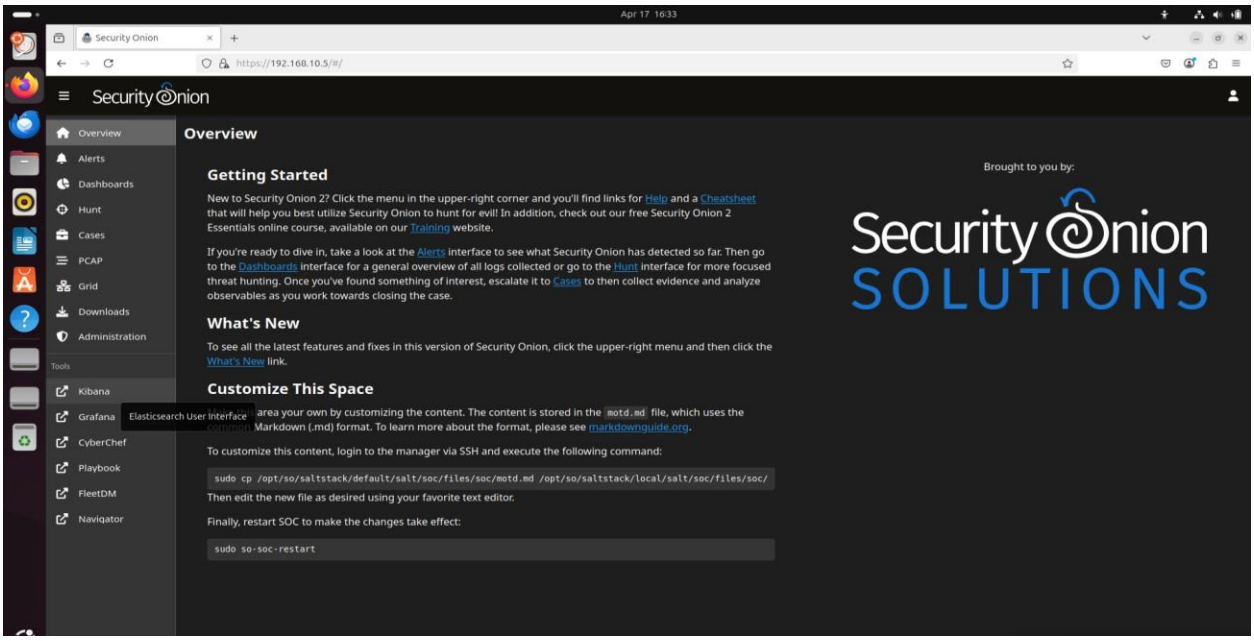
BFOR 650 – LAB REPORT 8 – Creating a Breach Detection Dashboard in Kibana

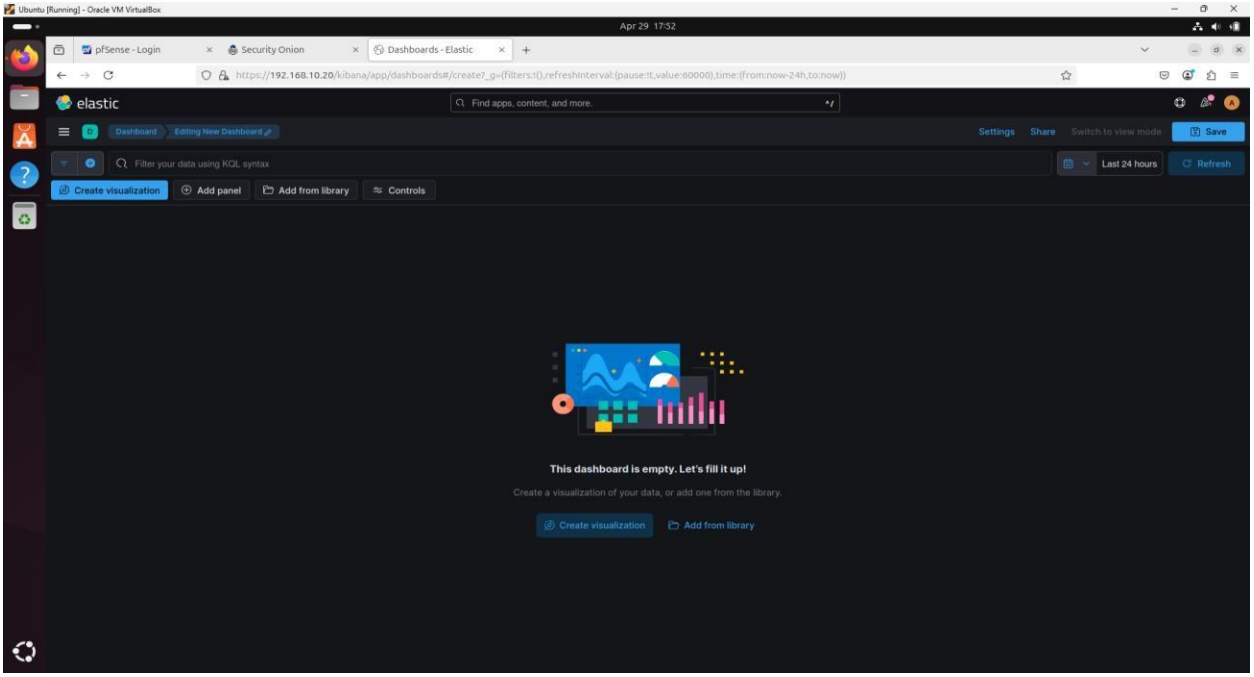
NAME: Phanindhar Reddy Karnati
ID: 001667635

8.1: NIDS Alerts

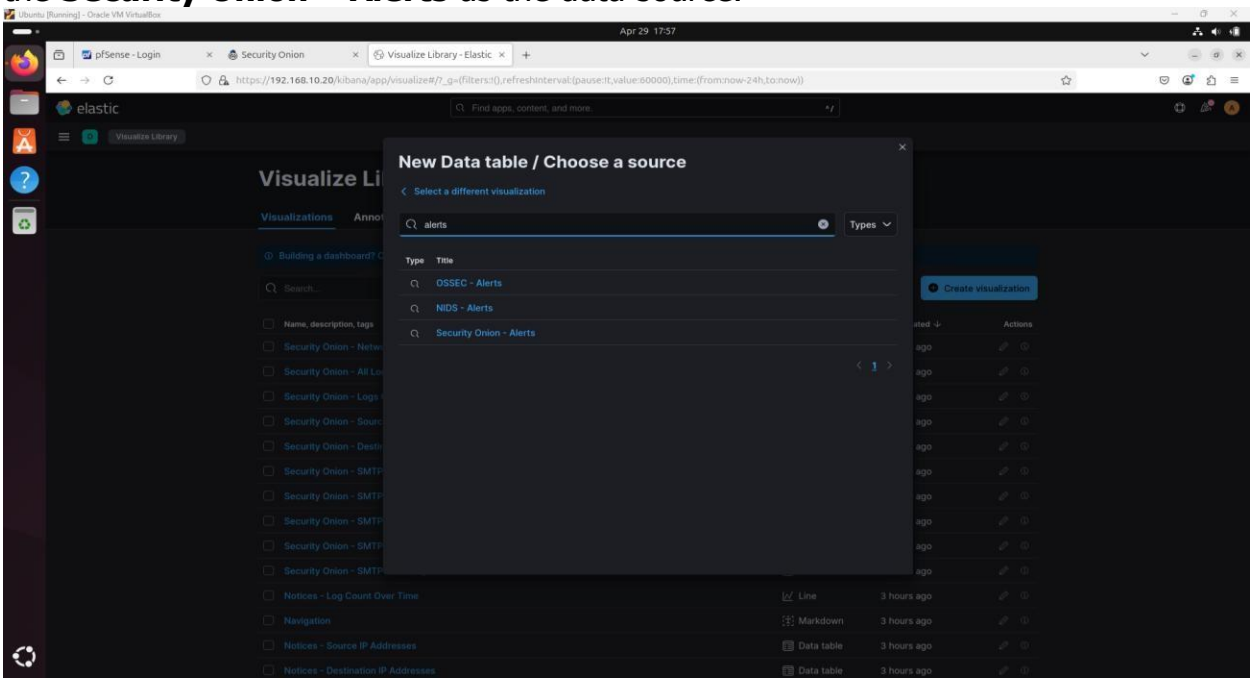
1. I logged into Security Onion through the Ubuntu VM and opened the Kibana interface using the left-hand panel.



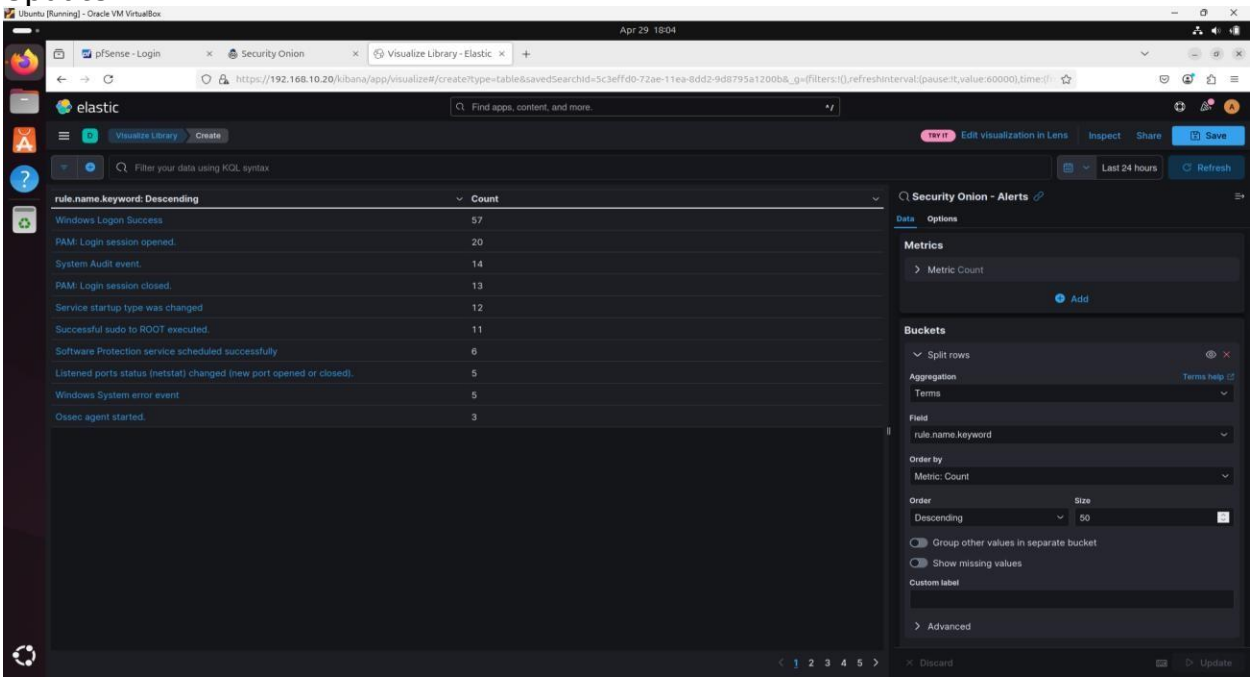
2. I navigated to the Dashboards section and clicked on Create Dashboard, which opened a blank dashboard editor.



3. I chose to create a new **Data Table** visualization, searched for "Alerts," and selected the **Security Onion – Alerts** as the data source.



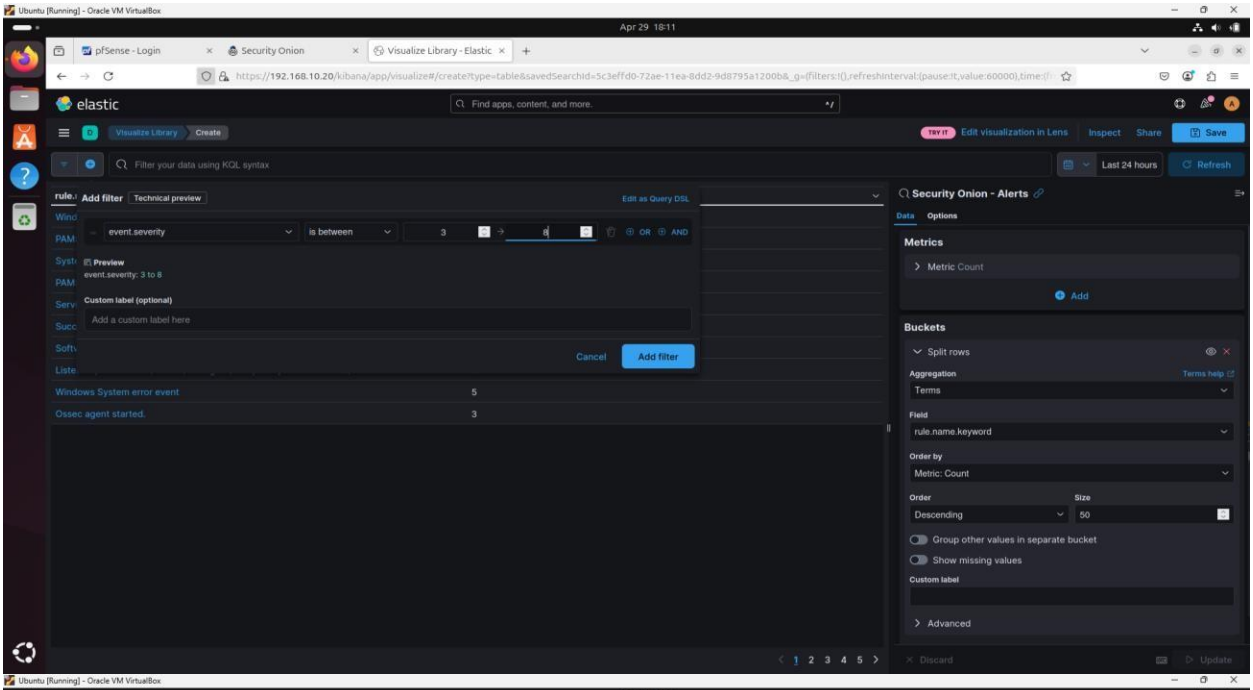
4. In the data panel, I added a bucket by choosing Split Rows, selected Terms for aggregation, and used rule.name.keyword as the field. I set the size to 50 and clicked Update.

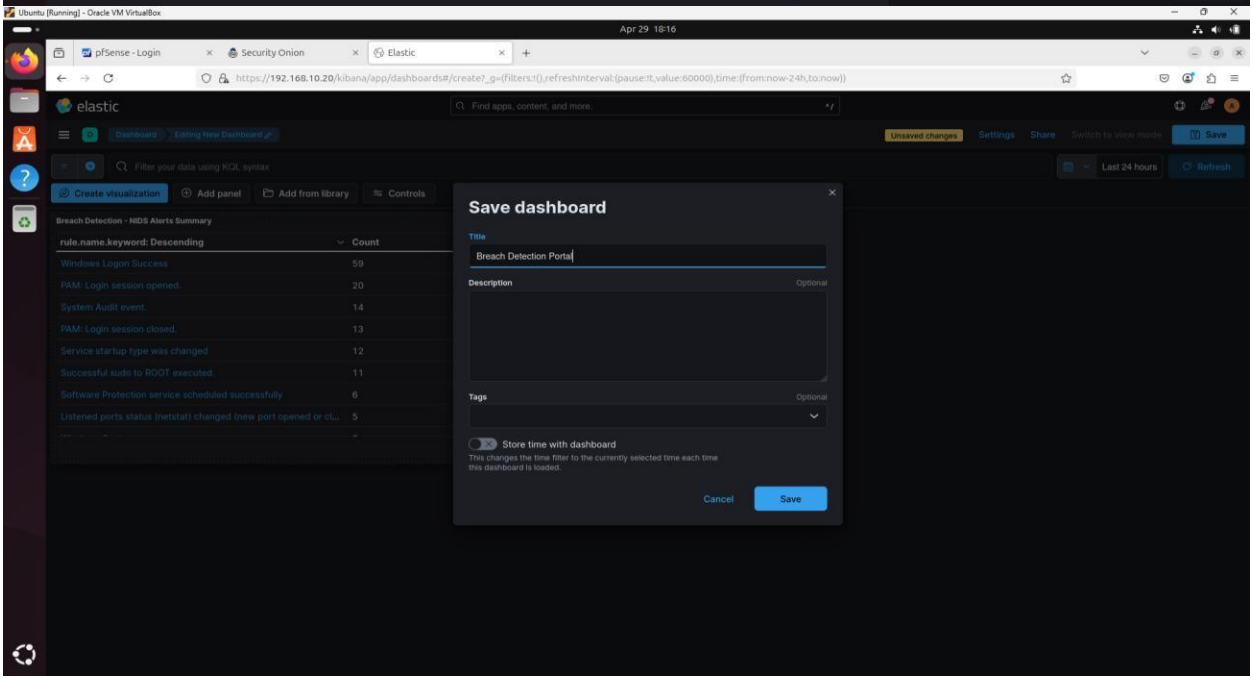


5. To filter by severity, I clicked Add Filter, set the field to event.severity, selected is between, and entered values 3 and 8.

6. After this, the data table provided a summary of frequent NIDS alerts. I saved it as Breach Detection – NIDS Alerts Summary and returned to the dashboard.

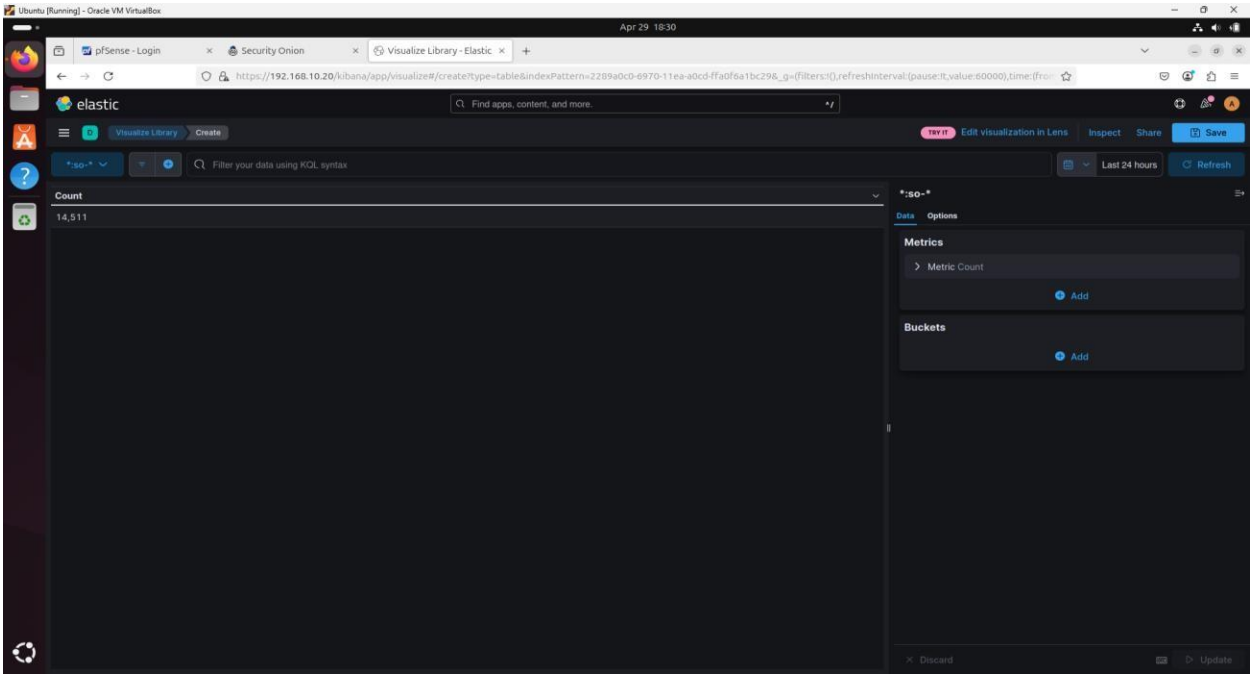
7. I saved the full dashboard as Breach Detection Portal using the Save button at the top.



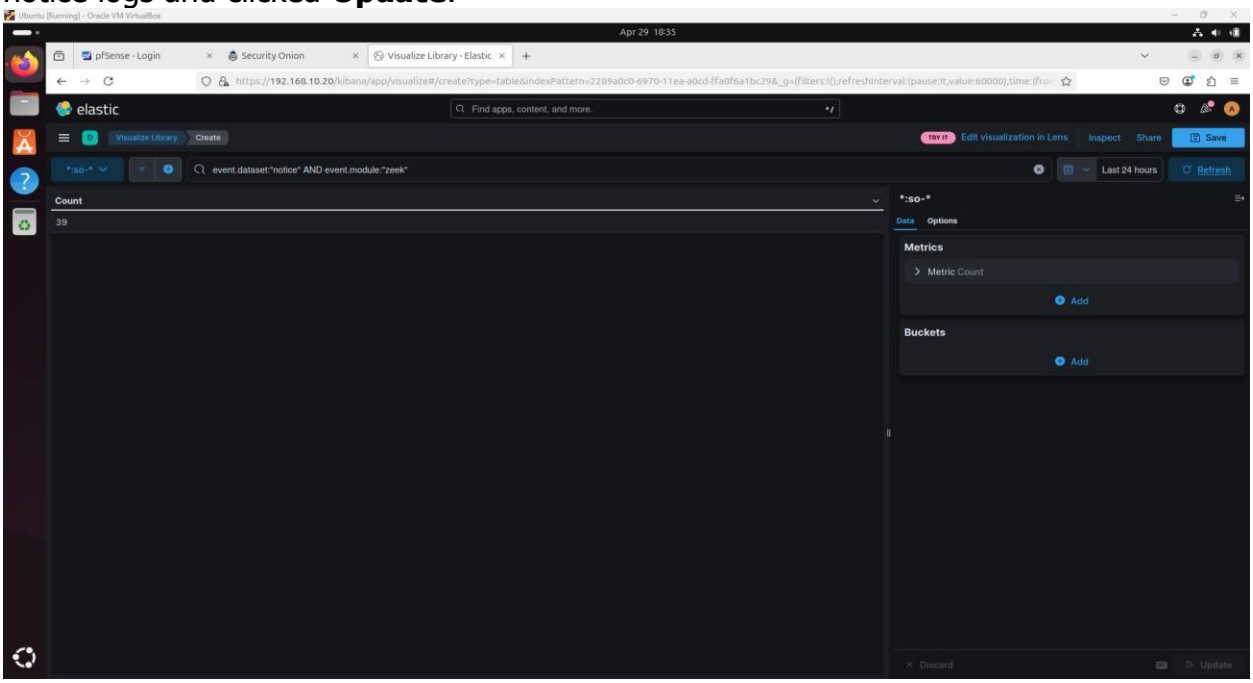


8.2: Zeek Notices

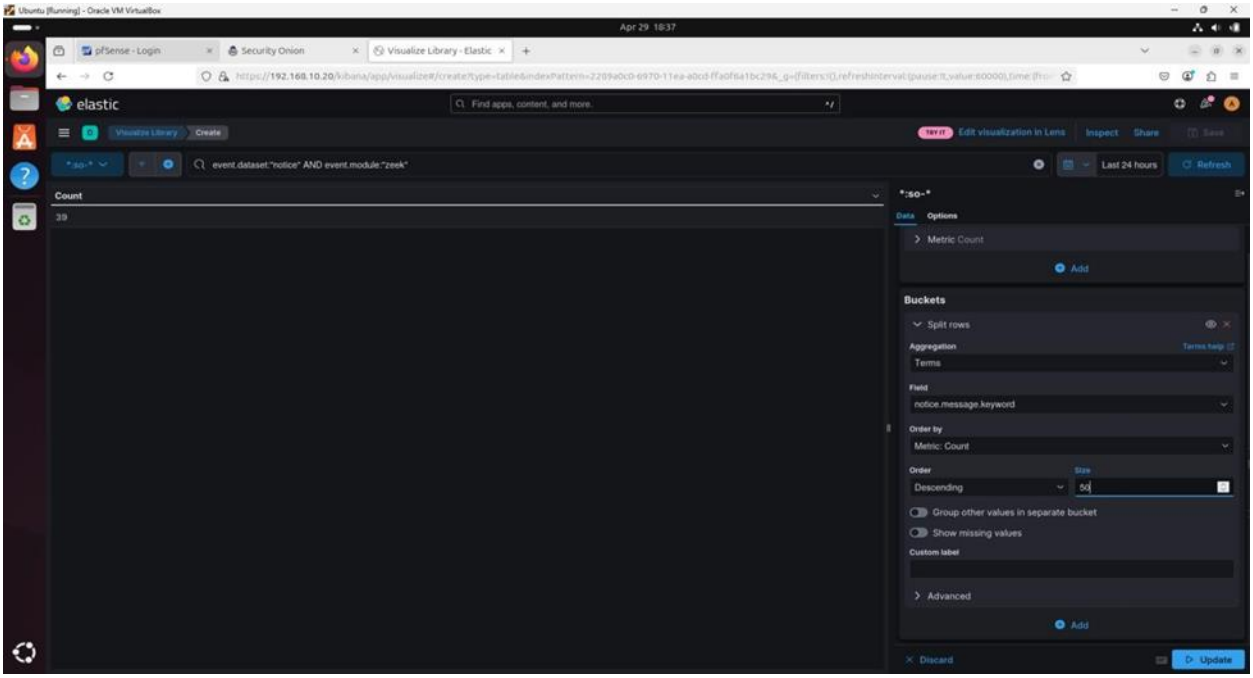
1. I clicked **Create Visualization** from inside the Breach Detection Portal and chose a **Data Table** type. I used the *.so-* data source.



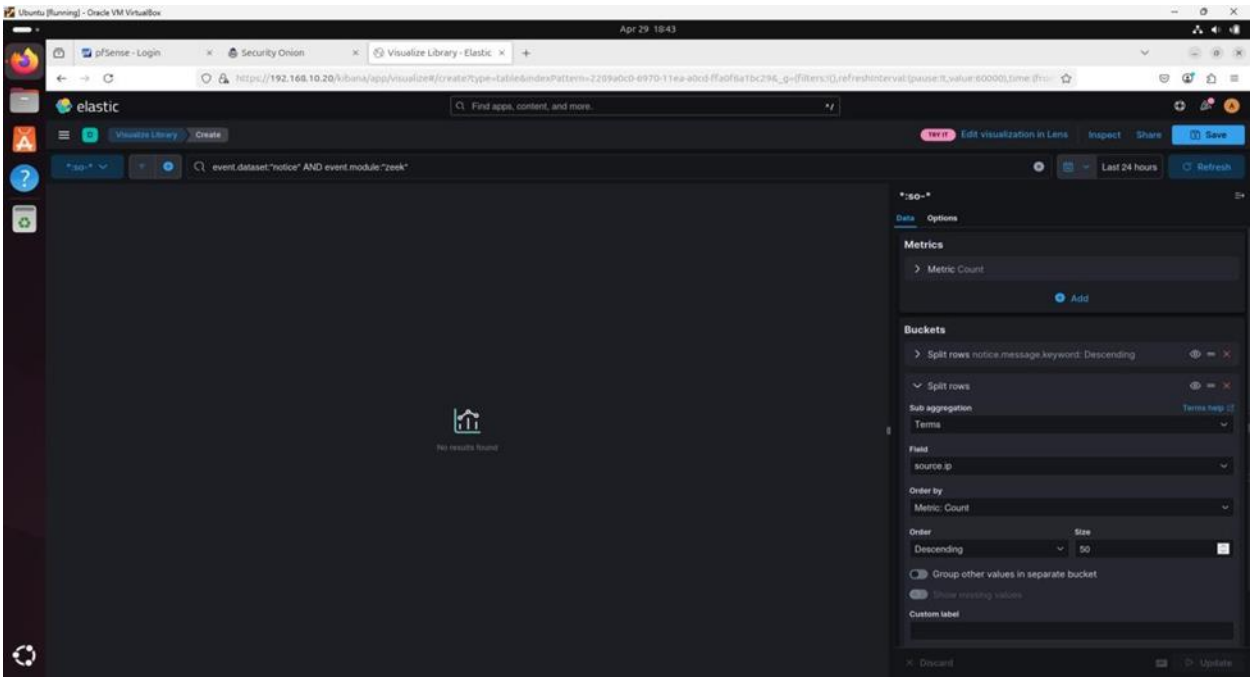
2. In the search bar, I typed `event.dataset:notice AND event.module:zeek` to focus on Zeek notice logs and clicked **Update**.



