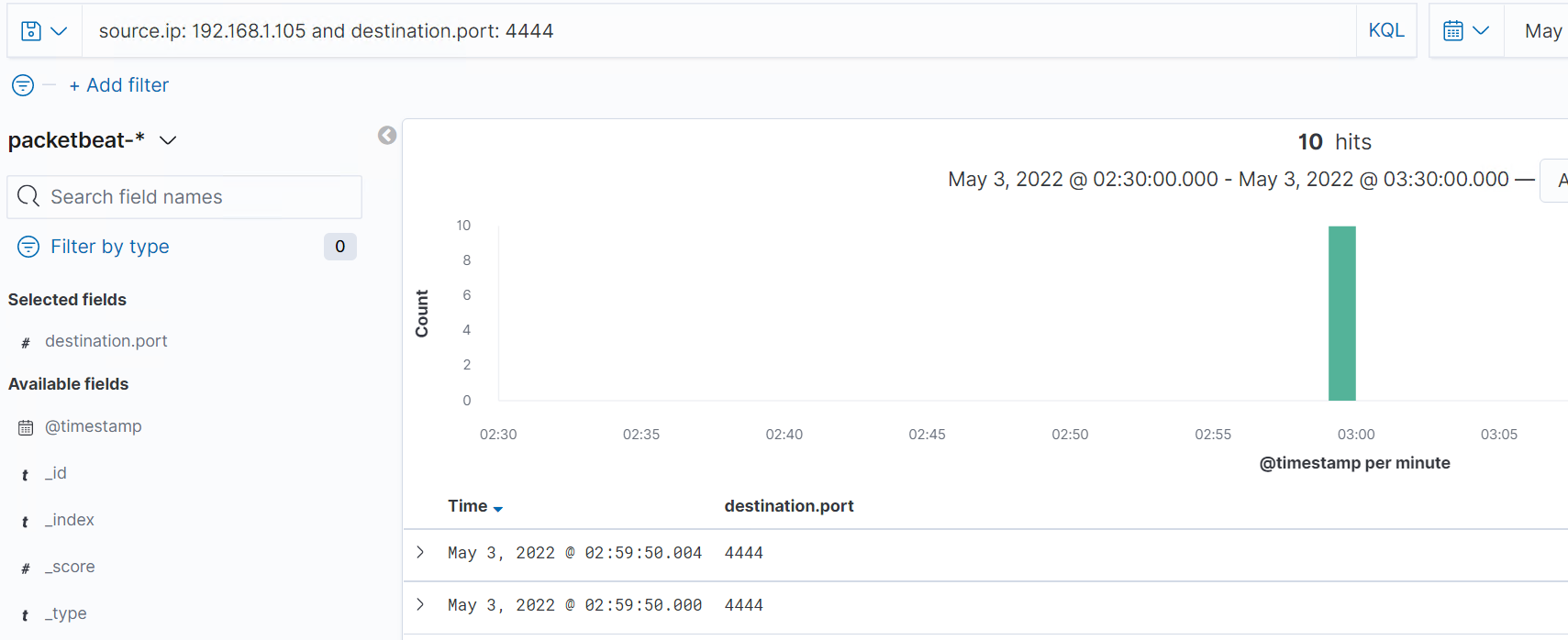
* + 1. How many requests had the attacker made before discovering the correct password in this one?
       1. 30394
    2. What kind of alarm would you set to detect this behavior in the future and at what threshold(s)?
       1. I would recommend setting an alert based on a specific threshold for the number of HTTP GET requests, in addition to setting an alert when the user\_agent.original is equal to or includes the term “Hydra.”
    3. Identify at least one way to harden the vulnerable machine that would mitigate this attack.
       1. Account Lockouts – lock out an account after x number of unsuccessful attempts. I would suggest that we put that number at 5 and after that the user would have to call some sort of support number to verify themselves and get the account unlocked again

1. Find the WebDav connection.
   1. Use your dashboard to answer the following questions:



* + 1. How many requests were made to this directory?
       1. 217
    2. Which file(s) were requested?
       1. shell.php
    3. What kind of alarm would you set to detect such access in the future?
       1. I would recommend setting an alert anytime this directory is accessed by a machine otherthan the machine that should have access.
    4. Identify at least one way to harden the vulnerable machine that would mitigate this attack.
       1. Connections to this shared folder should not be accessible from the web interface.
       2. Connections to this shared folder could be restricted by machine with a firewall rule.
       3. Ensure the firewall detects and cuts off the scan attempt in real time.
       4. Ensure the firewall is regularly patched to minimize new zero-day attacks

1. Identify the reverse shell and meterpreter traffic.
   1. To finish off the attack, you uploaded a PHP reverse shell and started a meterpreter shell session. Answer the following questions:
      1. Can you identify traffic from the meterpreter session?
         1. source.ip: 192.168.1.105 and destination.port: 4444



* + 1. What kinds of alarms would you set to detect this behavior in the future?
       1. I would recommend setting an alert that is set off any time traffic moves over port 4444 (or any open port). Additionally, I would recommend setting an alert that is triggered any time a file with the extension of .php is uploaded to the server.
    2. Identify at least one way to harden the vulnerable machine that would mitigate this attack.

1. Creating a whitelist of trusted IP addresses and ensure my firewall security policy prevents all other access
2. Ensure that any access to the WebDAV folder is only permitted by users with complex username and passwords.
3. Limiting the ability to upload files via the file manager/web interface to this specific directory
4. Ensure only necessary ports are open