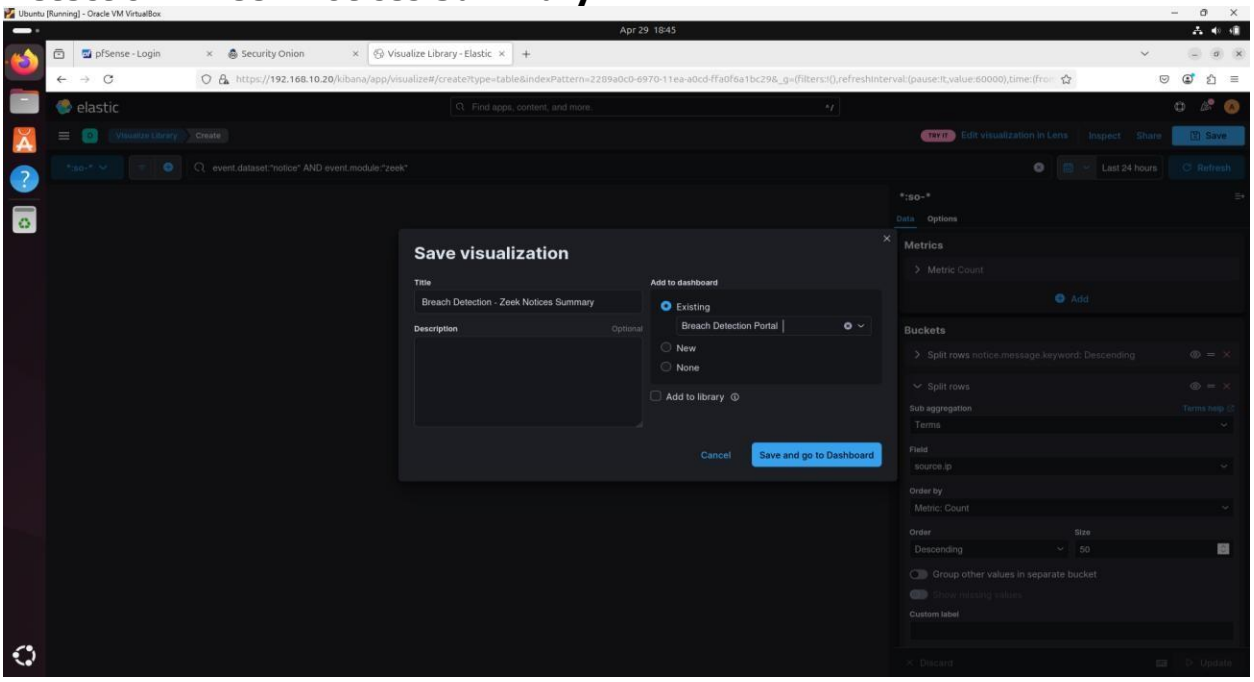
3. I added a **Split Rows** bucket, chose **Terms** for aggregation, and set the field to `notice.message.keyword`, with size 50.



4. I then added another Split Rows bucket for source.ip to pair source IPs with their notice messages.

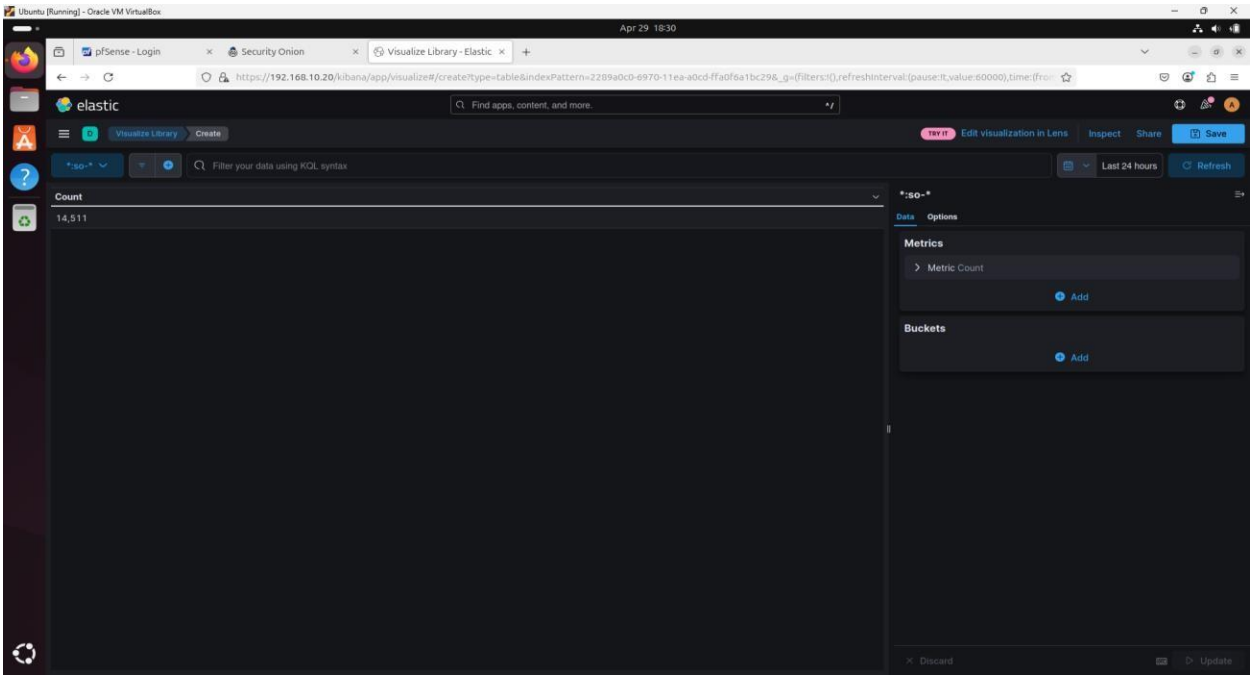


5. Since there were no Zeek notice logs visible, I saved the widget anyway as **Breach Detection – Zeek Notices Summary**.

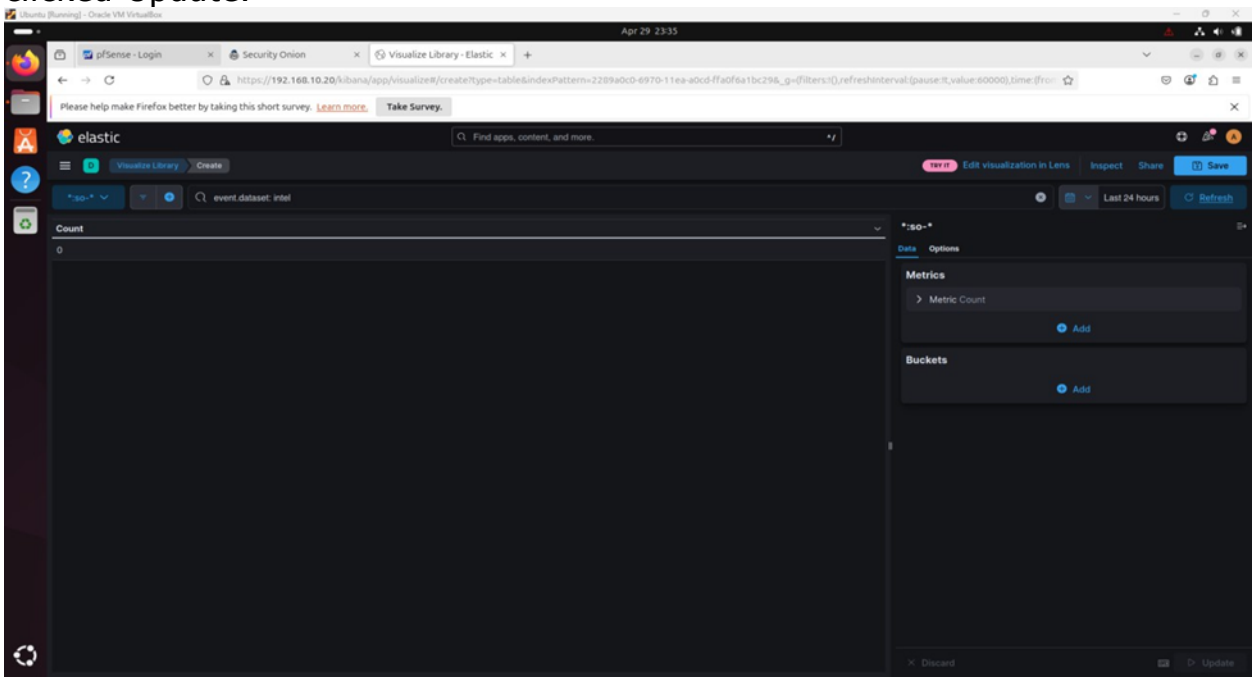


8.3: Zeek Intel Logs

1. I created another Data Table visualization and selected the *.so-* data source.



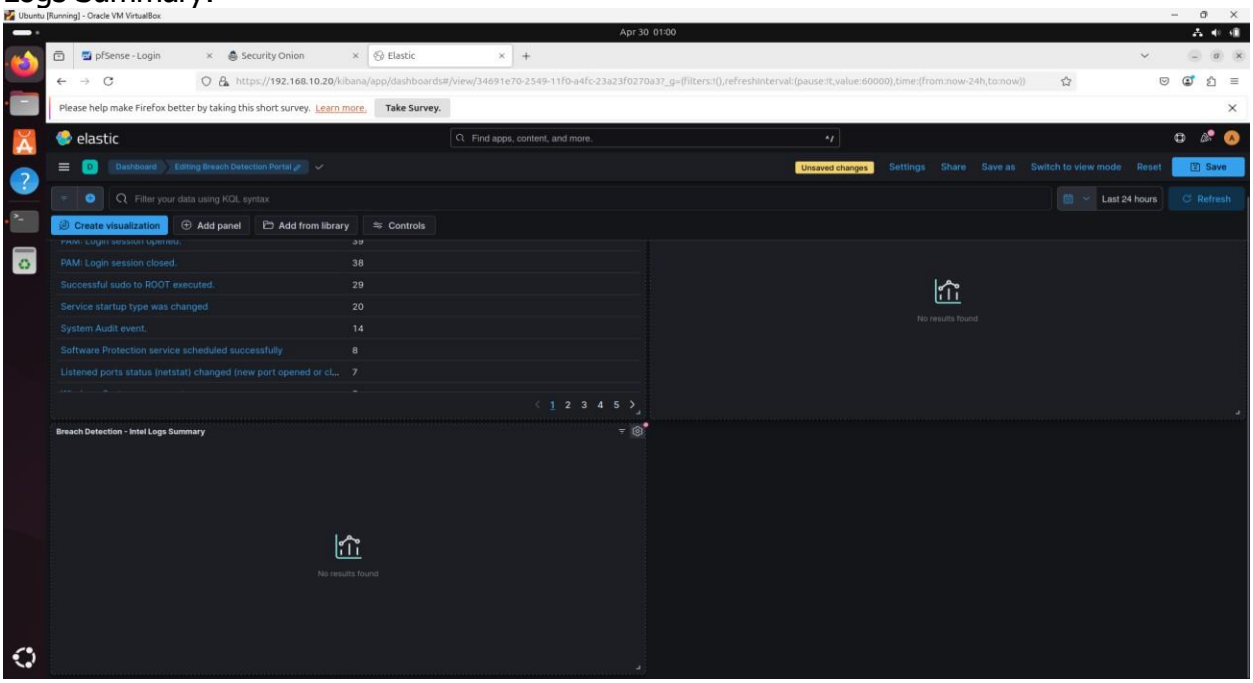
2. I entered event.dataset:intel in the filter bar to isolate intel-related logs and clicked Update.



3. I added a Split Rows bucket, set aggregation to Terms, and chose intel.sources.keyword as the field with size 50.

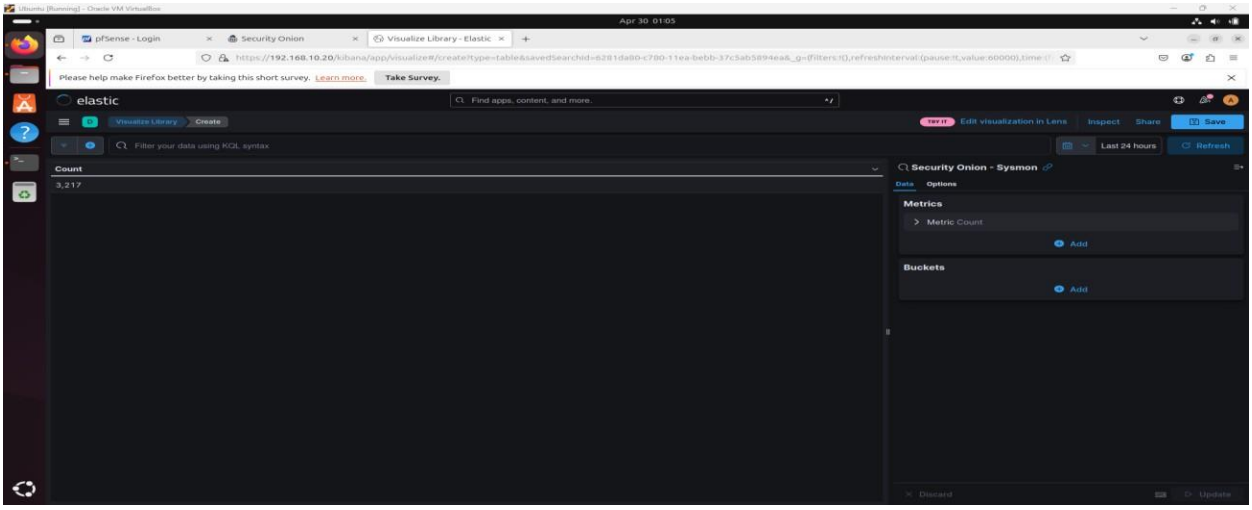
4. I added another Split Rows for source.ip to display the IPs alongside intel alerts.

5. There were no active Zeek Intel logs, so I saved the visualization as Breach Detection – Intel Logs Summary.

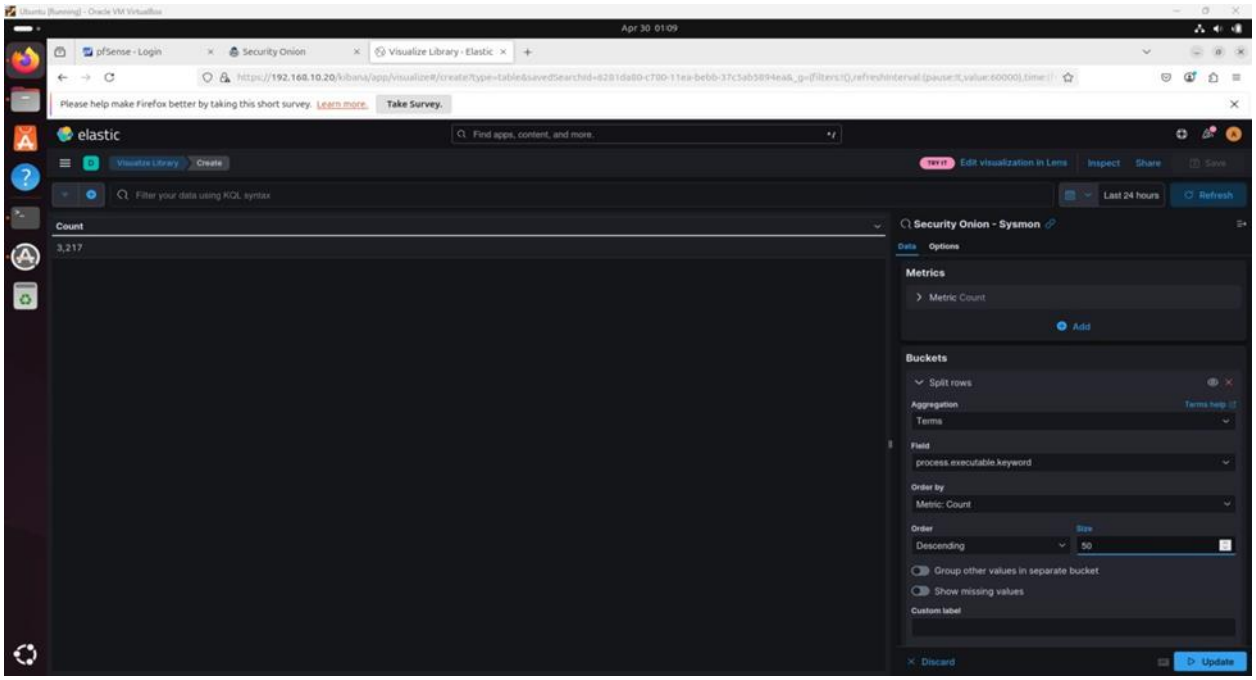


8.4: Suspicious Process and File Creation

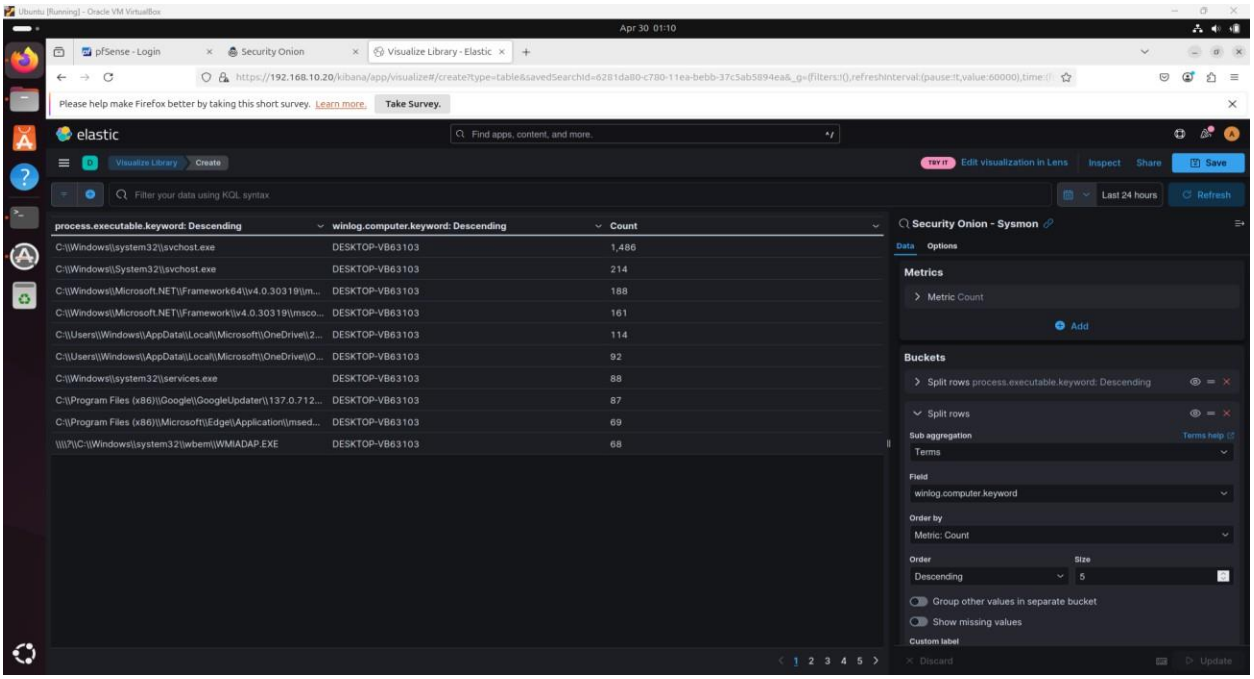
1. I created a new Data Table using the Security Onion Sysmon data source.



2. I added a Split Rows bucket using process.executable.keyword, set aggregation to Terms, and limited the size to 50.



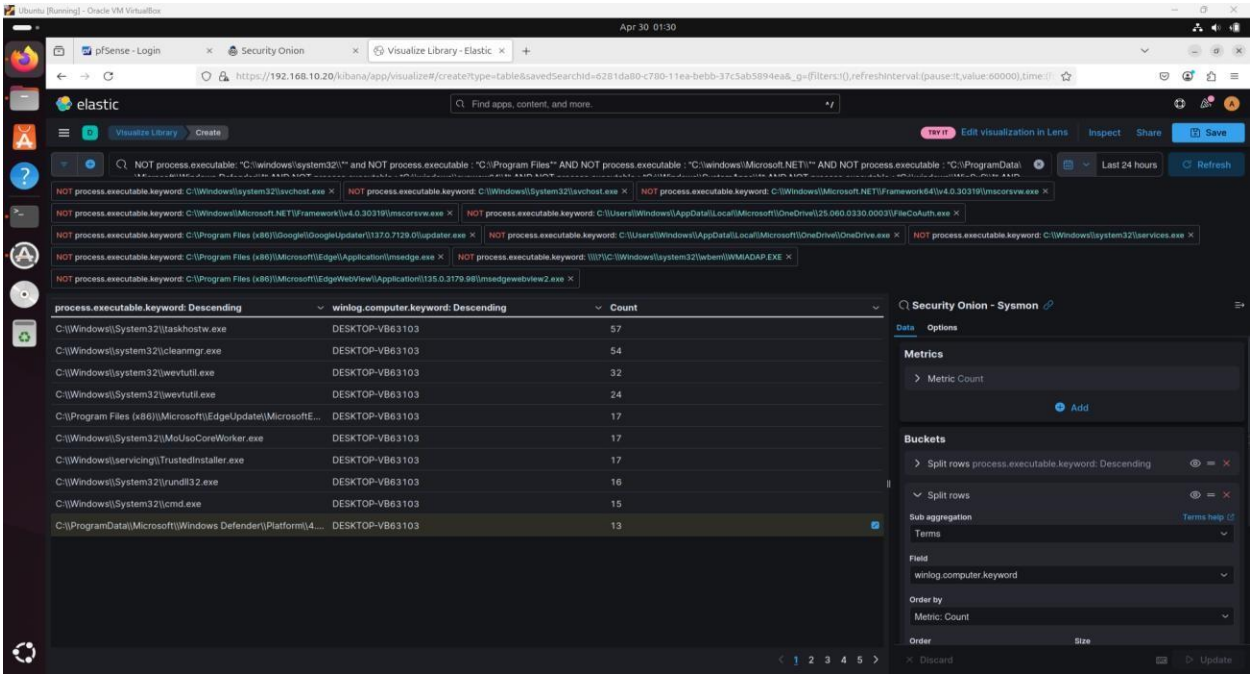
3. Then I added another split row using winlog.computer.keyword to display the computer associated with each process.



4. In the filter bar, I entered a complex `NOT` filter to exclude known safe executable paths from directories like System32, Program Files, and Windows Defender.

   NOT process.executable: "c:\\windows\\system32*" AND NOT process.executable: "c:\\Program Files*" AND NOT process.executable: "c:\\windows\\Microsoft.NET*" AND NOT process.executable: "c:\\ProgramData\\Microsoft\\Windows Defender*" AND NOT process.executable: "c:\\windows\\syswow64*" AND NOT process.executable: "c:\\Windows\\SystemApps*" AND NOT process.executable: "c:\\windows\\WinSxS*" AND NOT process.executable: "c:\\windows\\servicing*" AND NOT process.executable: "c:\\windows\\softwaredistribution*"

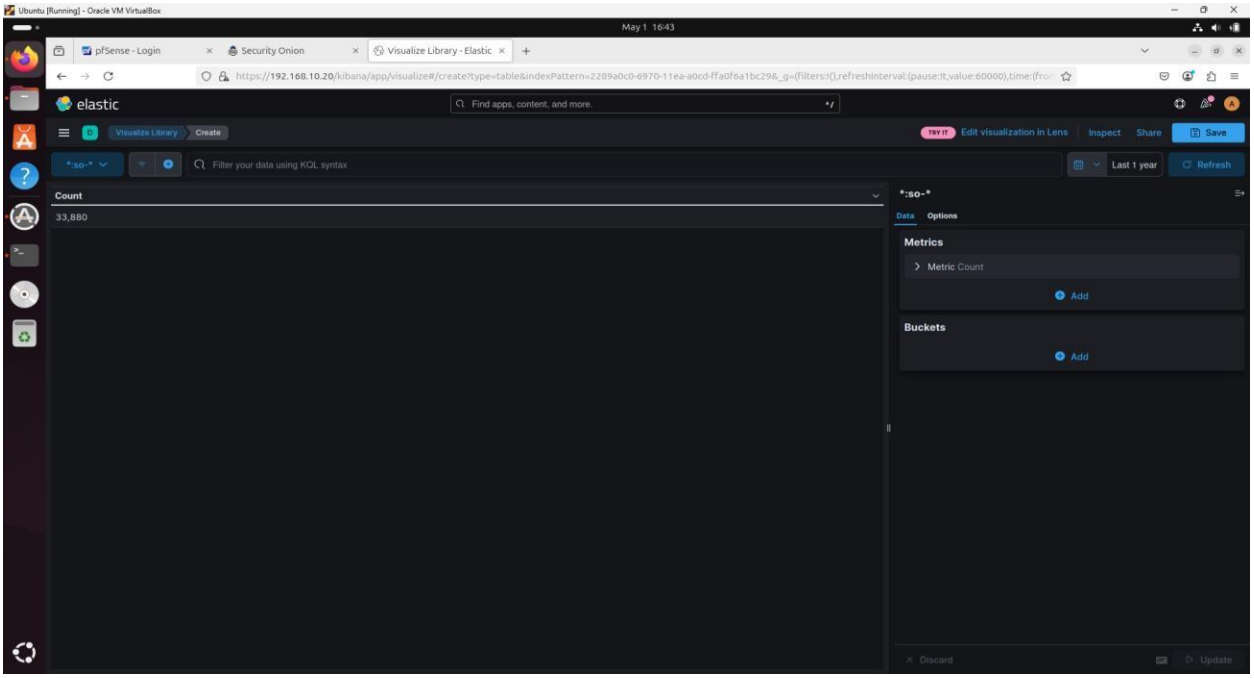
   Click on **Update** and manually eliminate all other safe executables and do this process over time to keep the safe executables filtered out.



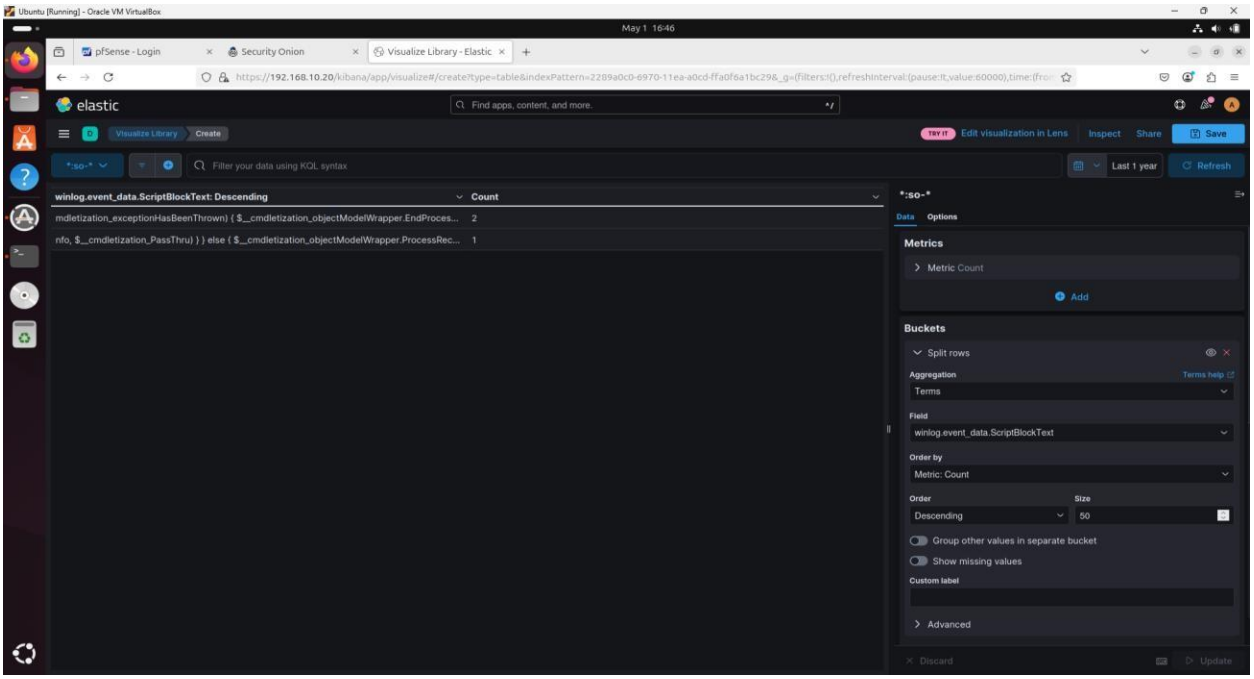
5. I saved this visualization as Breach Detection – Suspicious Image Paths.

8.5: Suspicious PowerShell Commands

1. I created a **Data Table** in the Breach Detection Portal using the *:so-* data source.

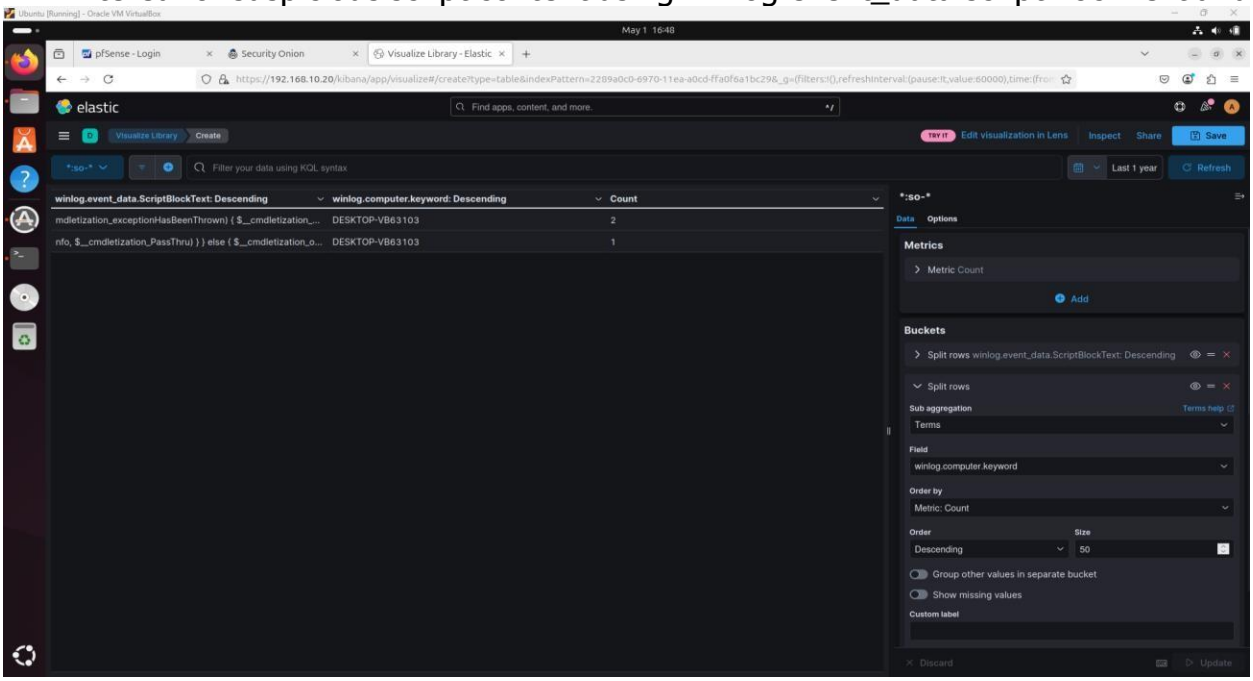


2. I added a Split Rows bucket with winlog.event_data.scriptBlockText.keyword and set the size to 50.



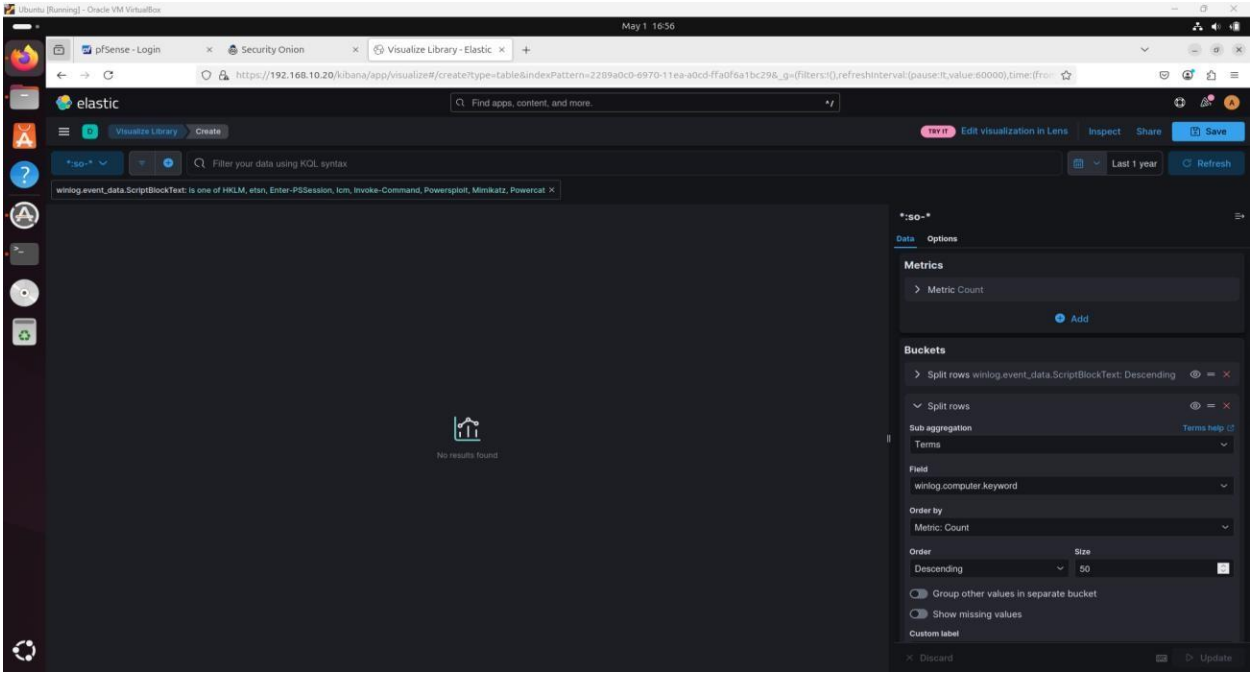
3. I added another Split Rows using winlog.computer.keyword to group the script logs by host.

4. I filtered for suspicious script content using winlog.event_data.scriptBlockText and

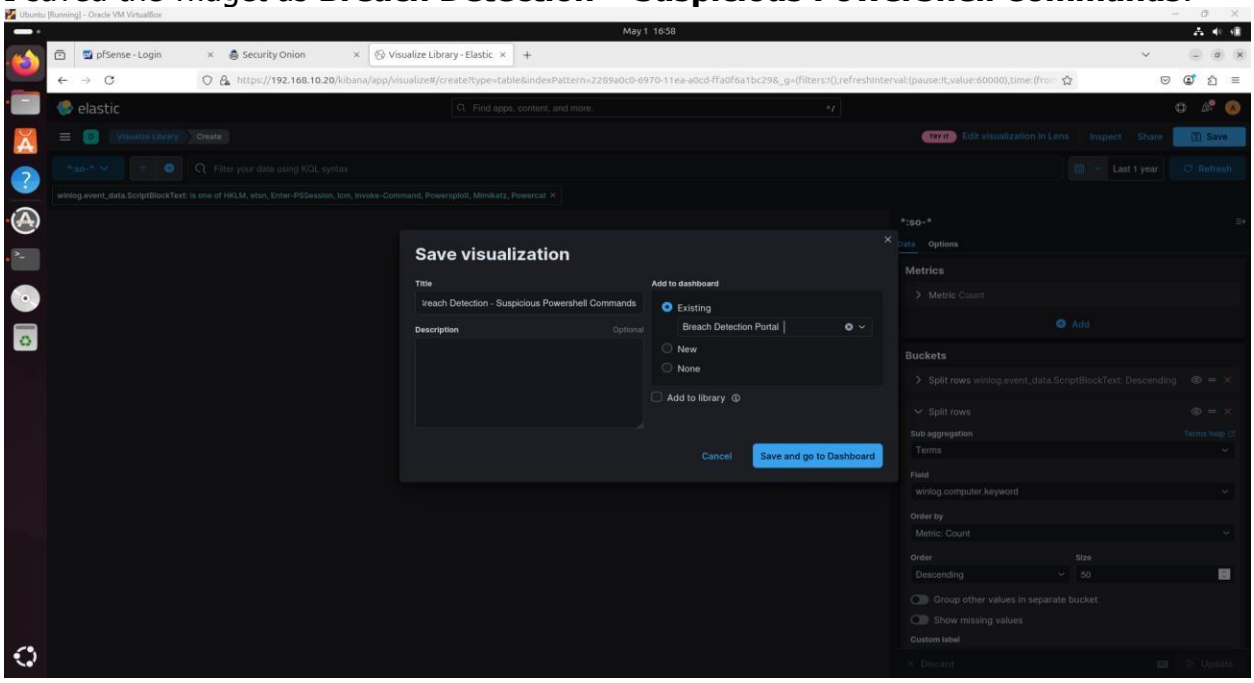


the operator is one of, then input known suspicious PowerShell command patterns.

```
HKLM             - for detecting interaction with the registry
etsn             - to detect remote access attempts
Enter-PSSession  - to detect remote access attempts
Icm              - to detect remote command execution attempts
Invoke-Command   - to detect remote command execution attempts
Powersploit      - to detect the use of the powersploit framework
Mimikatz         - to detect the use of the hacking tool Mimikatz
Powercat         - to detect the use of the powercat framework
```



5. I saved the widget as **Breach Detection – Suspicious PowerShell Commands**.

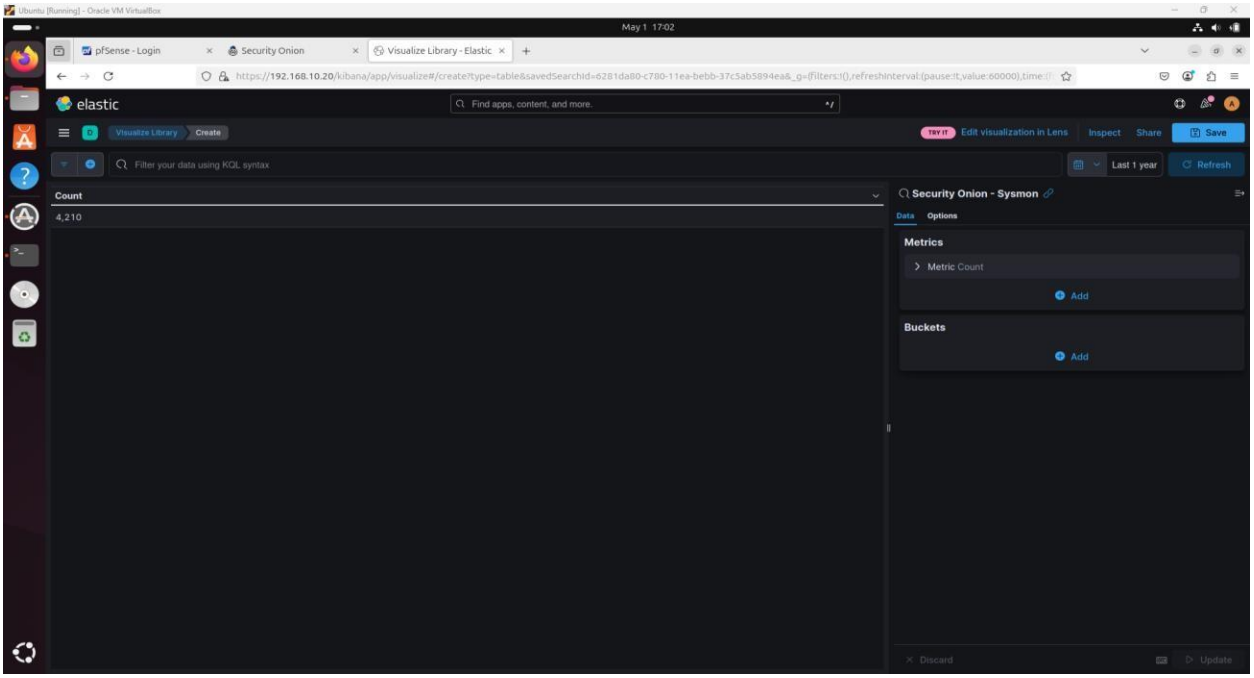


6. The way PowerShell is started can give away any malicious intent or intended abuse as well. The following table summarizes some suspicious ways PowerShell can be invoked (started):

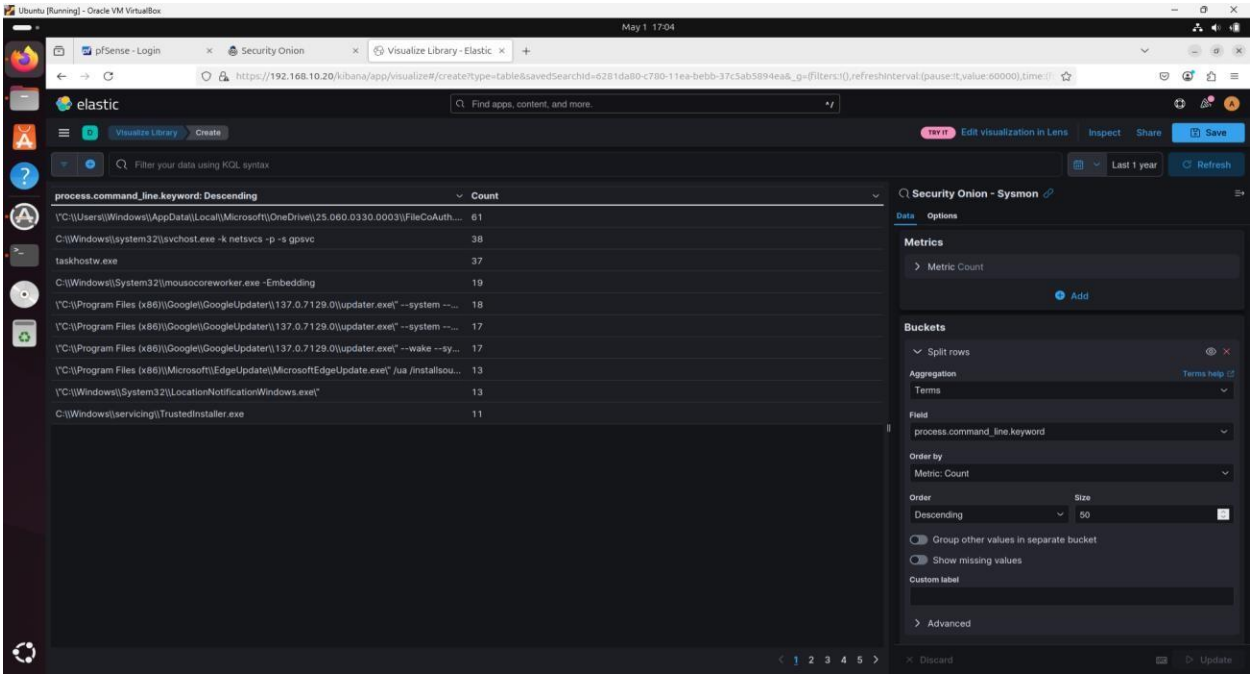
Parameter	Variations	Purpose of parameter
`-noprofile`	`-nop`	Skips loading of `profile.ps1` and avoids logging
`-encoded`	`-e -en -enc`	Lets a user run encoded PowerShell scripts
`-ExecutionPolicy bypass`	`-ep bypass`, `-exp bypass`, `-exec bypass`	Bypasses any execution policy
`-windowStyle hidden`		Prevents the creation of a window; may generate false positives
`-version 2`	`-v 2`, `-version 2.0`	Forces use of PowerShell version 2

7. Detecting suspicious invocation of the PowerShell engine can show malicious intent. We will add a data table that lists any process creation logs that contain these suspicious command-line parameters. The logs this data is in are provided by the Sysmon logging engine that we deployed earlier. Using Wazuh to add Sysmon logging.

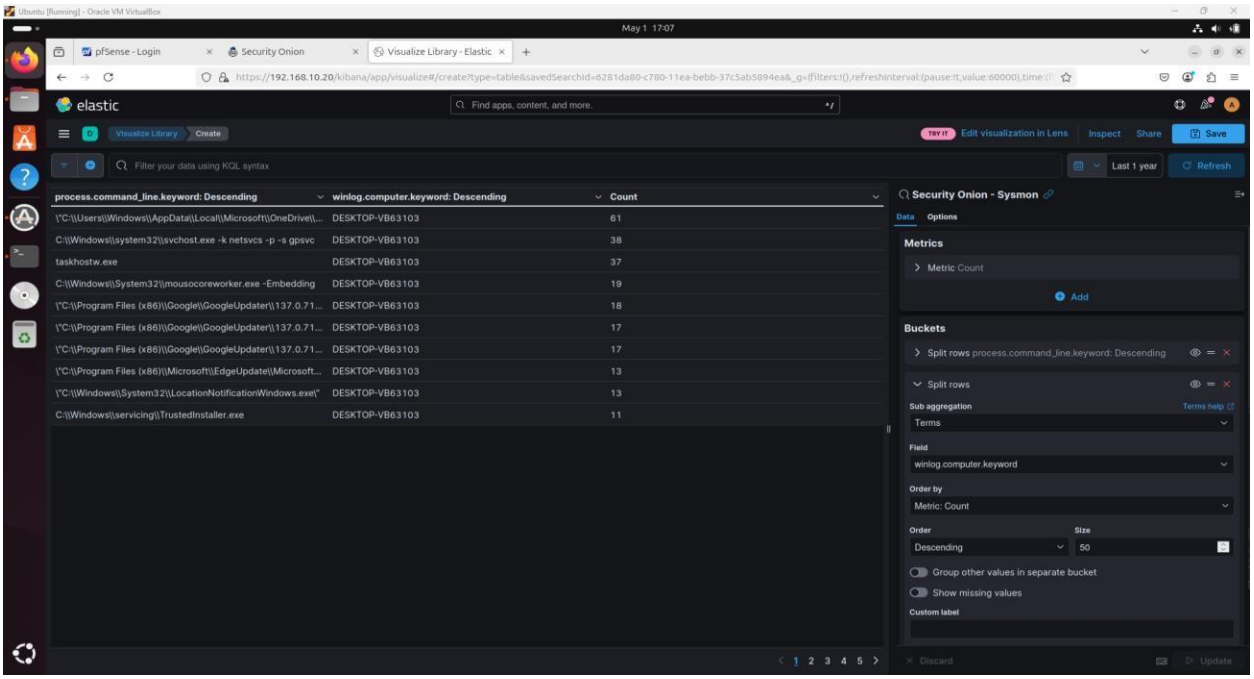
8. Click on **Create Visualization** to add a new widget in the **Breach Detection Portal** dashboard and select **Data table** and search for and add **Security Onion Sysmon** data source.



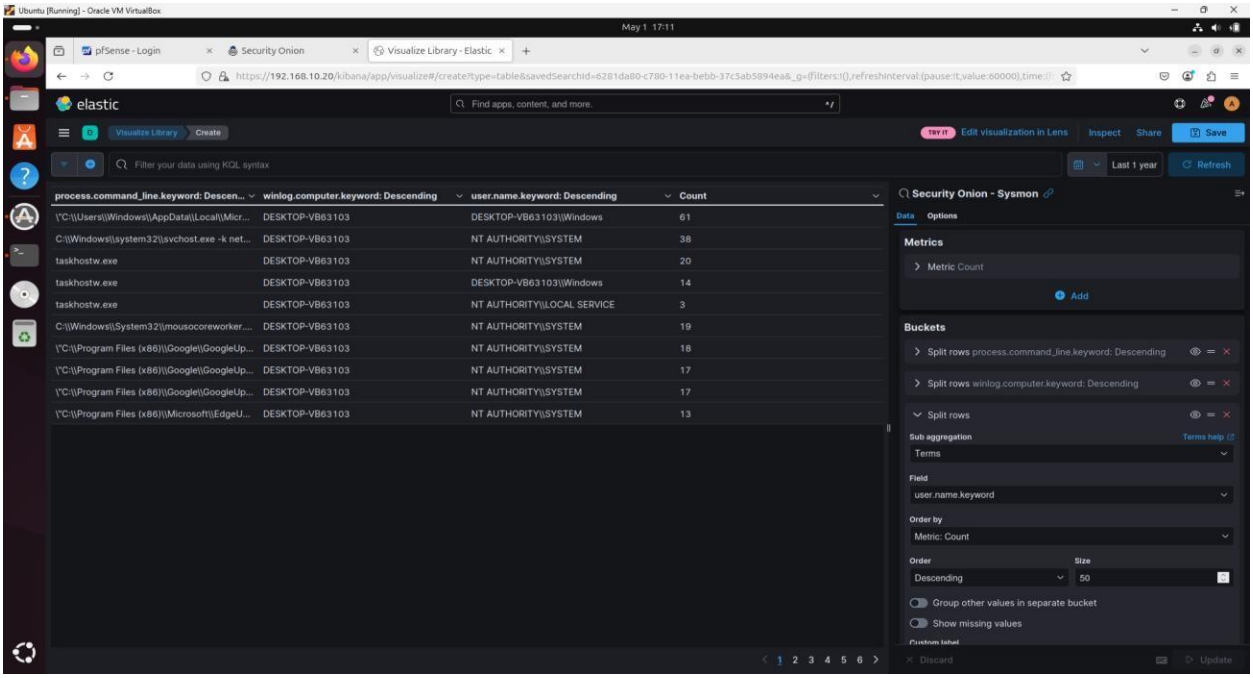
9. Add a **Split Rows** data bucket and set **aggregation** to **Terms**, with **process.command_line.keyword** as the **Field**, set **Size** as **50** and click **Update**.



10. We will be adding the source computer name to show this with the PowerShell invocation event entries. To do this, add an additional **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.computer.keyword** as the **Field** selection. Set the **Size** option to **50** and click **Update**.



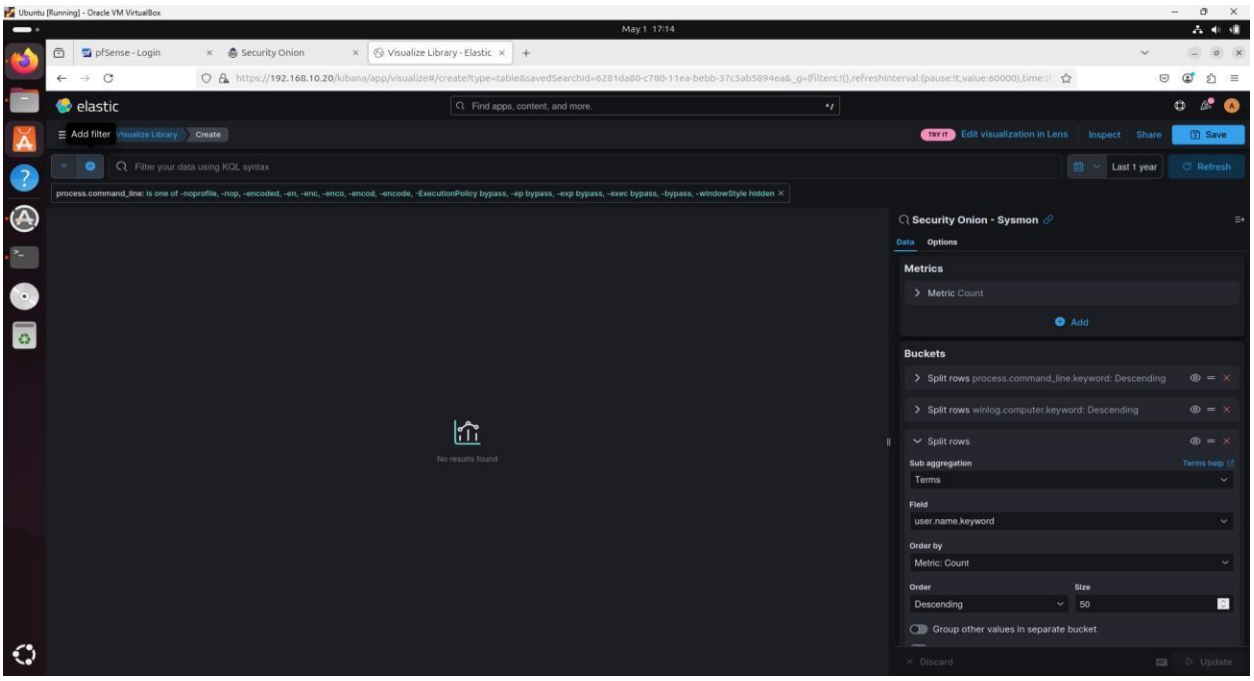
11. We will be adding the username that started the PowerShell process to show with the PowerShell invocation event entries. To do this, add an additional **Split Rows** data bucket and set **Aggregation** to **Terms**, with the **user.name.keyword** as the **Field** selection. Set the **Size** option to **50** and click **Update**.



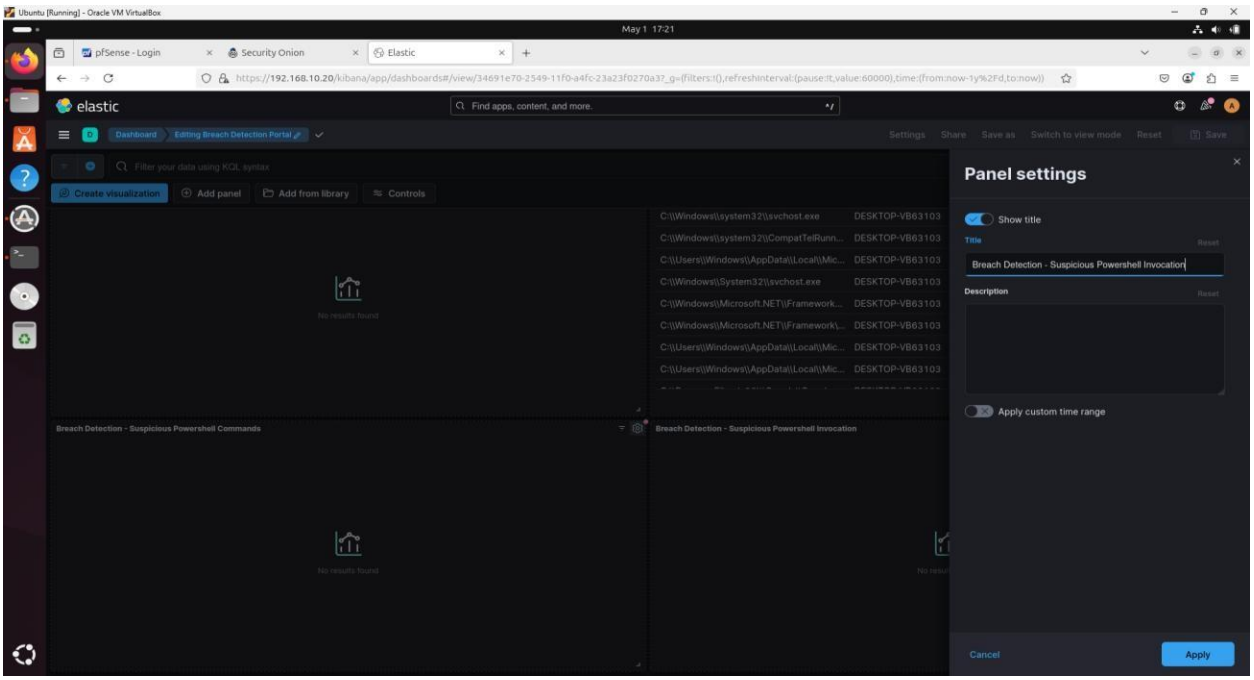
12. This generates a complete list of command-line logs. We are only going to filter for the interesting ones. Click on the **Add Filter** button and set the **Field** option to **process.command_line** and the **Operator** option to **is one of.** Now, one at a time, add in the following values and click **Save** when finished:

-noprofile
-nop
-encoded
-en
-enc
-enco
-encod
-encode
-ExecutionPolicy bypass
-ep bypass
-exp bypass
-exec bypass
-bypass
-windowStyle hidden

The next screenshot shows the configured widget:



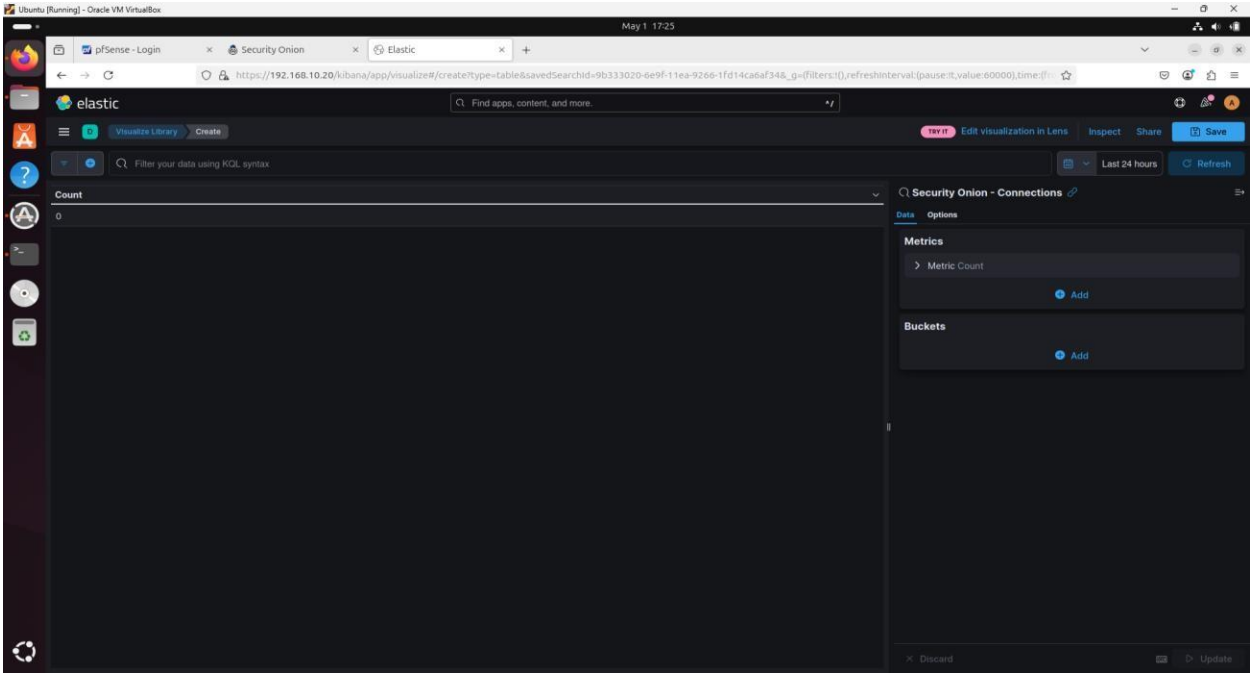
13. Click on **Save**, saving the widget as **Breach Detection – Suspicious PowerShell Invocation**.



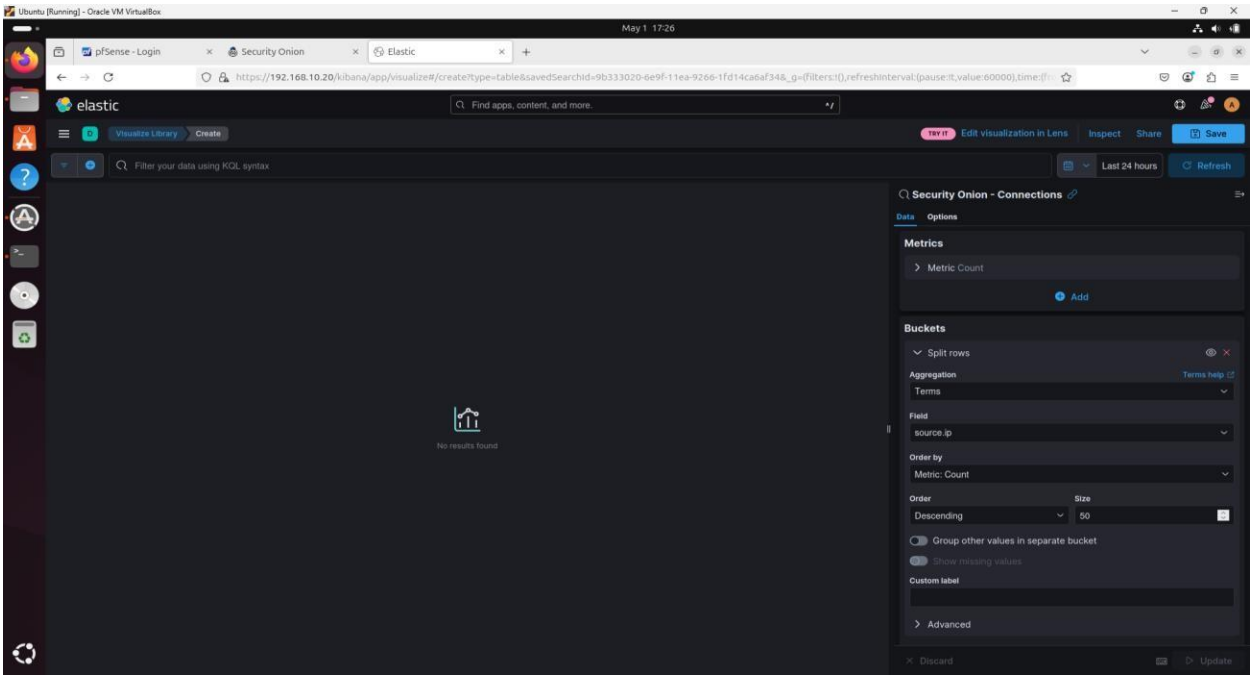
We now have a visualization around suspicious PowerShell commands and how PowerShell is invoked. Next, we are going to look at suspicious network connections.

8.7: Suspicious Egress Connections

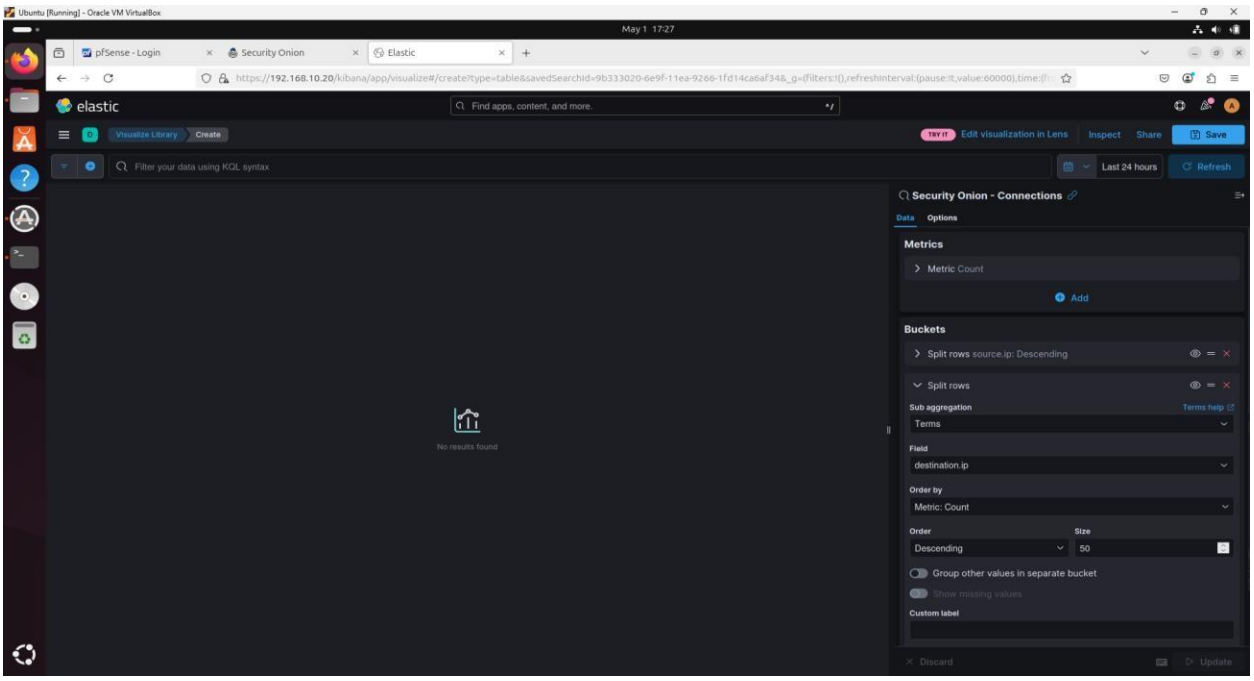
1. I created a Data Table using the Security Onion - Connections data source.



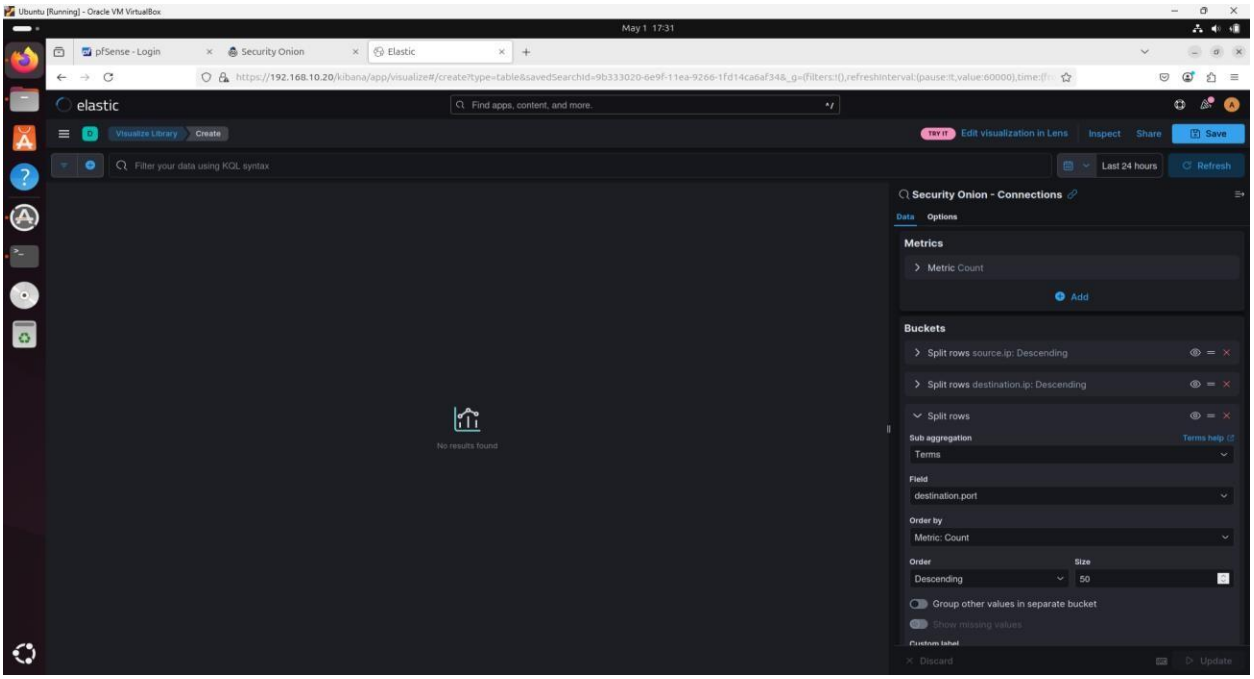
2. I added **Split Rows** buckets for `source.ip`, `destination.ip`, `destination.port`, `network.transport.keyword`, `network.bytes`, and `connection.state_description.keyword`. a **Split Rows** data bucket and set **aggregation** to **Terms**, with **source.ip** as the **Field**, set **Size** as **50** and click **Update**.



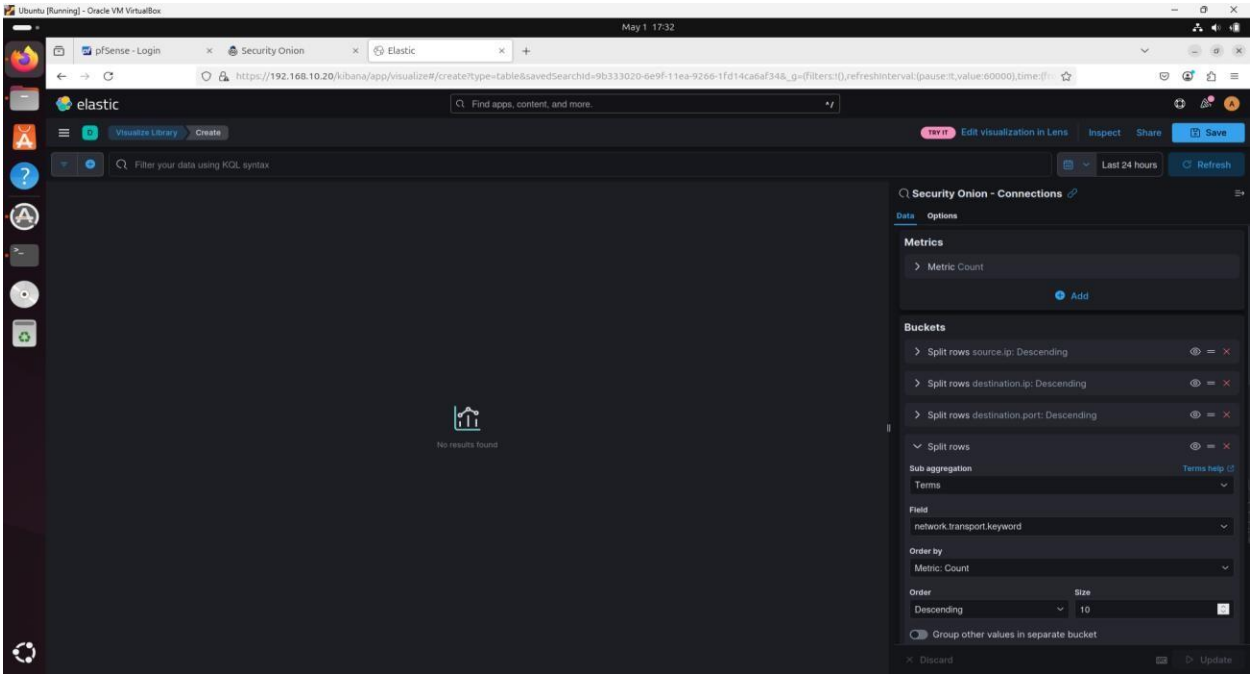
3. I filtered the logs using `connection.local.originator: true` to show outgoing traffic from the internal network.



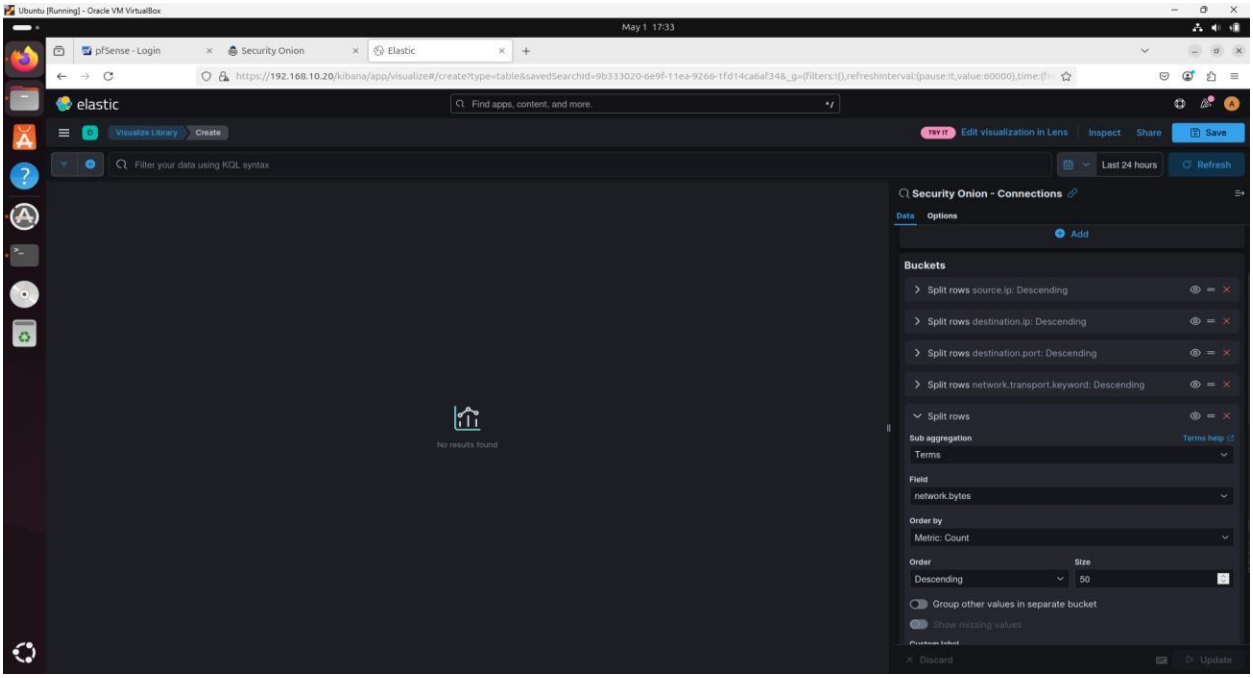
4. I also added connection.local.responder: false to exclude local destinations.



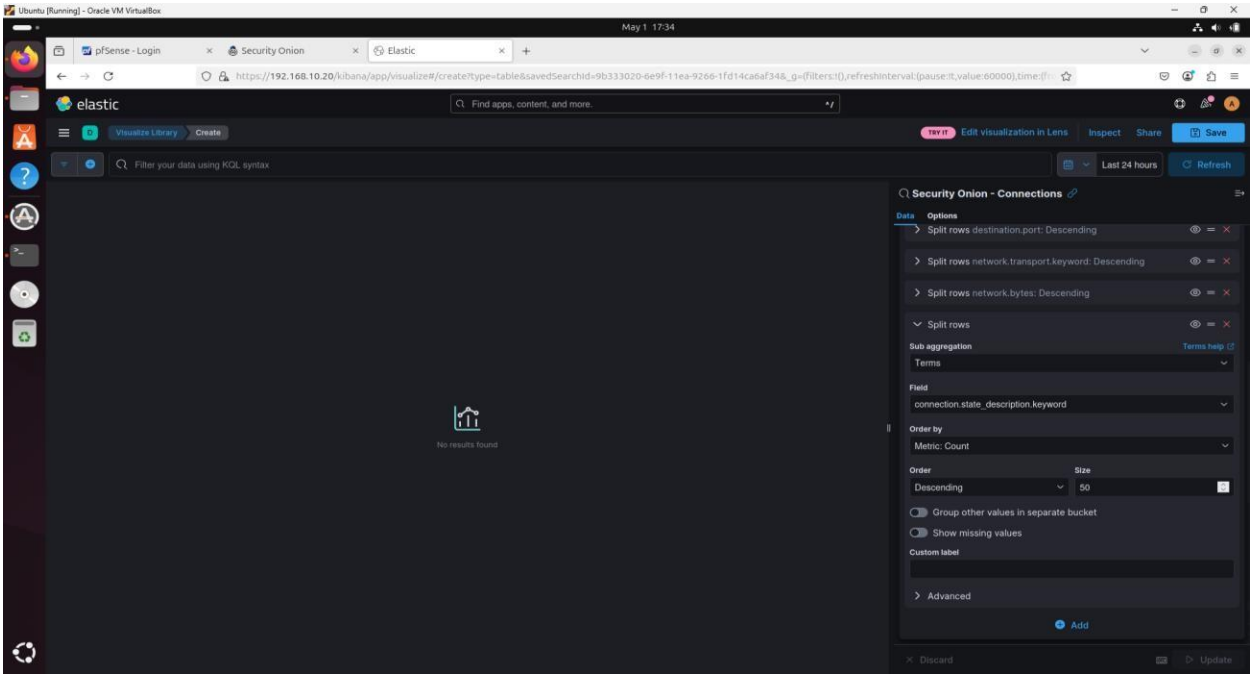
5. After sorting by count in descending order, I saved the visualization as **Breach Detection – Suspicious Egress Connections**.



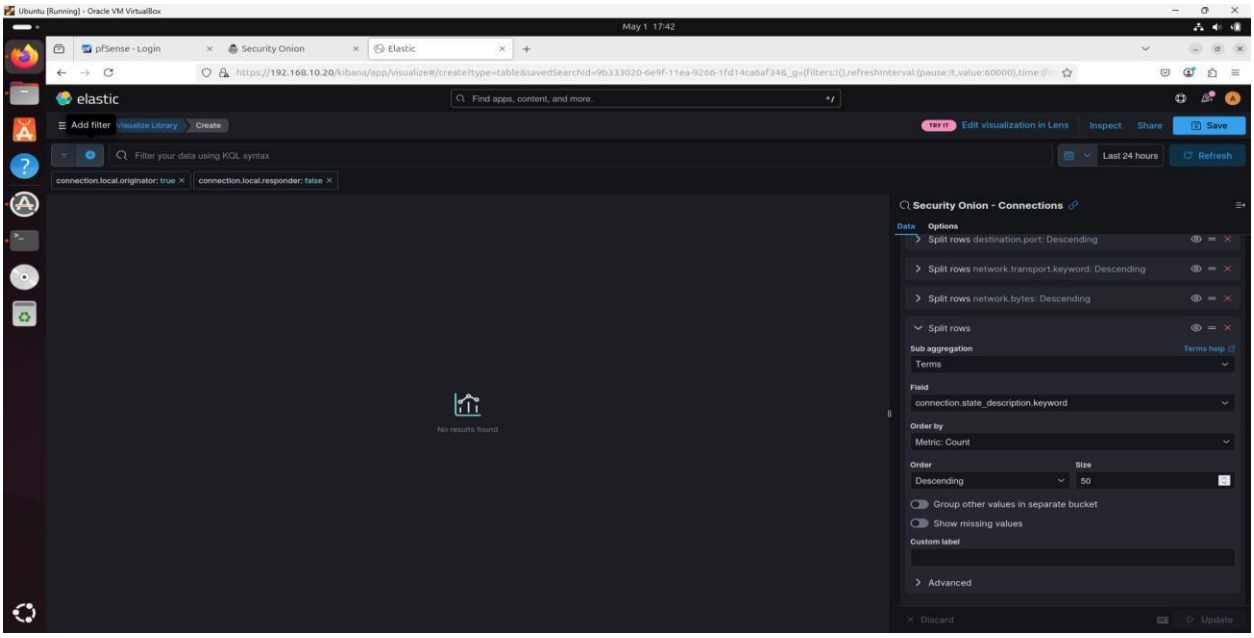
6. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **network.bytes** as the **Field** selection. Set the **Size** option to **50**.



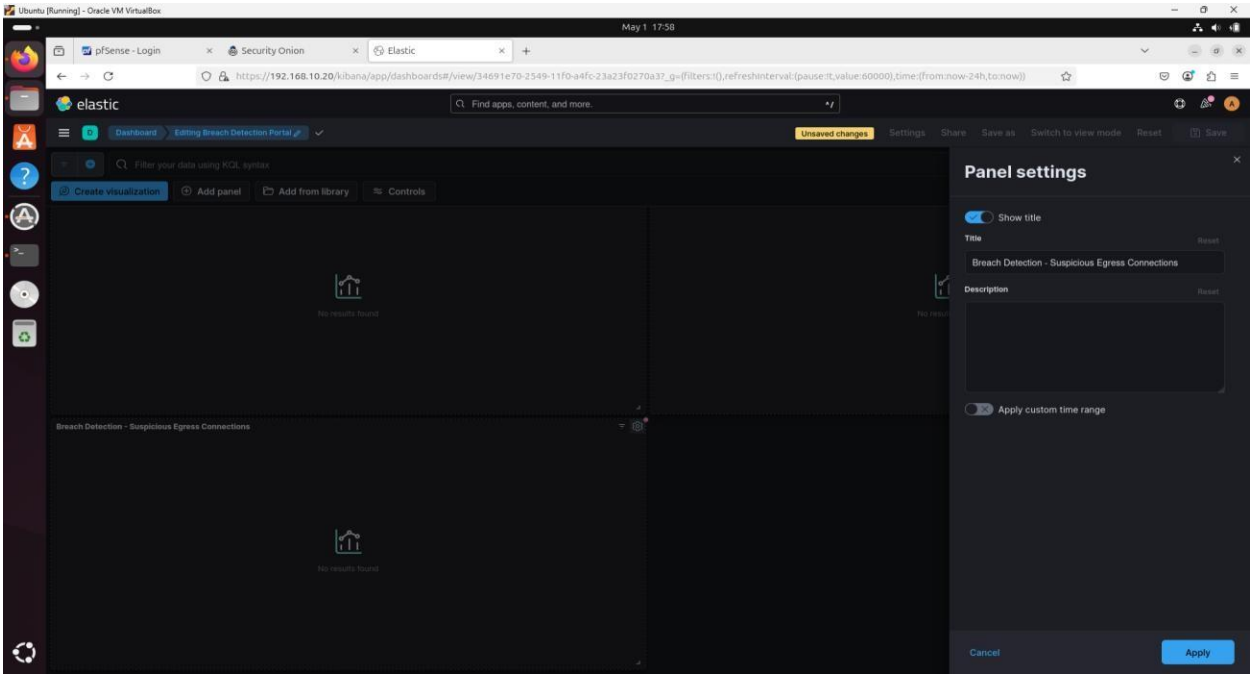
7. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **connection.state_description.keyword** as the **Field** selection. Set the **Size** option to **50.**



8. This generates a summary view of all connection logs. We want to view established connections, but egress only. In order to do this, add the following filters: Filter the data table on **connection.local.originator: true**—this filters the view to show connections originating from the local network (Industrial Zone) only. Filter the data table on **connection.local.responder: false**— this filters the view to show connections to external destinations (enterprise and internet) only.

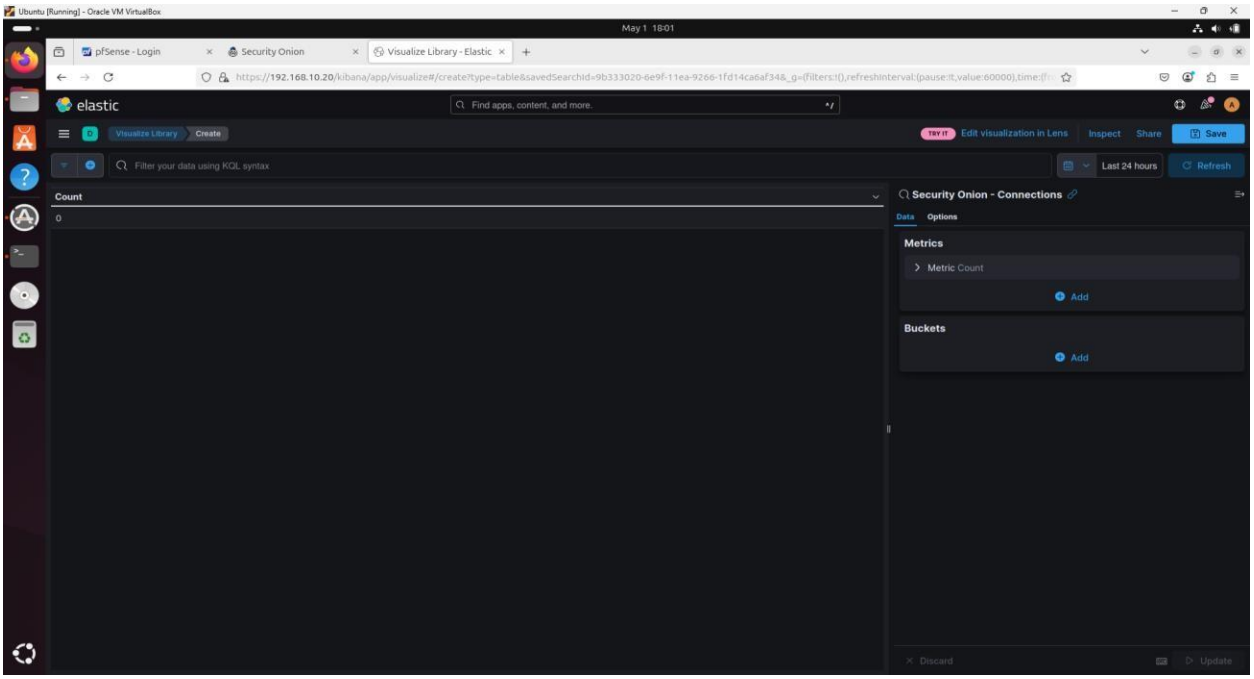


9. Make sure to sort by **Count** (descending) and click on **Save**, saving the widget as **Breach Detection – Suspicious Egress Connections**.

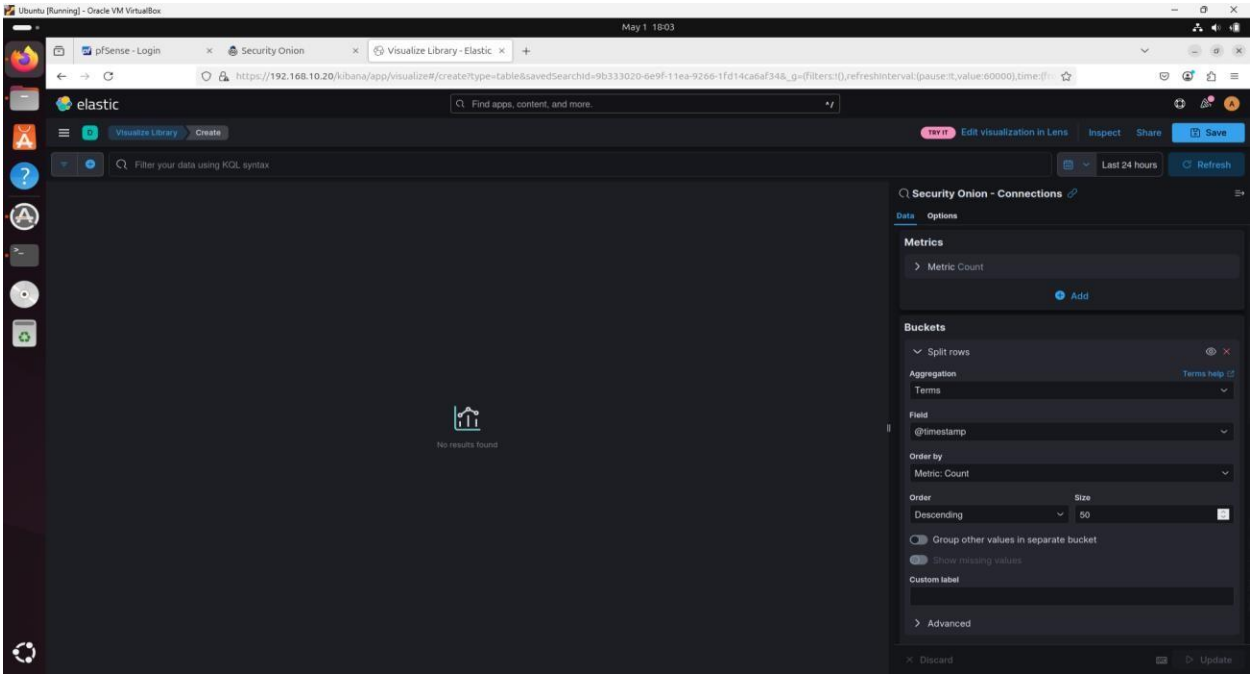


8.8: Suspicious Ingress Connections

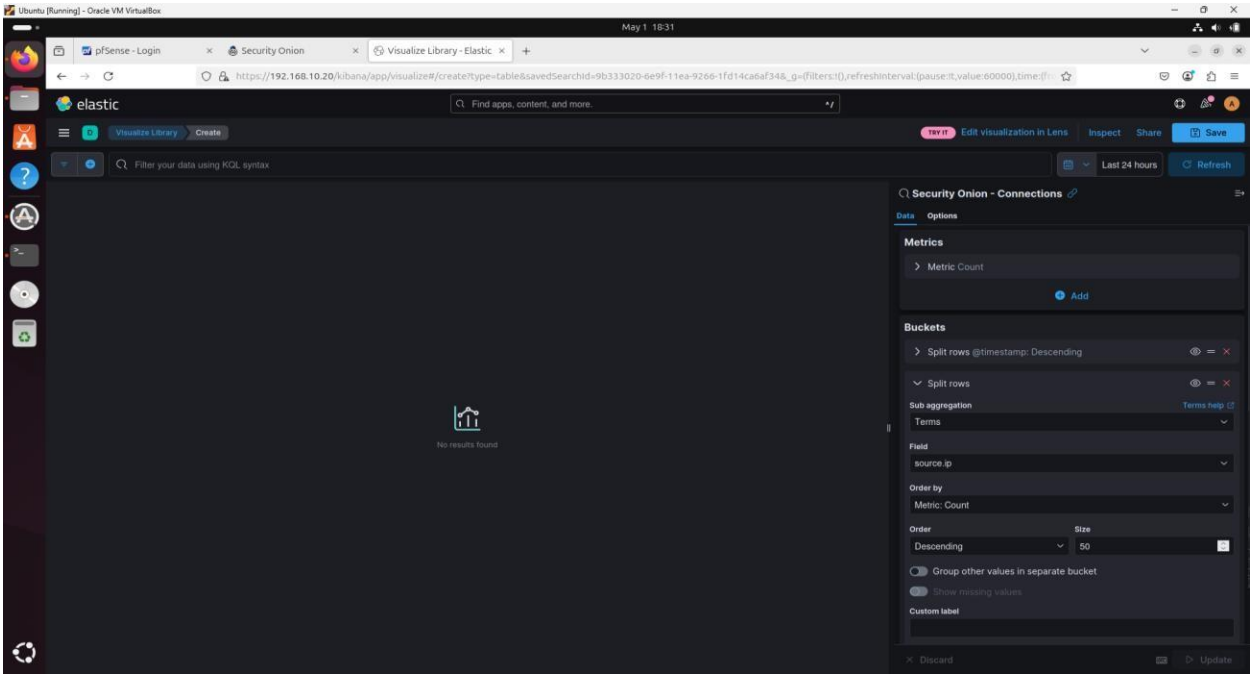
1. I created another **Data Table** using the **Connections** data source.



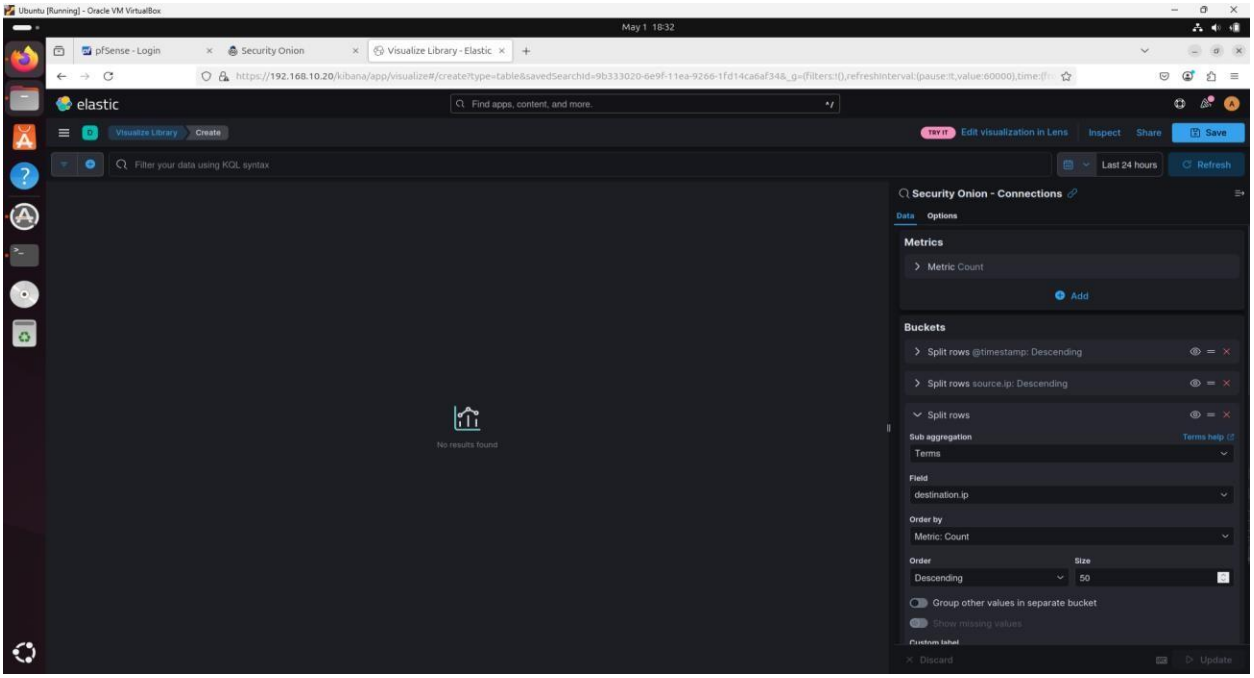
2. I added **Split Rows** for `@timestamp`, `source.ip`, `destination.ip`, `destination.port`, `network.transport.keyword`, `event.duration`, `network.bytes`, and `connection.state_description.keyword`.



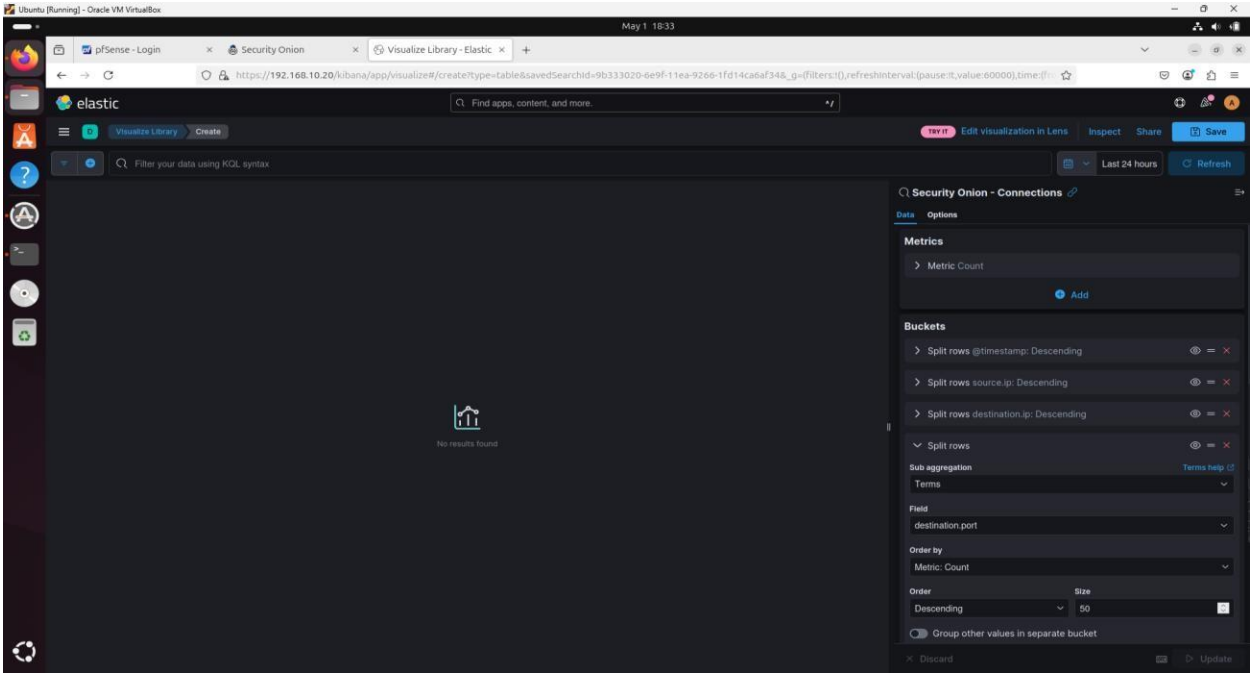
3. I filtered the logs using `NOT source.ip: 172.25.100.0/24` to remove local sources.



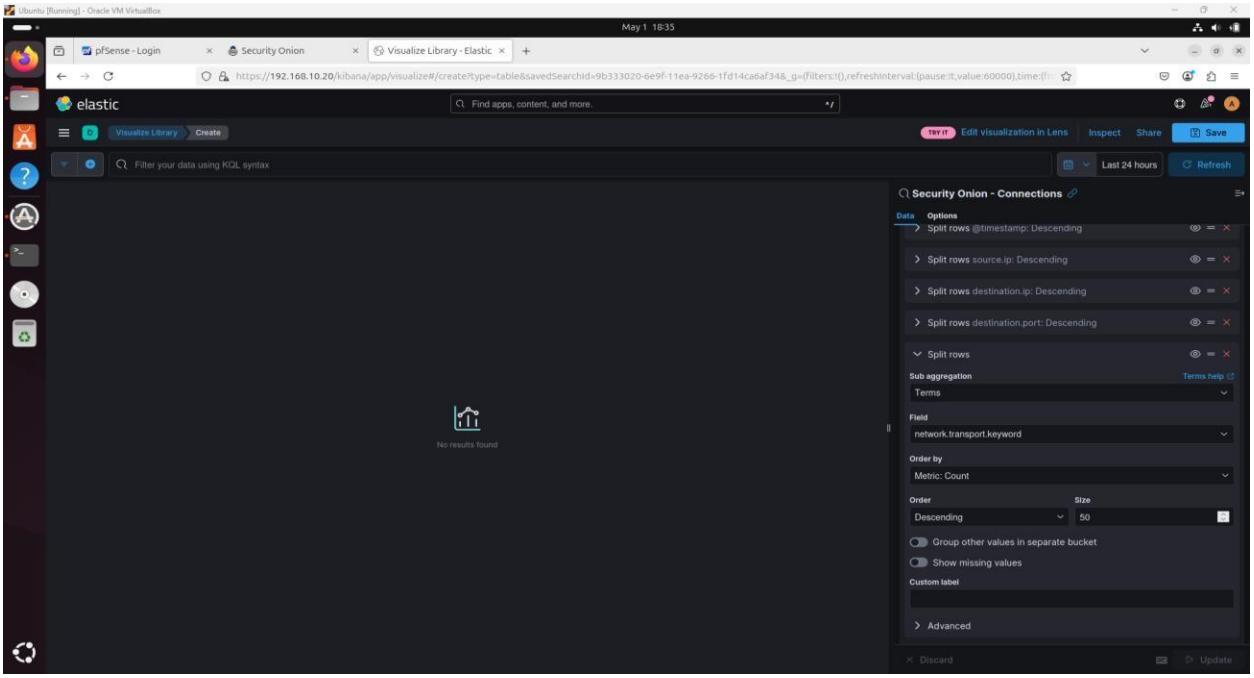
4. I applied a second filter using connection.local.responder: true to show incoming traffic to internal systems.



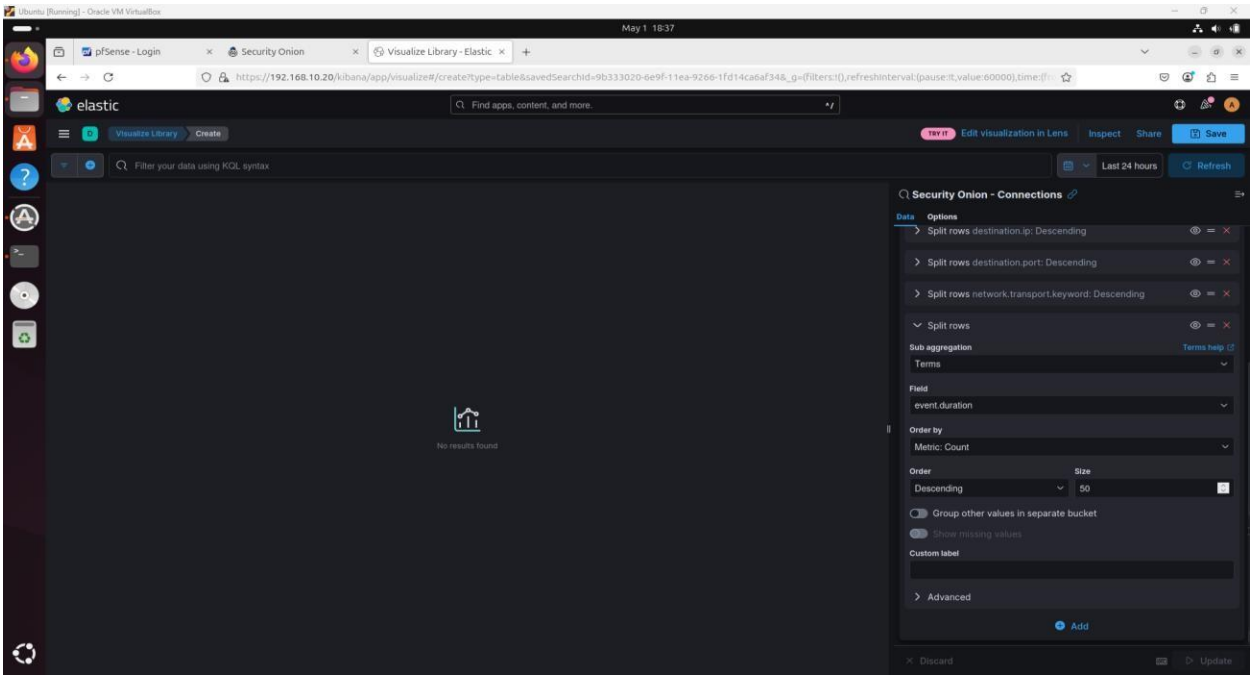
5. After sorting by duration in descending order, I saved the widget as **Breach Detection – Suspicious Ingress Connections**.



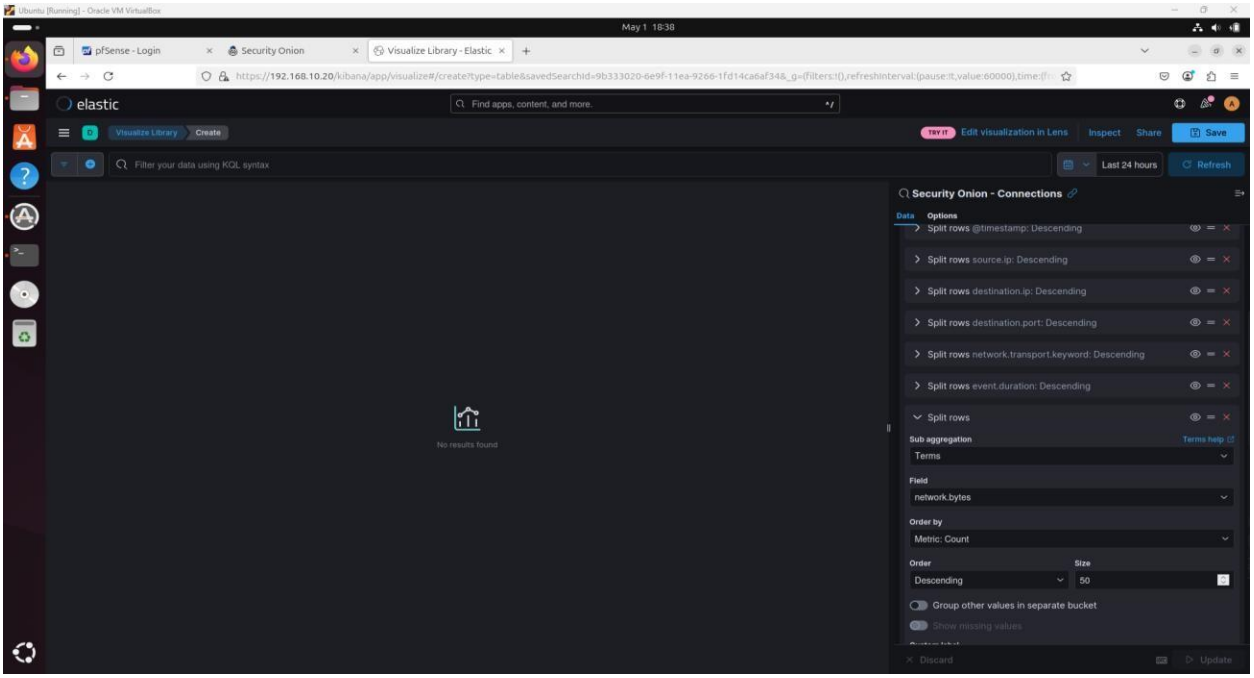
6. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **network.transport.keyword** as the **Field** selection. Set the **Size** option to **50**.



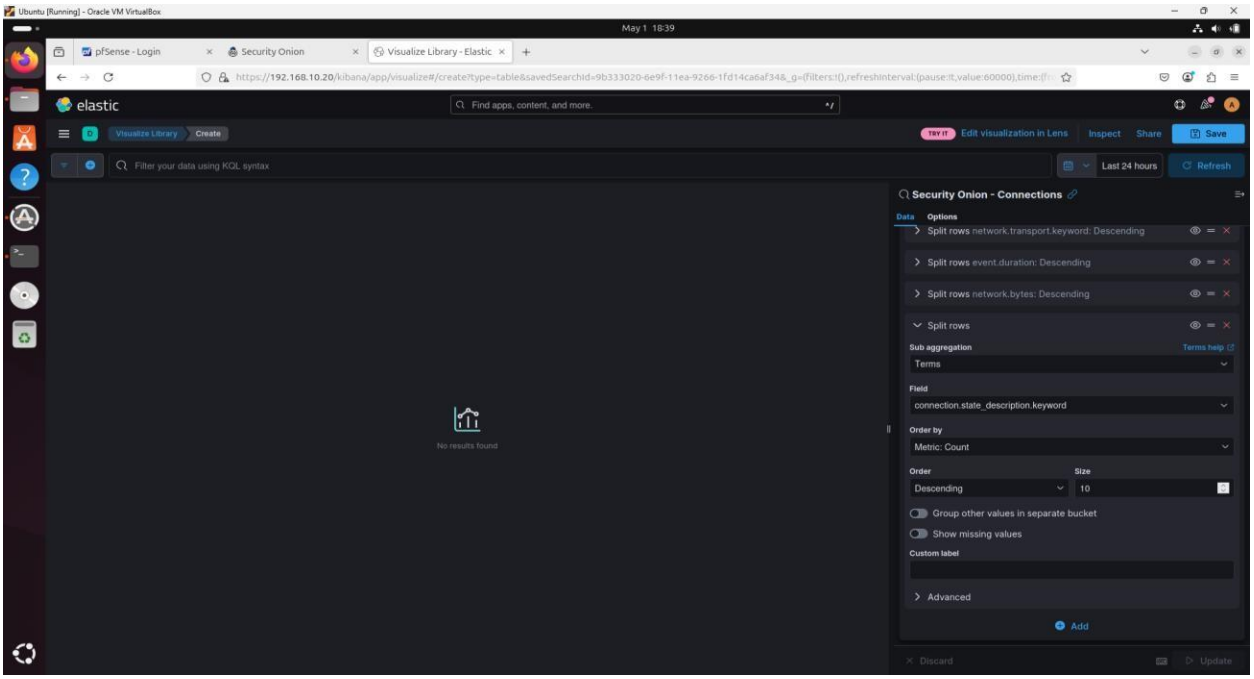
7. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **event.duration** as the **Field** selection. Set the **Size** option to **50**.



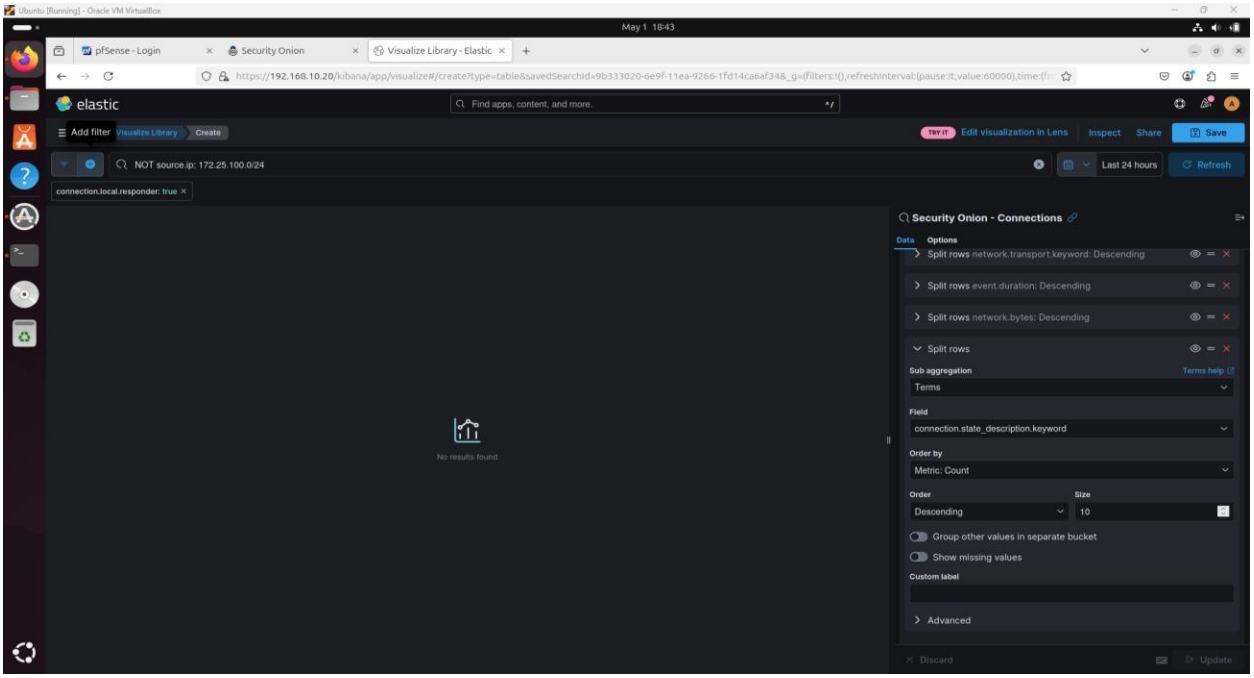
8. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **network.bytes** as the **Field** selection. Set the **Size** option to **50**.



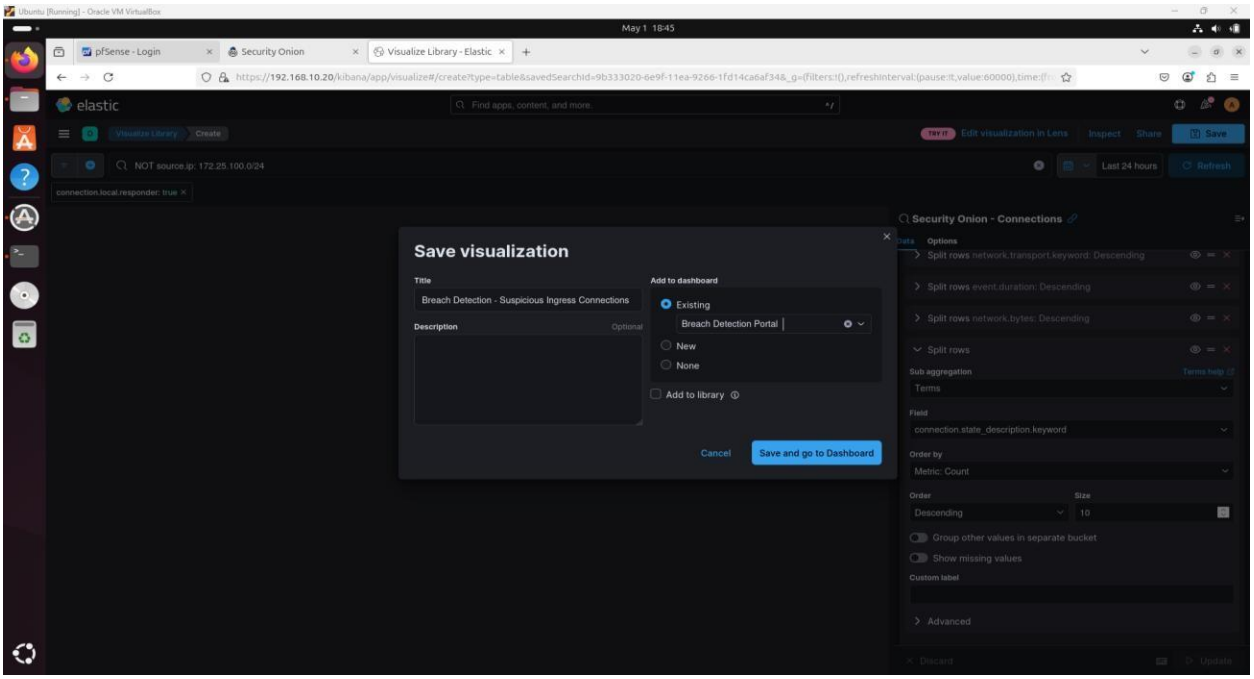
9. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **connection.state_description.keyword** as the **Field** selection. Set the **Size** option to **10**.



10. This generates a summary view of all connection logs. We want to view established connections, but egress only. To do this, add the following filters: Enter the following search string for the view: **NOT source.ip: 172.25.100.0/24**. This will filter out any IP addresses that are local to the industrial network (adapt this to your subnet). Filter the data table on **connection.local.responder: true**—this filters the view to show connections to local destinations (Industrial Zone) only.

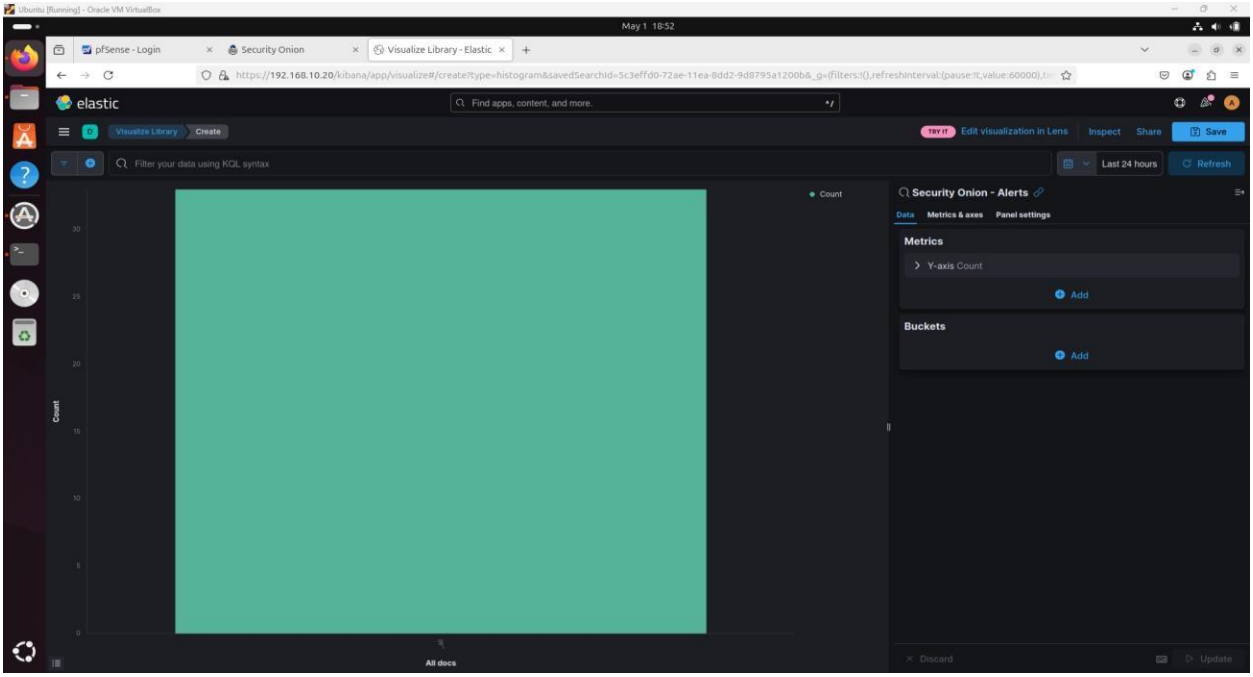


11. Make sure to sort by **Duration** (descending) and click on **Save**, saving the widget as **Breach Detection – Suspicious Ingress Connections.**

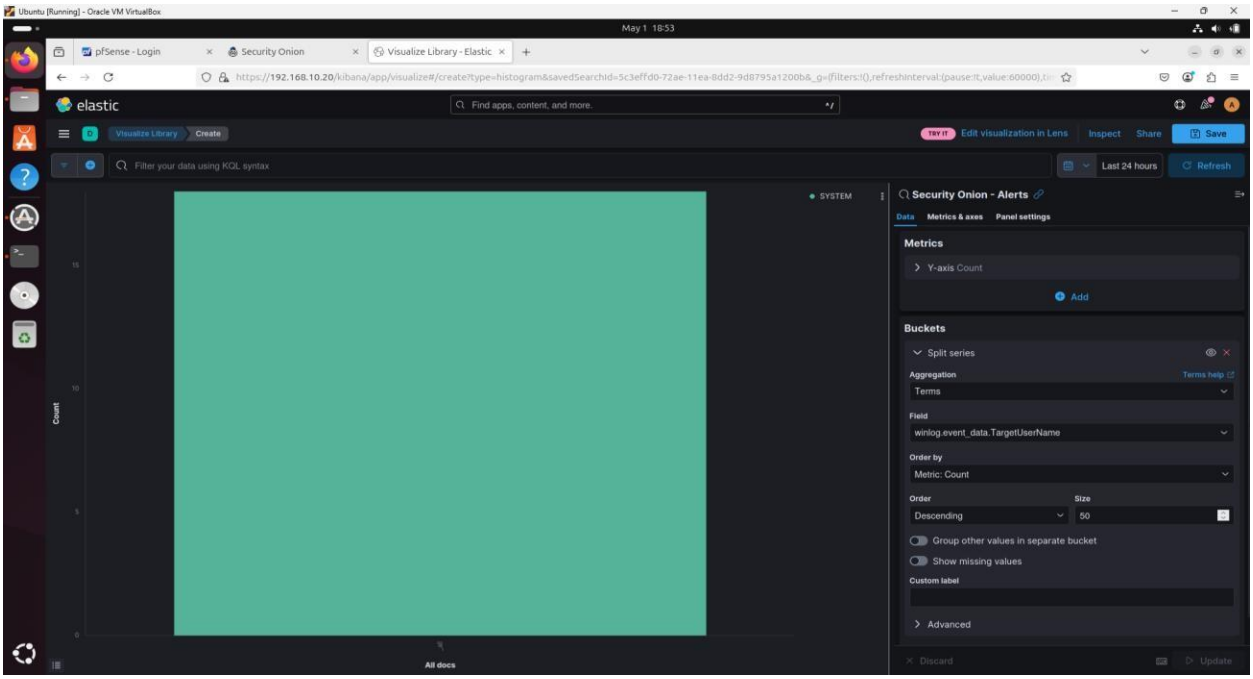


8.9: Failed User Login Attempts

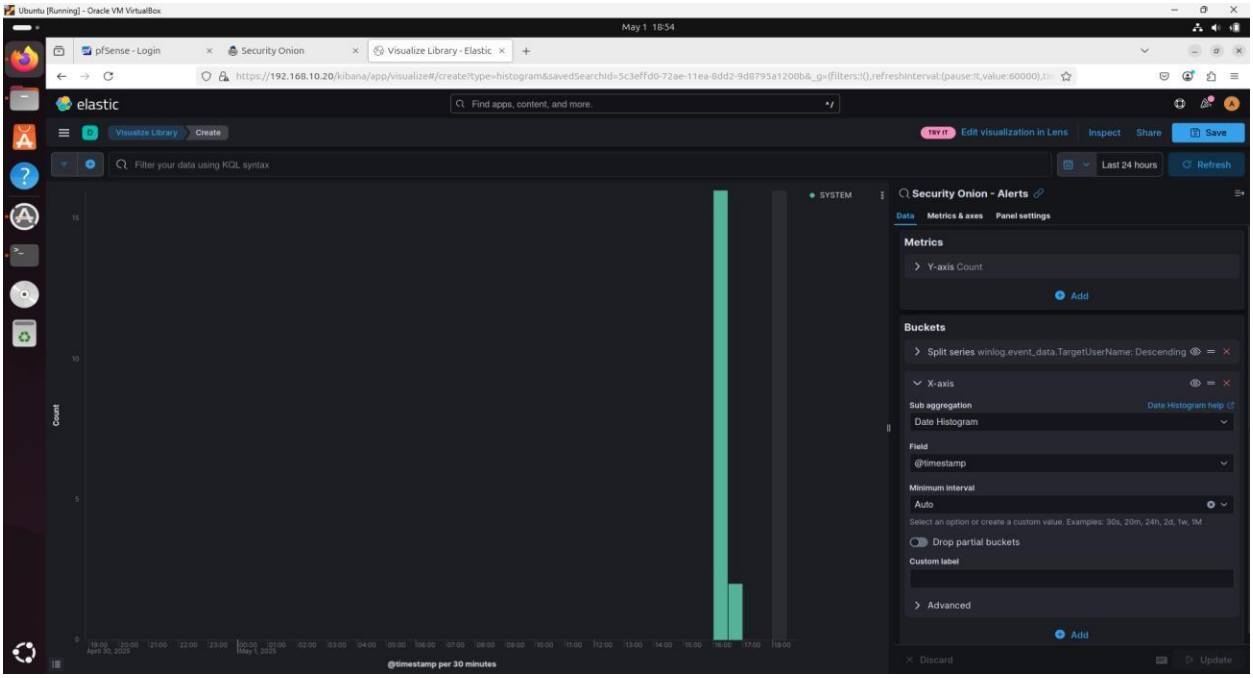
1. I added a **Vertical Bar** visualization using **Security Onion – Alerts**.



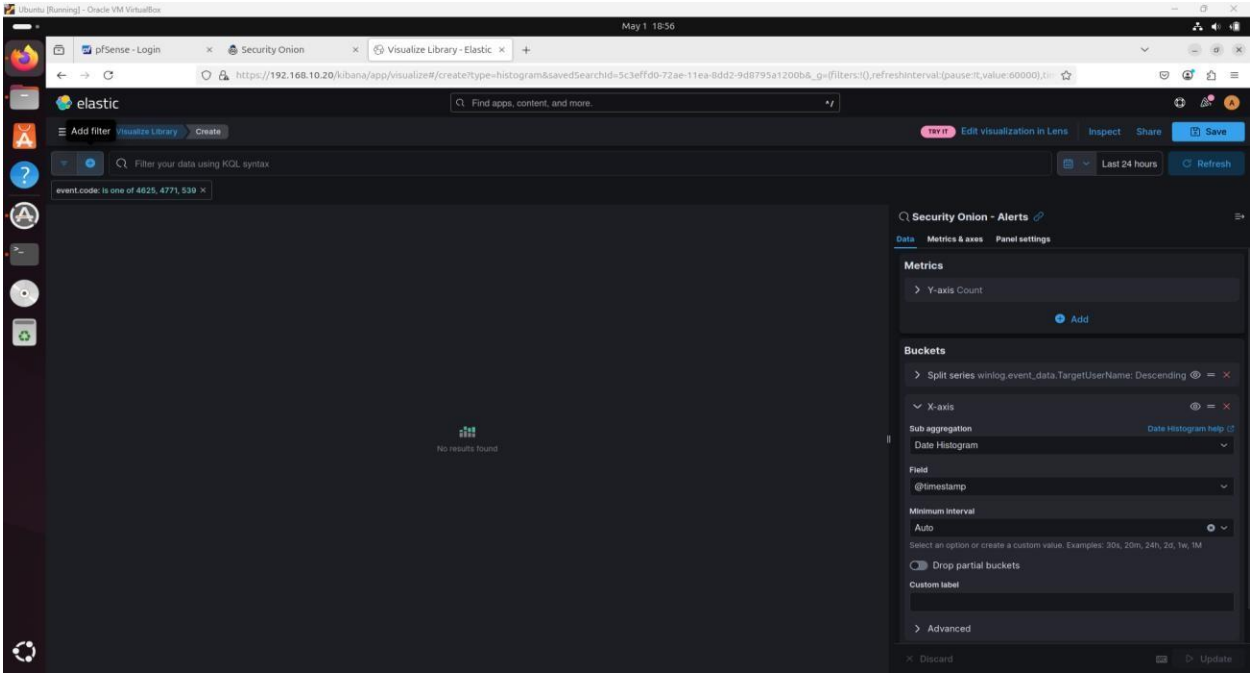
2. I set a **Split Series** using `winlog.event_data.targetUserName.keyword`, size 50.



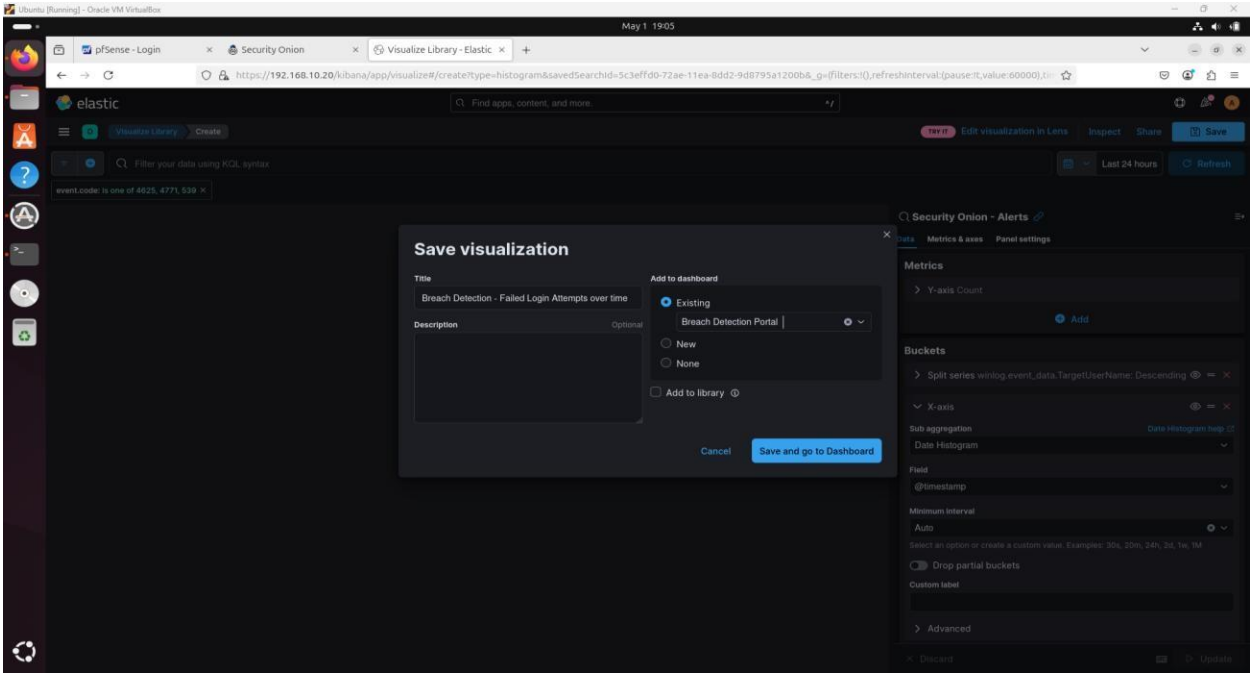
3. I set a Split Series using winlog.event_data.targetUserName.keyword, size 50.



4. I filtered by event codes 4625, 4771, and 539 which indicate failed login attempts.

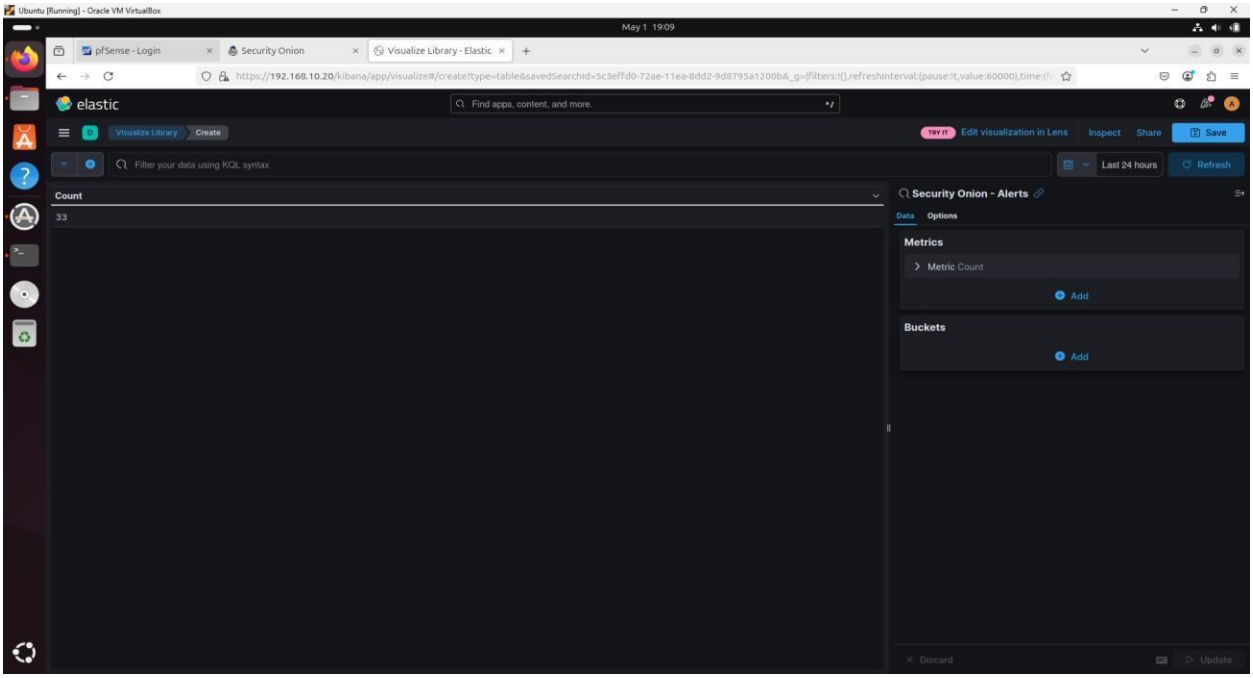


5. I saved this chart as Breach Detection – Failed Login Attempts Over Time.

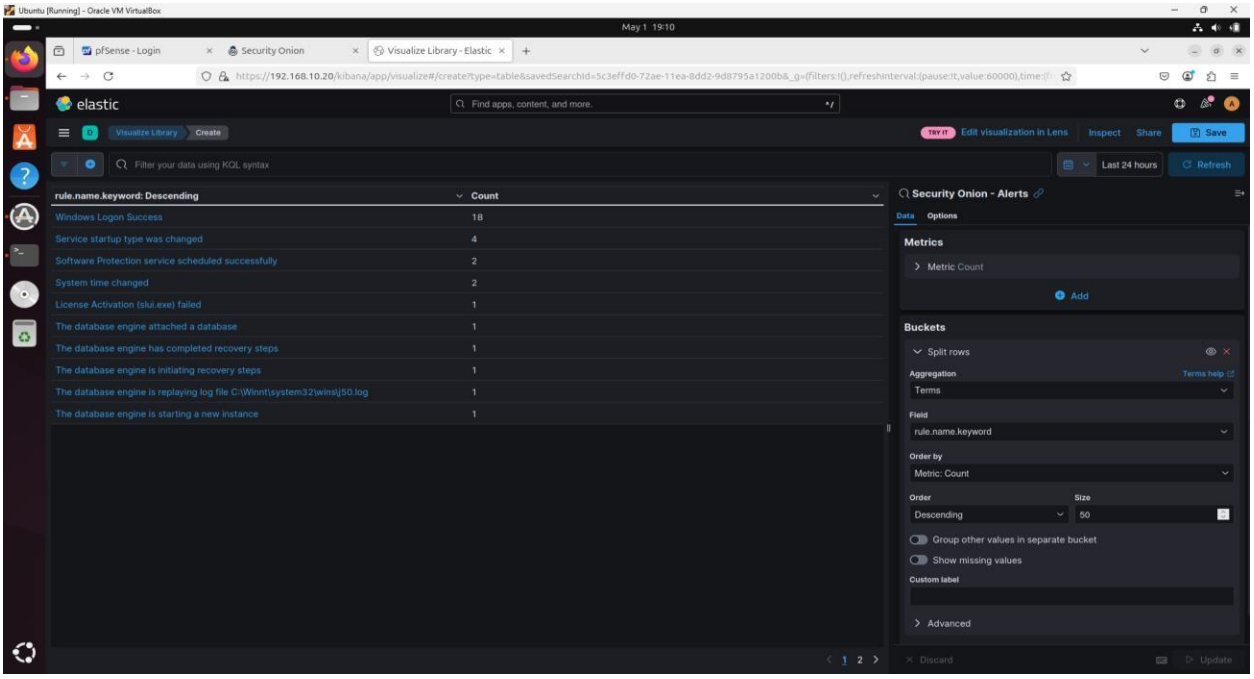


8.10: New user creation and changes to user accounts

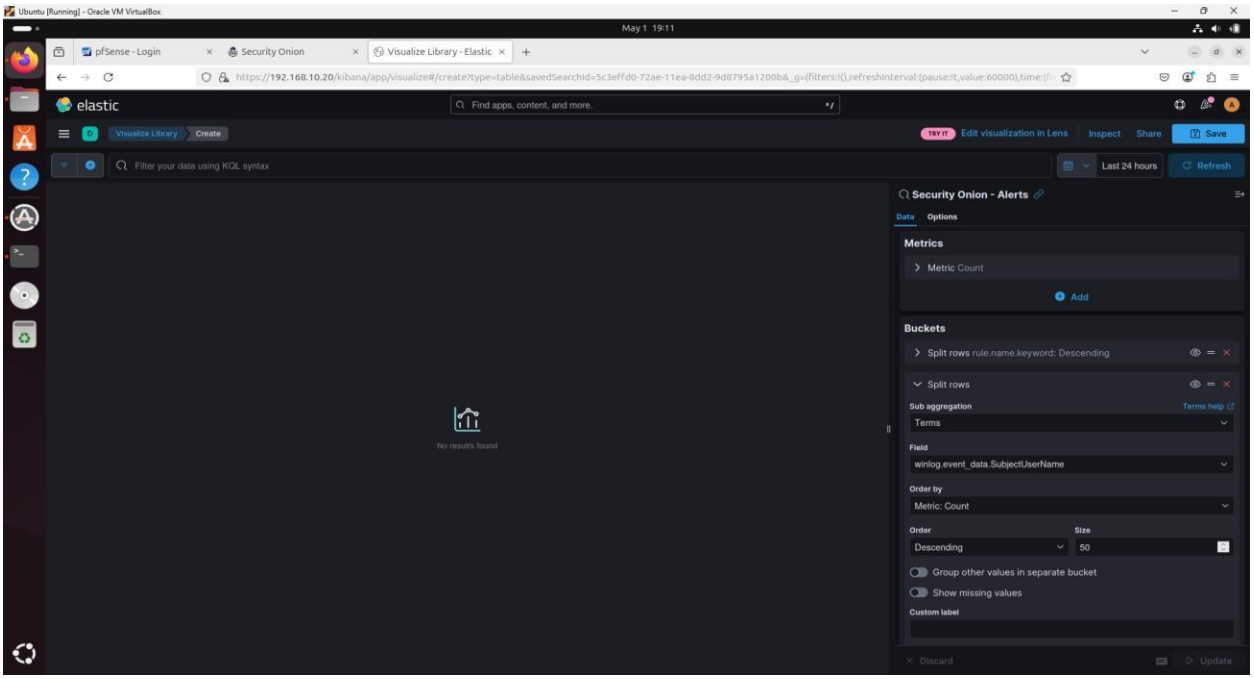
1. I created a Data Table using Security Onion – Alerts as the data source.



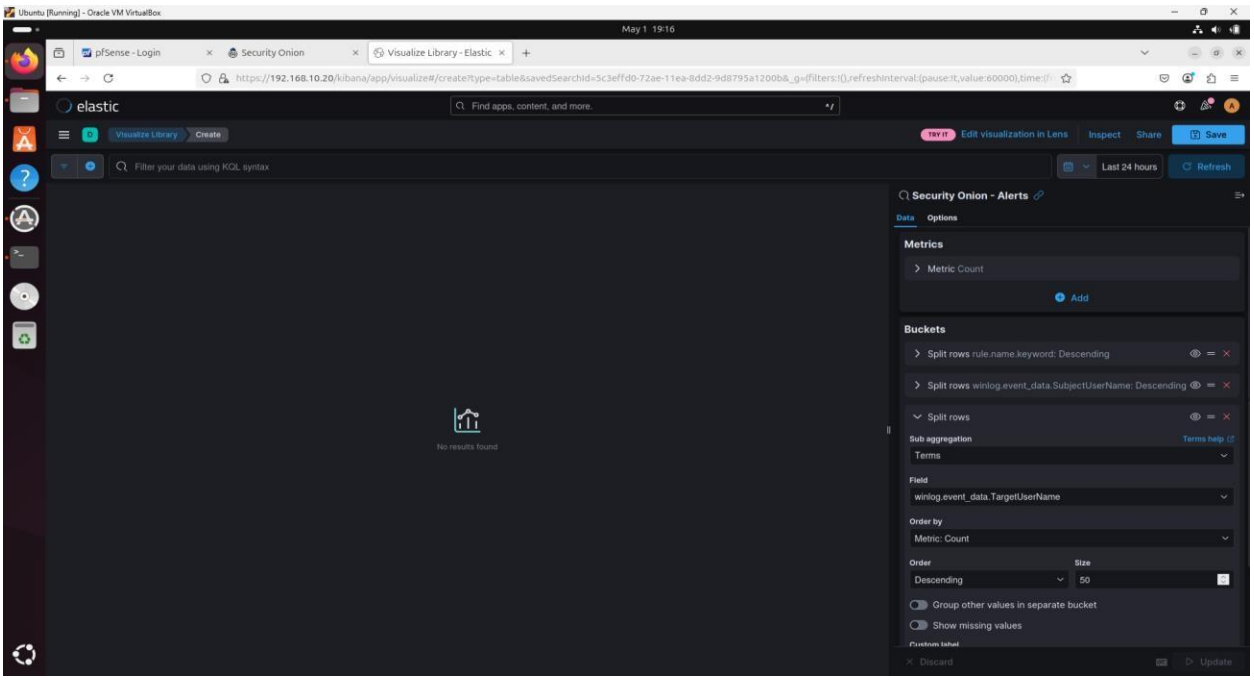
2. I added **Split Rows** buckets for `rule.name.keyword`, `winlog.event_data.subjectUserName.keyword`, `winlog.event_data.targetUserName.keyword`, and `agent.name.keyword`.



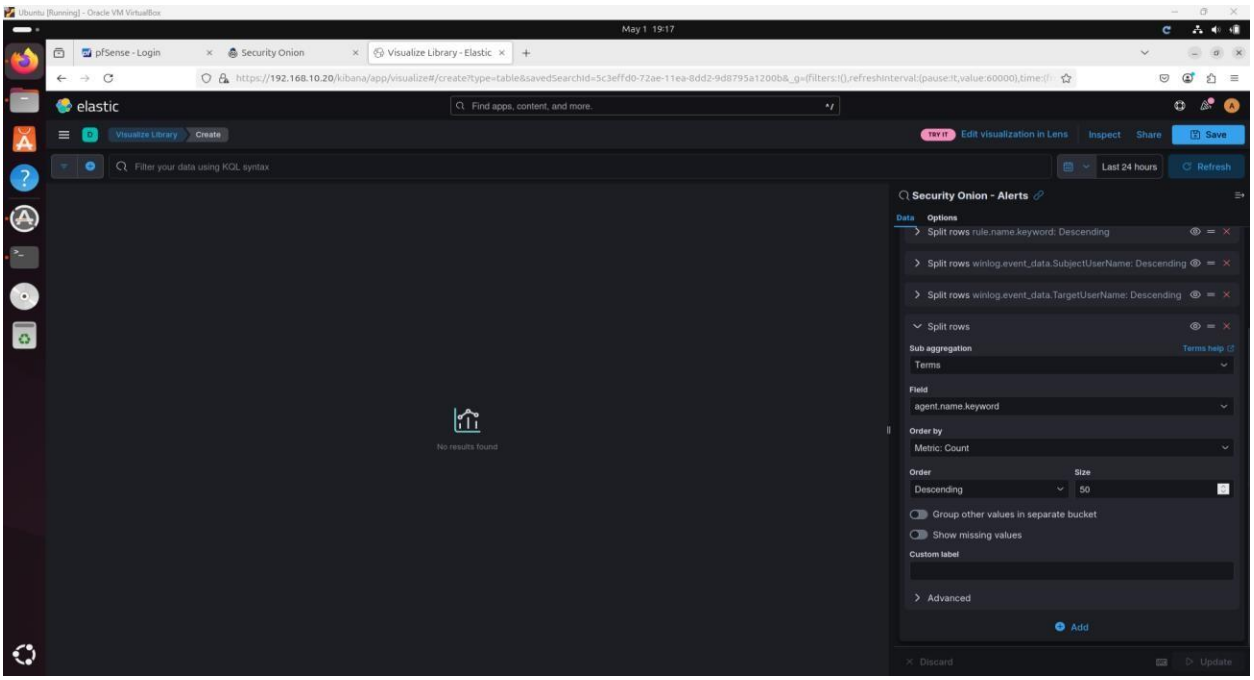
3. I filtered by event codes: `4720`, `4738`, `4732`, `4728`, and `4722`—each corresponding to account creation, privilege changes, or enabling user accounts.



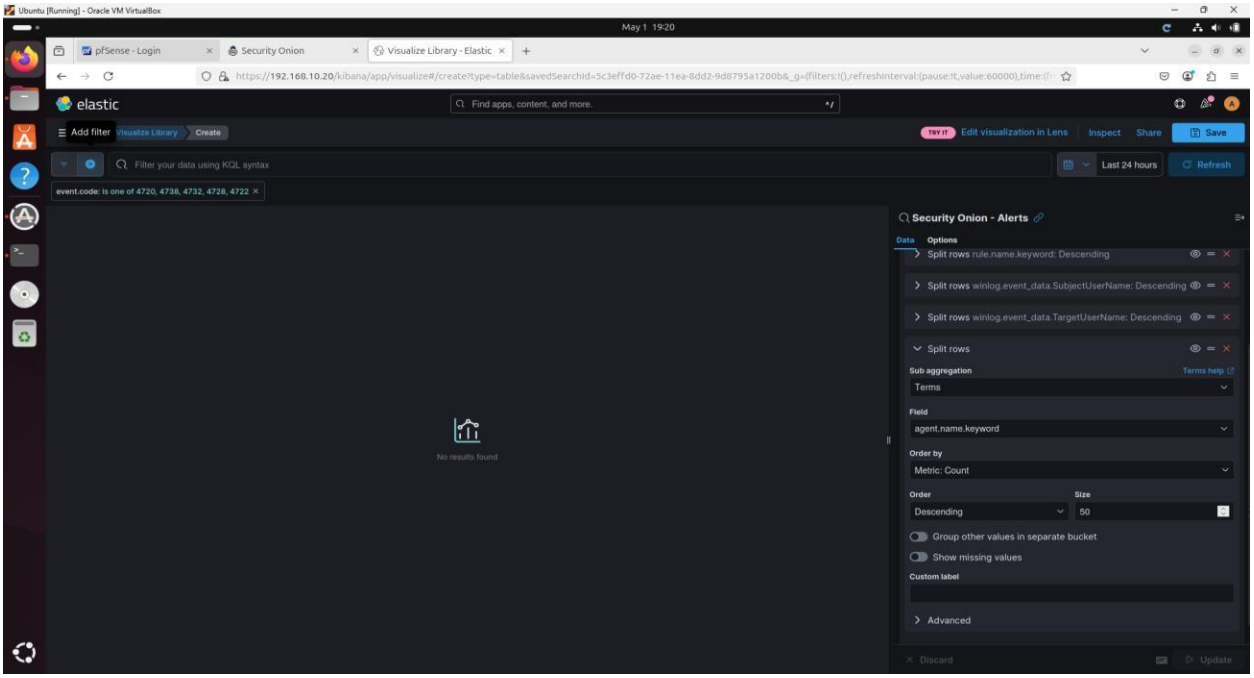
4. I saved this as **Breach Detection – User Account Alerts Summary.**



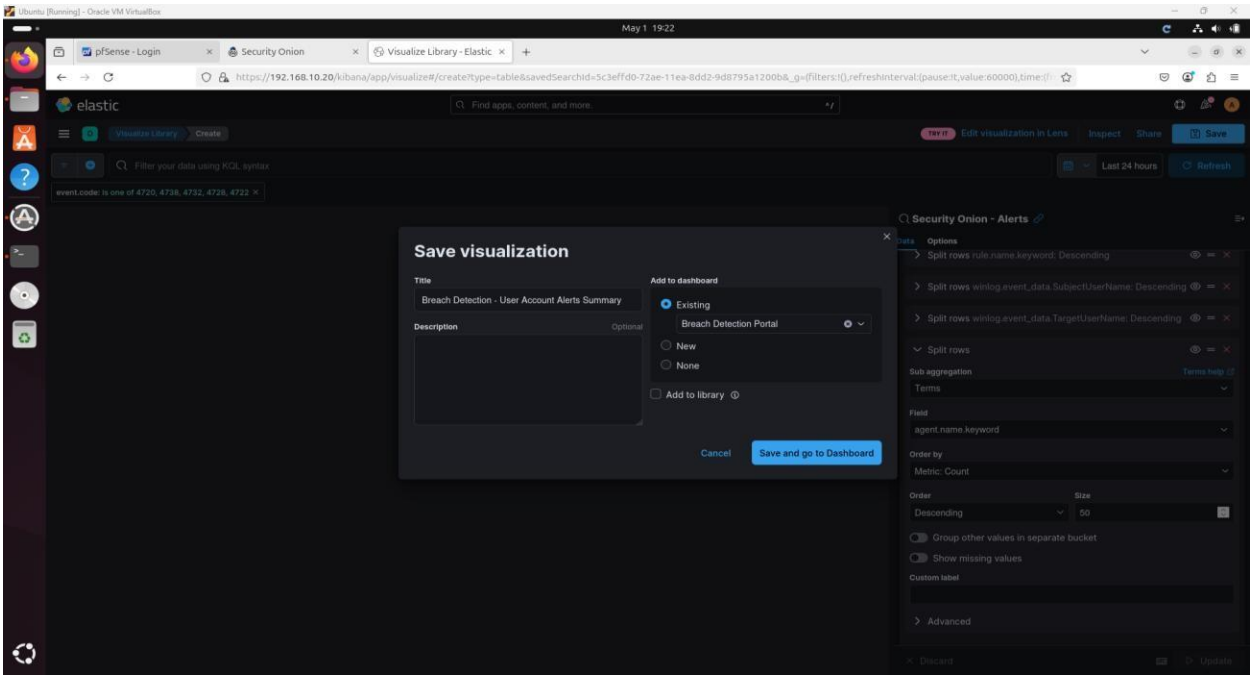
5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **agent.name.keyword** as the **Field** selection. Set the **Size** option to **50**.



6. click on the **Add Filter** button, then set the **Field** option to **event.code** and the **Operator** option to **is one of**. Set the values to the following: **4720**—This is the event ID for new user creation; **4738**—This is the event ID for account changes; **4732**—This is the event ID indicating a user was added to a local privileged group; **4728**—This is the event ID indicating a user was added to a global privileged group; **4722**—This is the event ID indicating a user account was enabled.

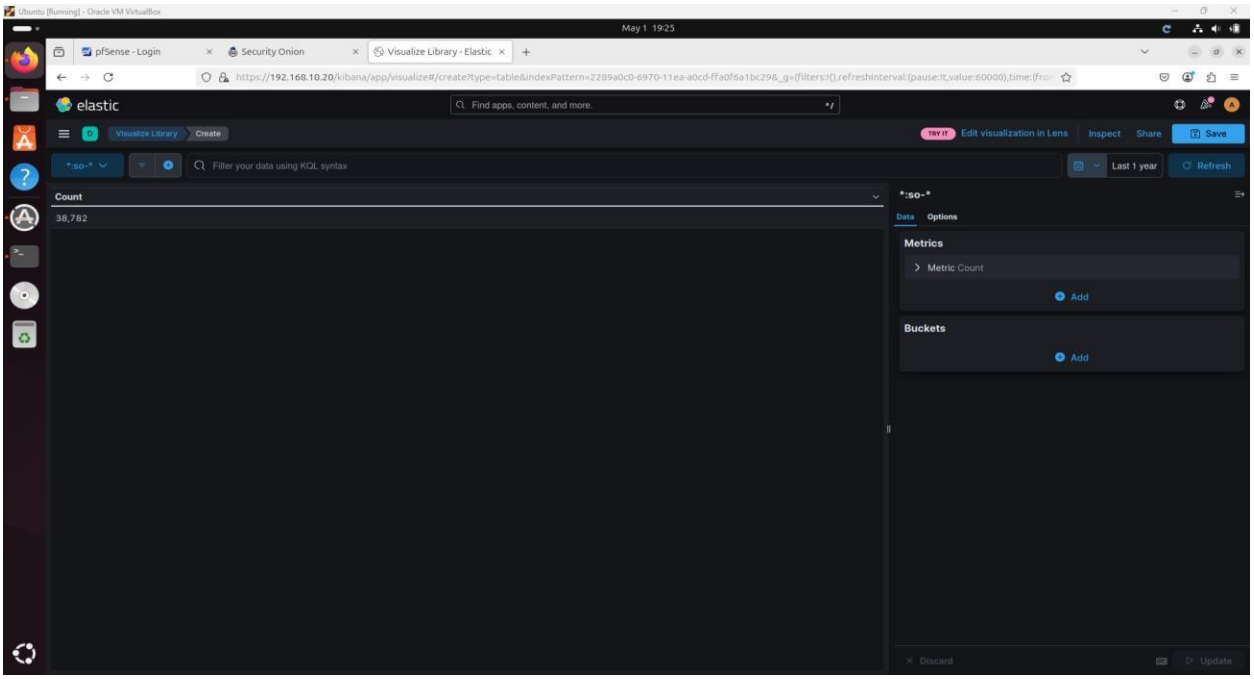


7. Click on **Save**, saving the widget as **Breach Detection – User Account Alerts Summary.**

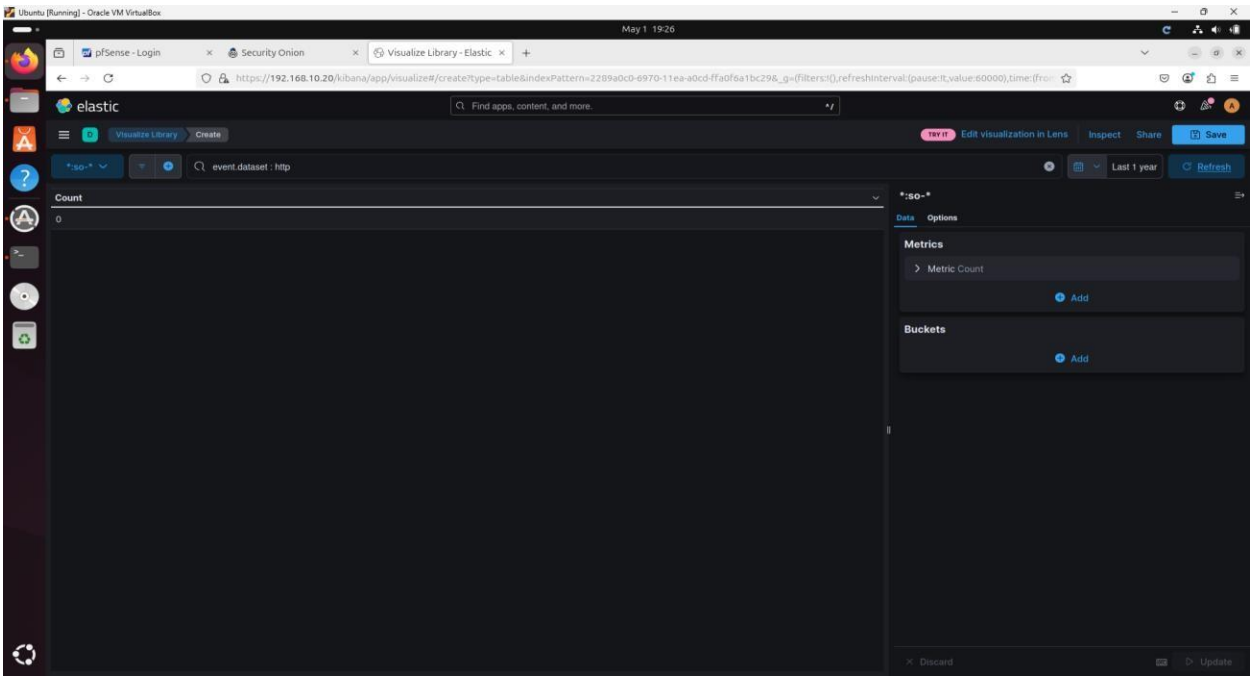


8.11: Downloaded files

1. I attempted to create a **Data Table** using `*:so-*` with the filter `event.dataset:http`.

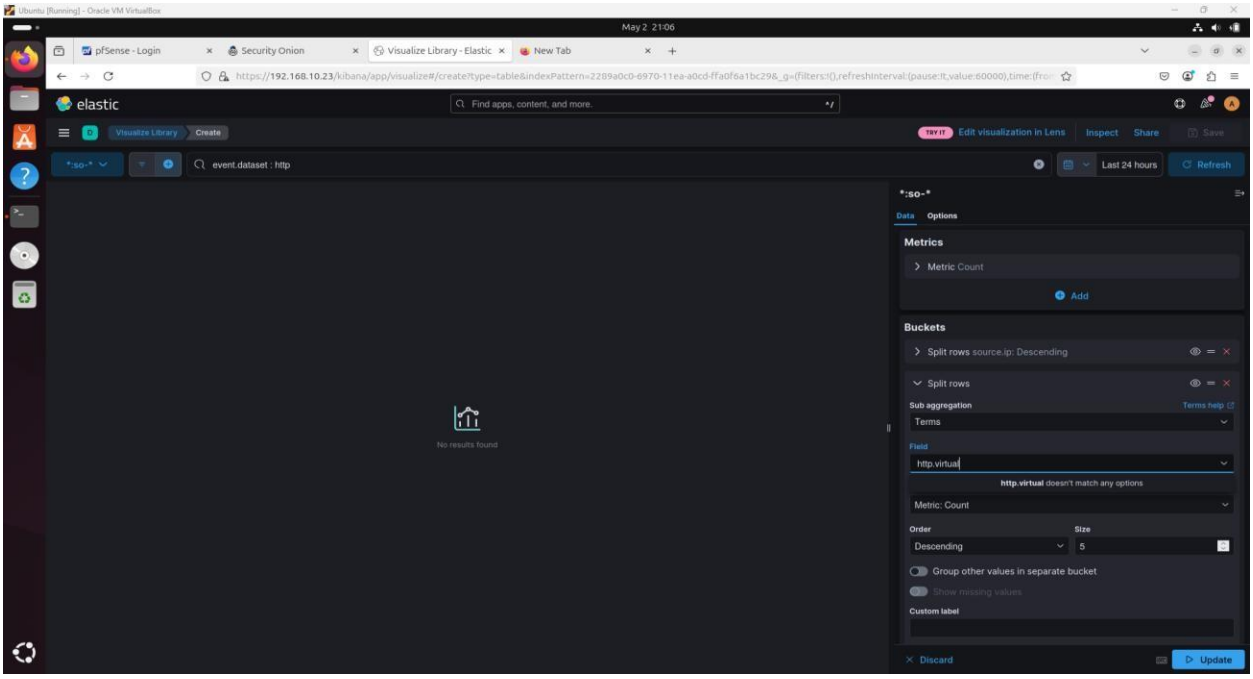


2. I added split rows for `source.ip` and `http.virtual_host.keyword`.



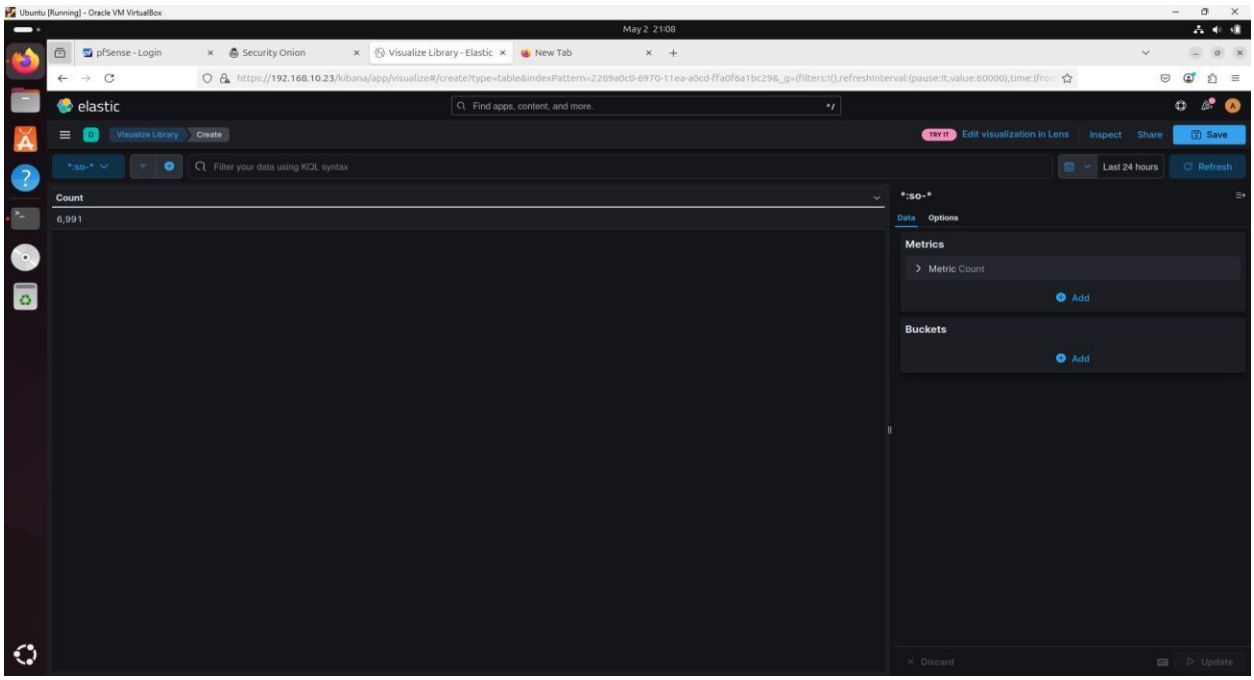
3. Due to a misconfigured Zeek service, HTTP logs were not being collected, so I was unable to complete this visualization.

4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **http.virtual_host.keyword** as the **Field** selection.
   Note: Even after multiple tries, I couldn't get the http logs to be collected since zeek wasn't configured properly. I've tried configuring it saltstack and node.cfg but nothing worked. So skipping this virtualization.

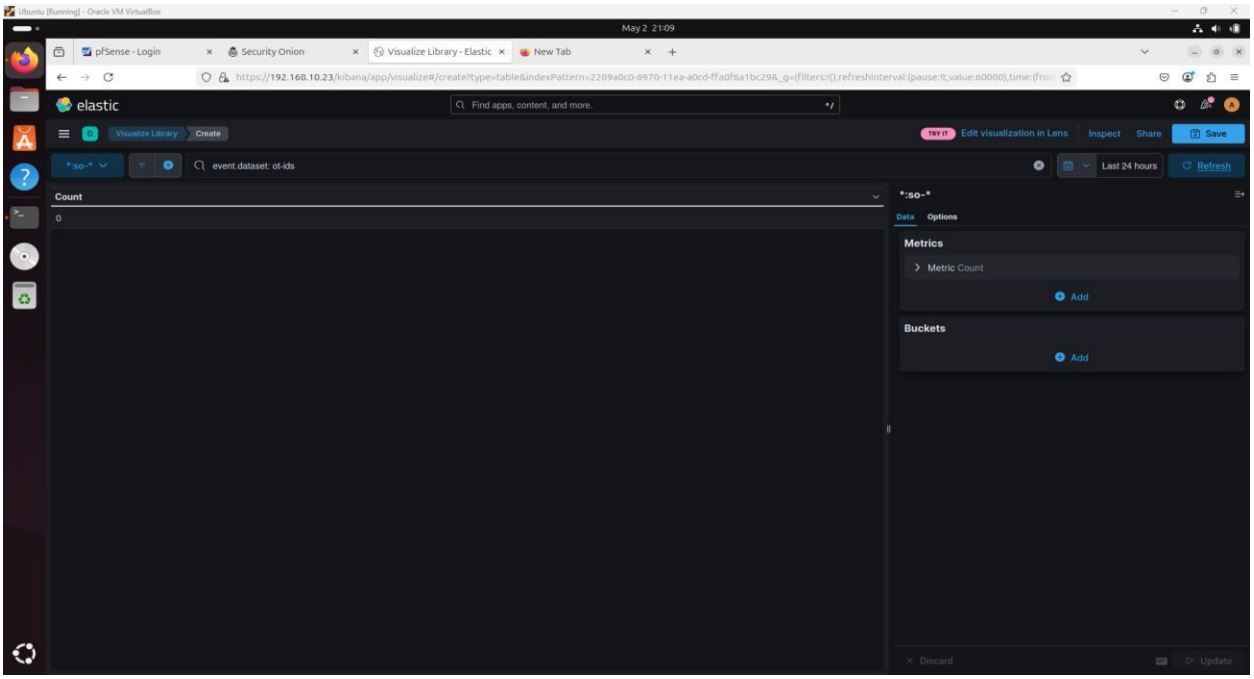


8.12: SilentDefense alerts

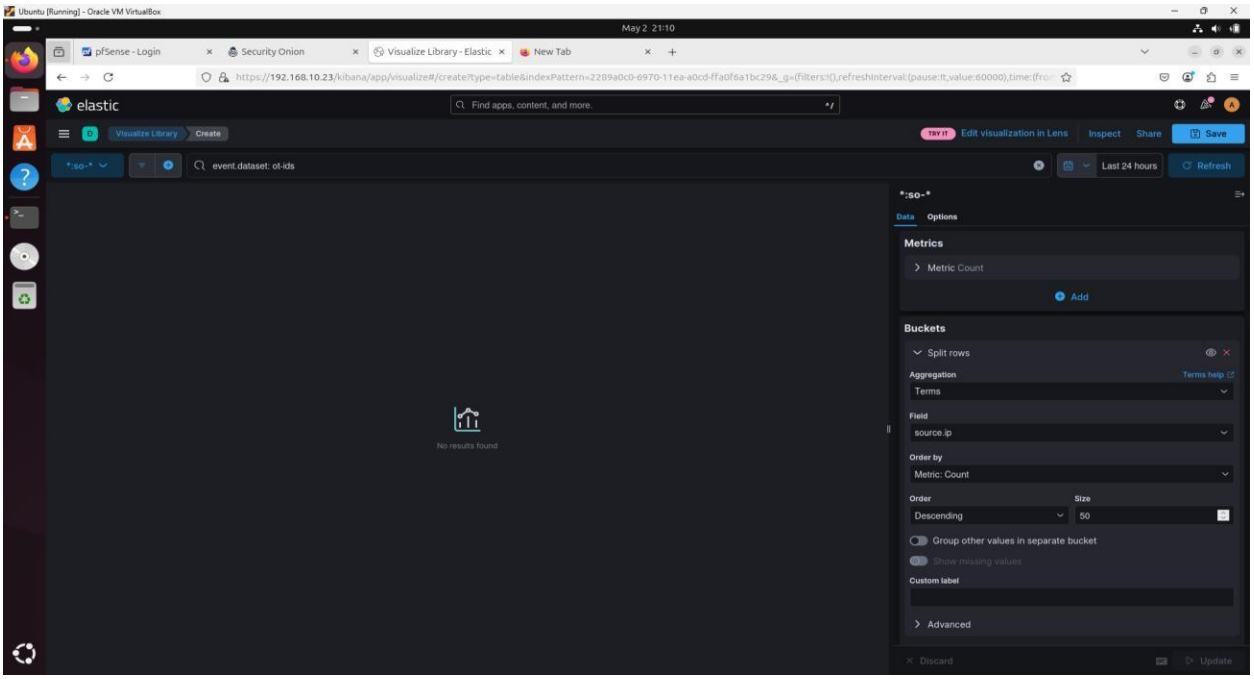
1. I created a **Data Table** using `*:so-*` as the data source and filtered with `event.dataset:ot-ids`.



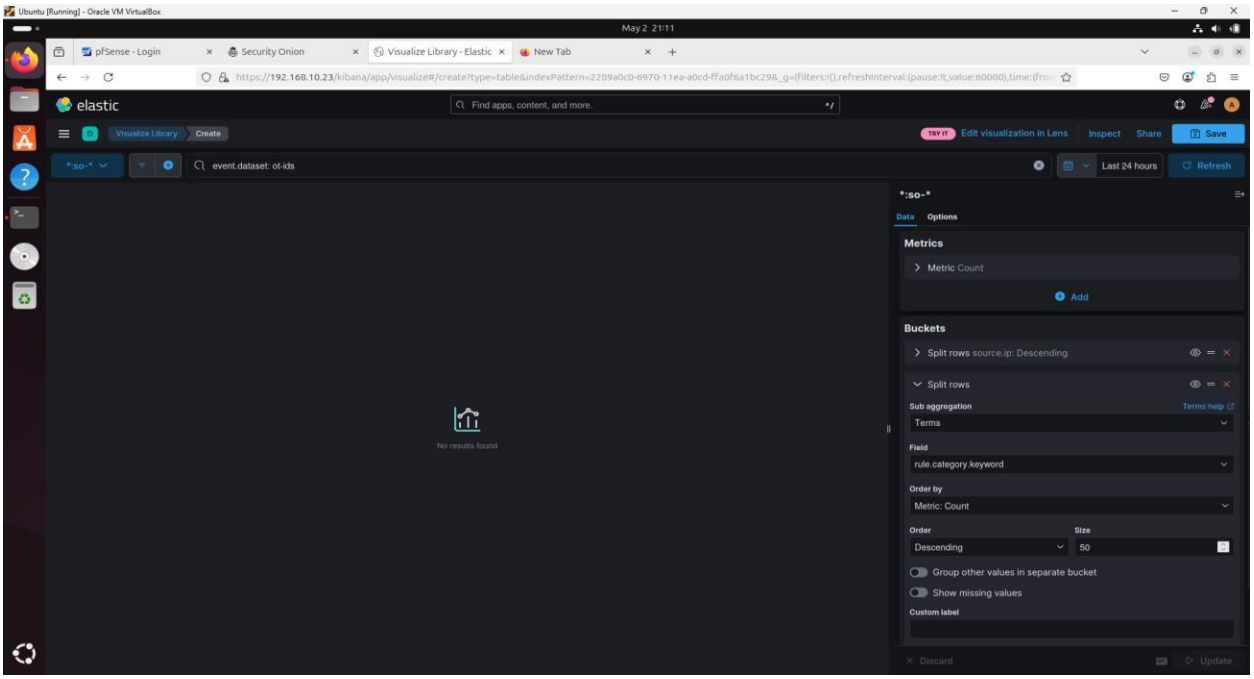
2. Enter an **event.dataset:ot-ids** search term to filter out SilentDefense logs only, then hit **Update.**



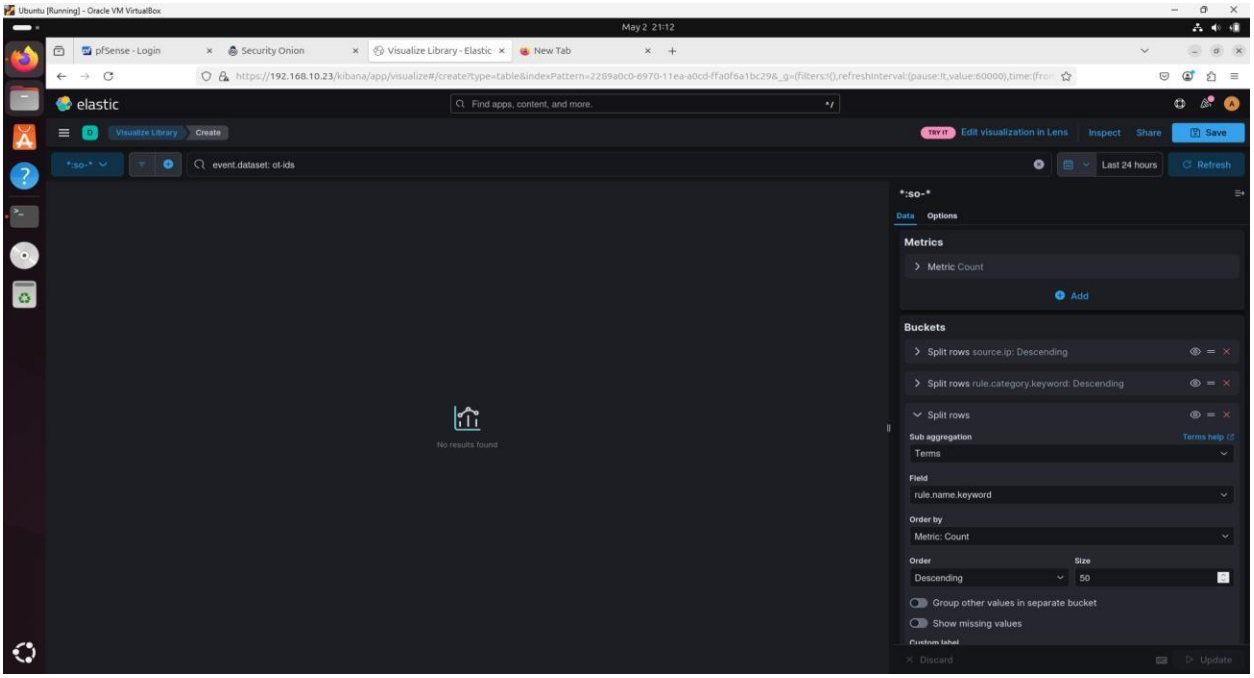
3. I added split rows for `source.ip`, `rule.category.keyword`, and `rule.name.keyword`, each with size 50.



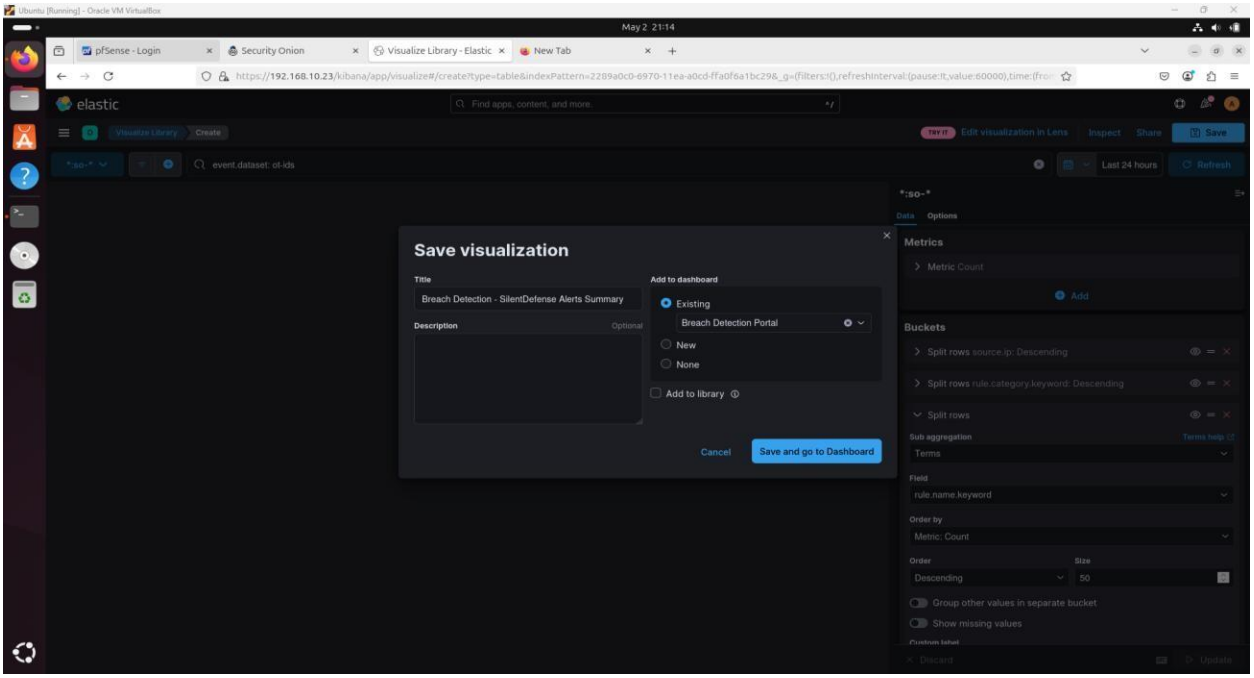
4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **rule.category.keyword** as the **Field** selection. Set the **Size** option to **50**.



5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **rule.name.keyword** as the **Field** selection. Set the **Size** option to **50** and click **Update.**

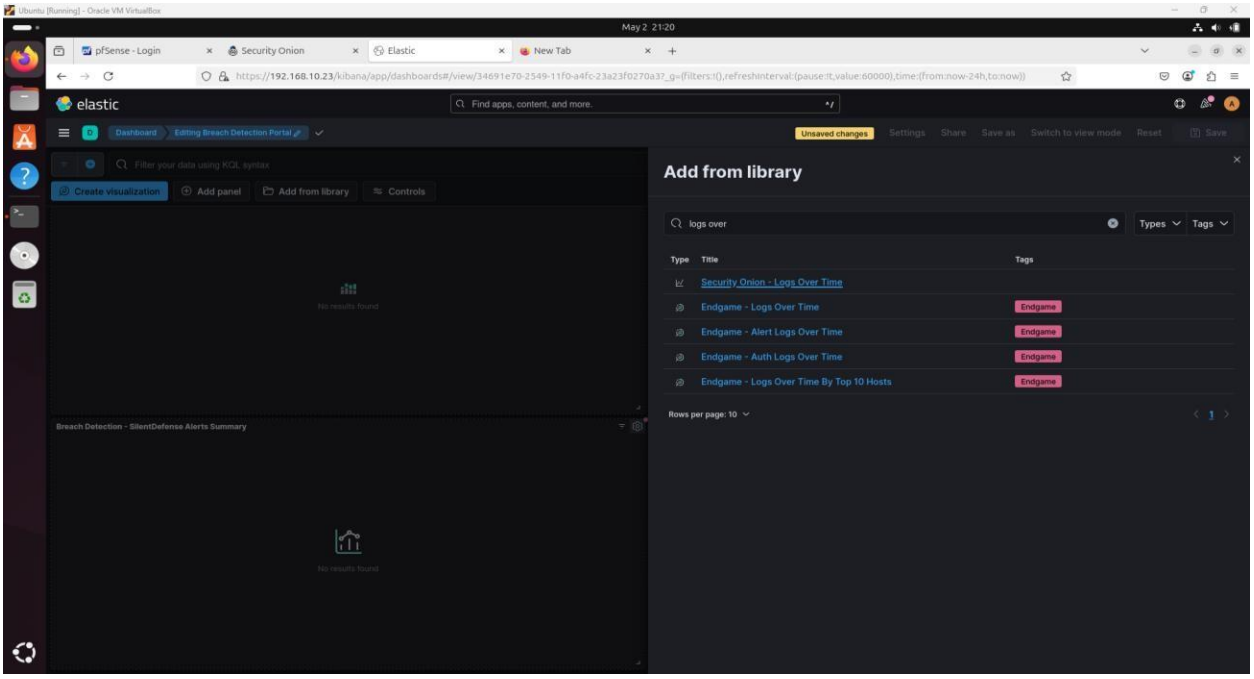


6. I saved the visualization as **Breach Detection – SilentDefense Alerts Summary.**

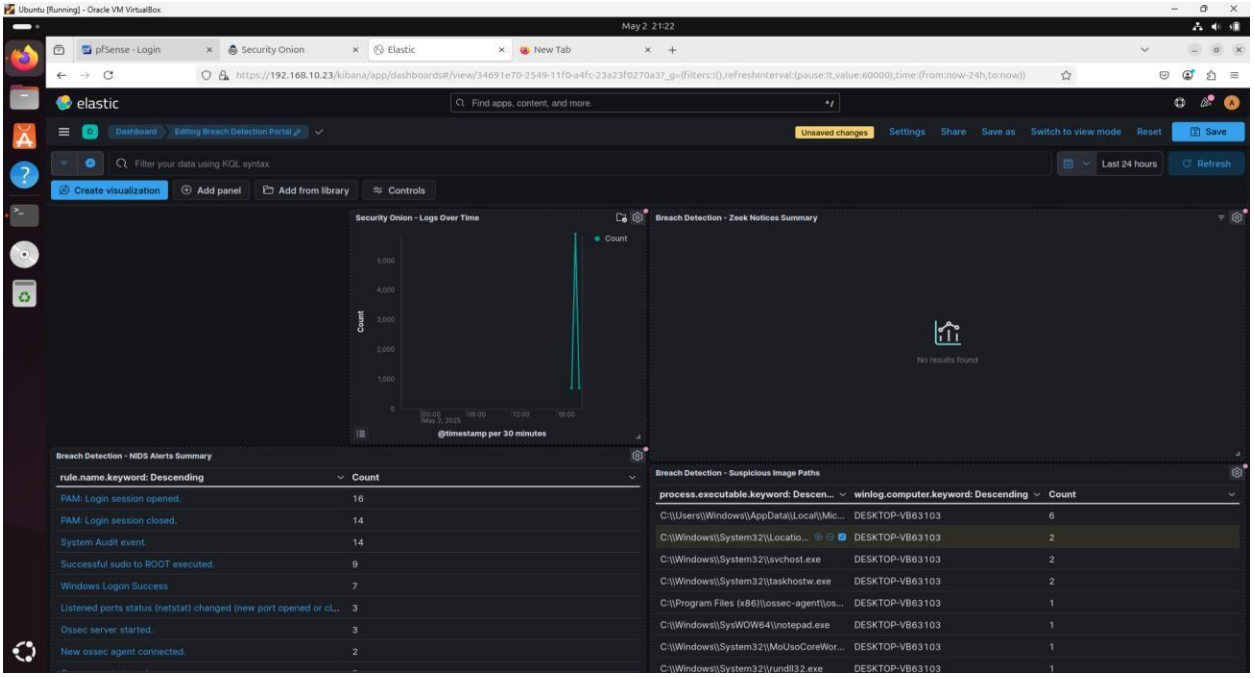


8.13: Finishing up the dashboard

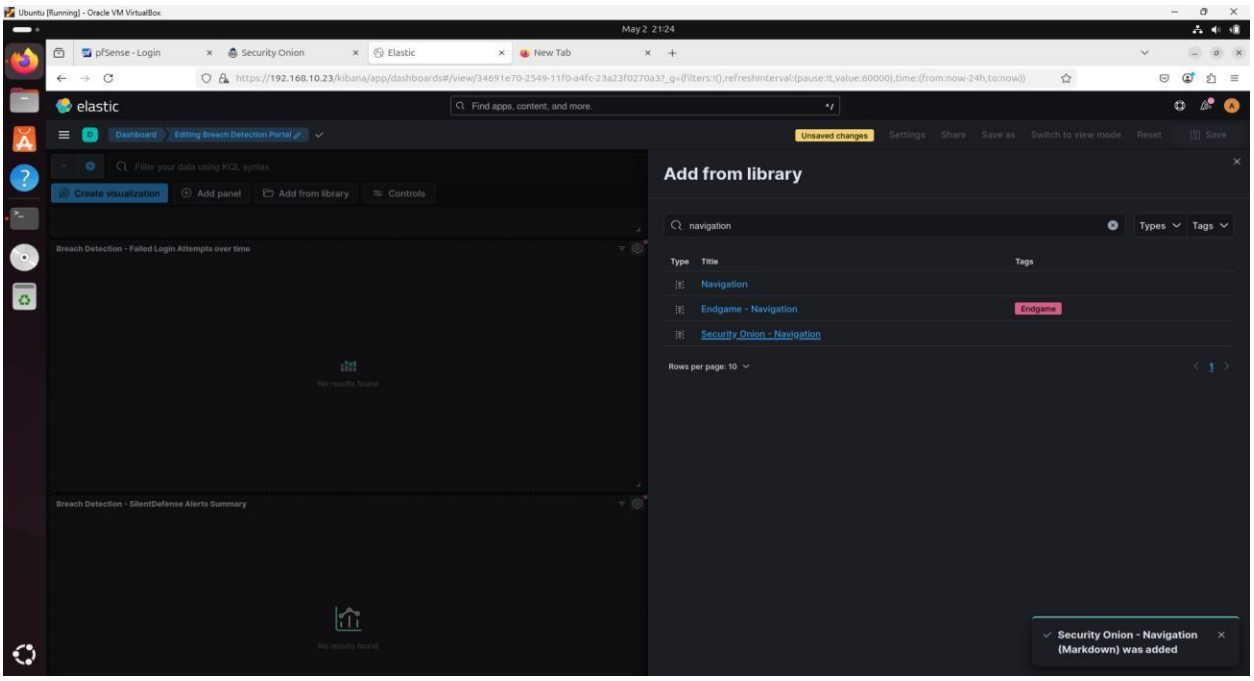
1. I clicked **Add from library** and inserted the **Security Onion – Logs Over Time** visualization.



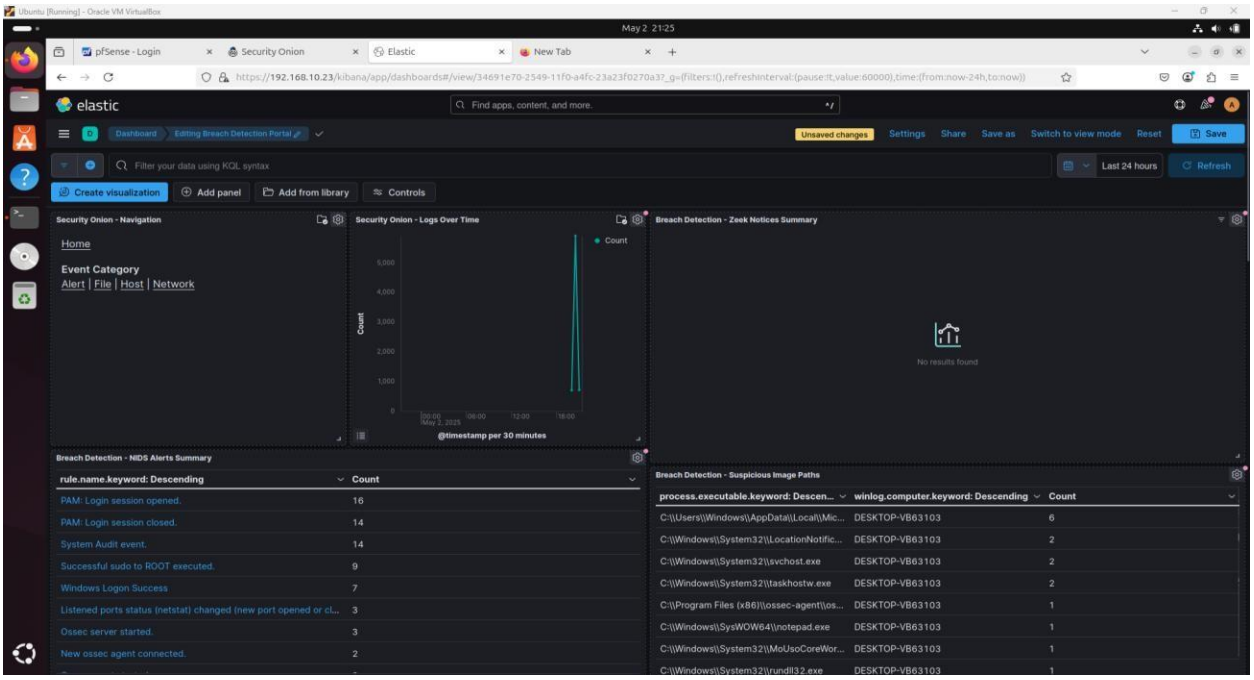
2. I moved this chart to the top center of the dashboard. screen.



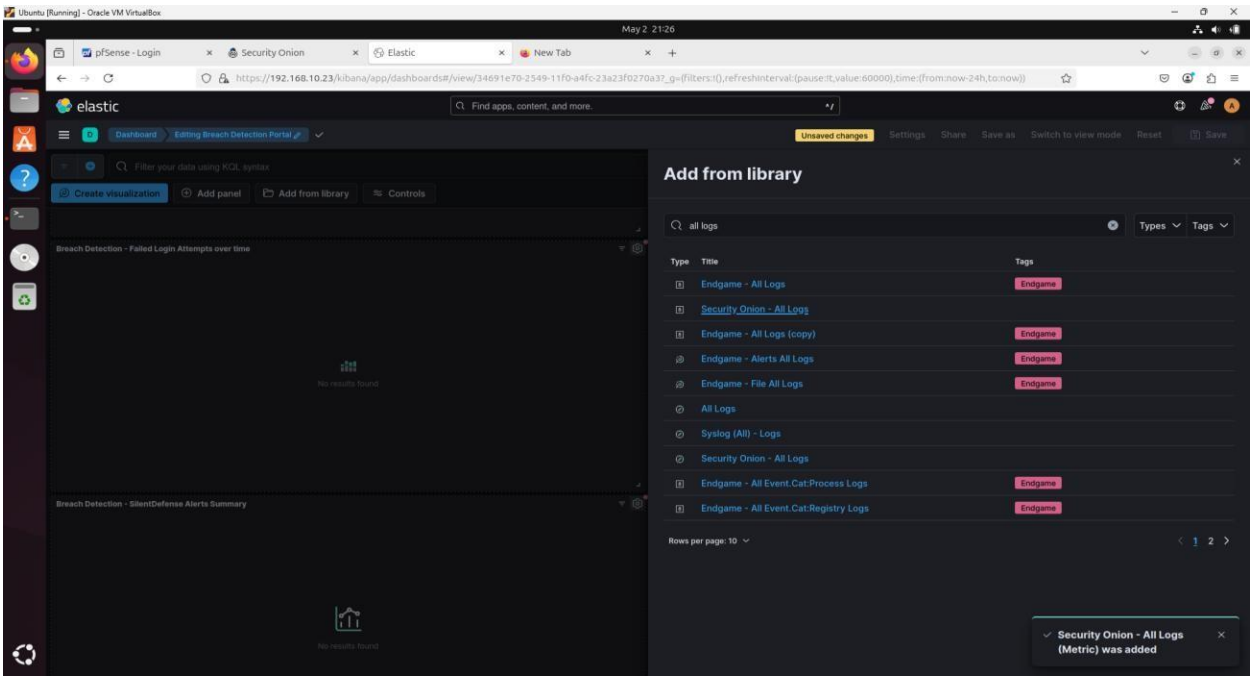
3. I added the **Security Onion – Navigation Panel** and placed it to the top-left.



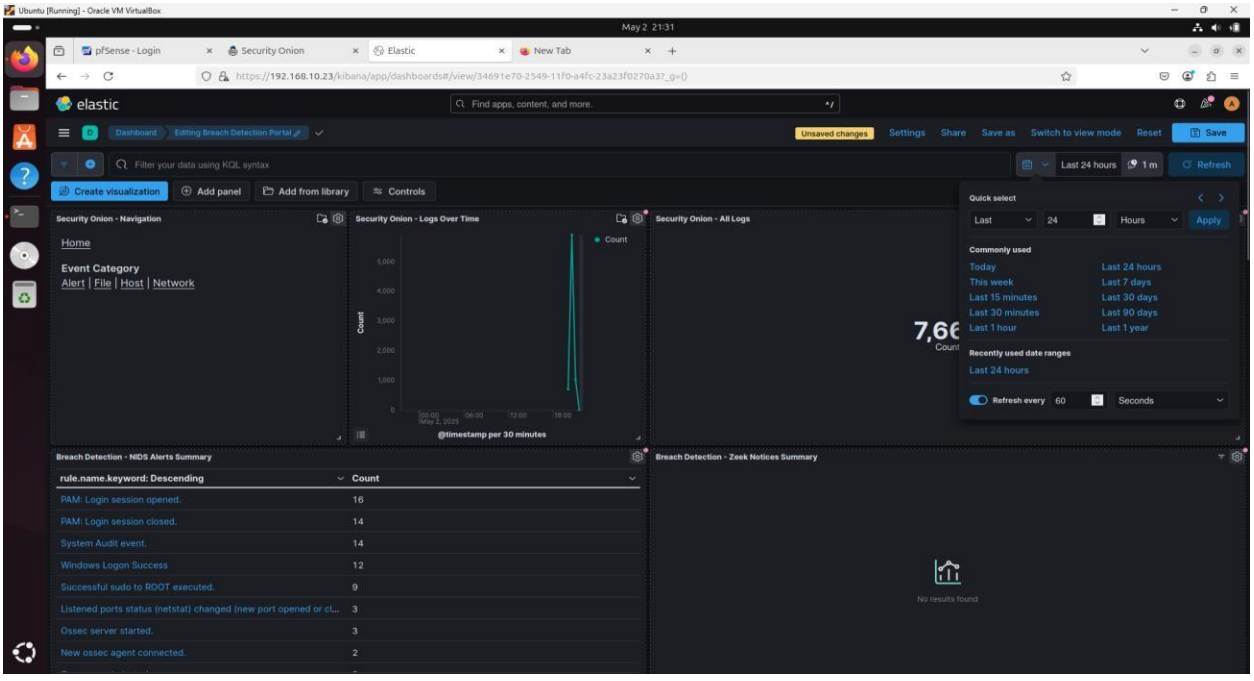
4. Then I added the **Security Onion – All Logs saved search** for deeper context.

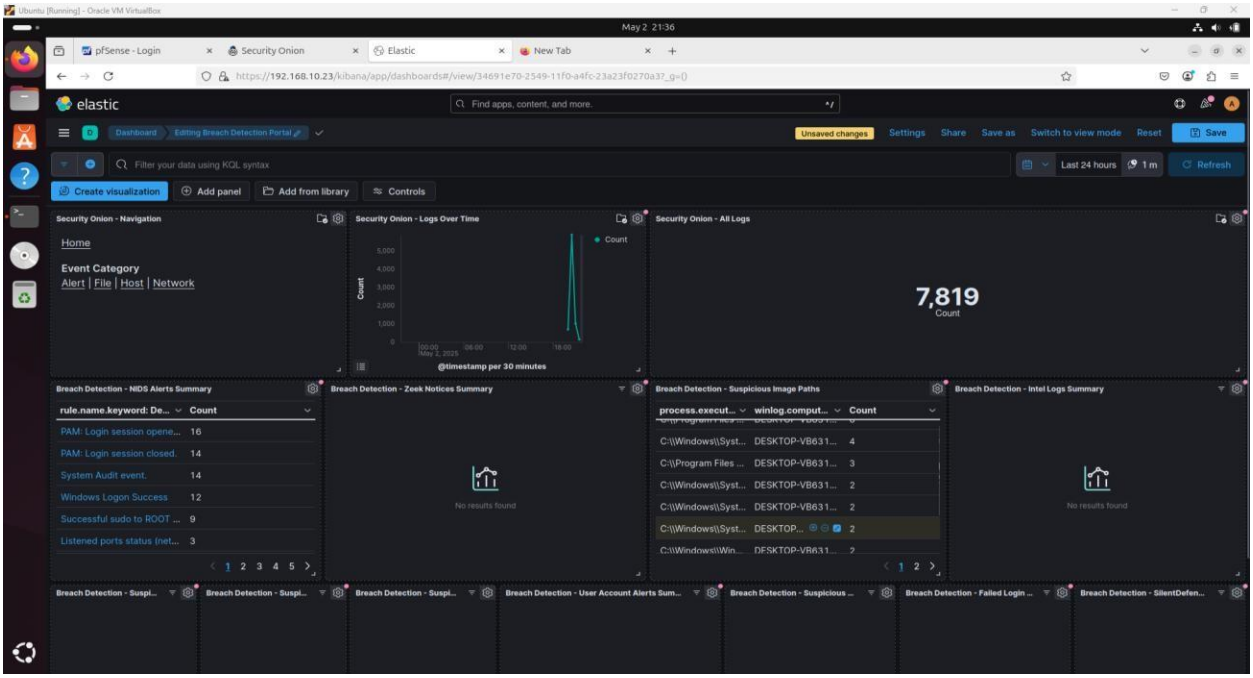


5. I configured the dashboard to show a 24-hour window and refresh every minute.



6. Set the dashboard to show a 24-hour timespan and automatically refresh every minute. Once you are happy with how things look, save the dashboard.





7. After reviewing the layout and confirming it looked clean and functional, I saved the full dashboard setup.

This dashboard can function as a starter to get a **security operations center (SOC)** view in place. This will likely change, so add and modify the dashboard to make it fit your needs, but this is a great starting point for your journey into a holistic security monitoring approach.

NAME: Phanindhar Reddy Karnati
ID: 001667635