

Lab – Mini SIEM

Instructor: Tesfaye W. Lemma

Mini-SIEM Lab: Windows Log Collection and Detection Using the Elastic Stack on VirtualBox

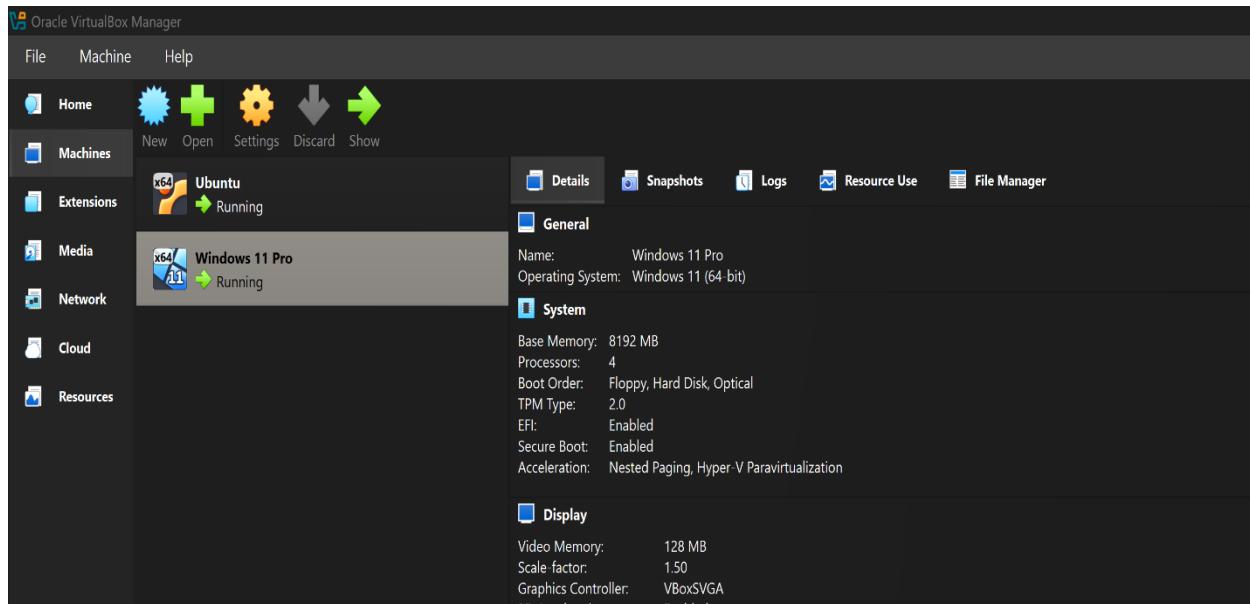
This project consists of implementing a mini-SIEM in a controlled lab environment using two virtual machines in Oracle VM VirtualBox. An Ubuntu virtual machine was configured as the monitoring server running Elasticsearch and Kibana, while a Windows 11 Pro virtual machine acted as the endpoint generating Windows Security logs. The primary objective was to validate the ability to collect Windows logs, centralize them in a search and analytics backend, visualize activity through dashboards, and create basic detections for security-relevant events such as user logons and privilege escalation.

Basic instructions were provided in class by the instructor, and the remaining configuration and troubleshooting were completed independently.

Throughout the project, real-world challenges related to service configuration, virtual networking, and hardware limitations were encountered and resolved, closely reflecting scenarios commonly faced in SOC and security operations environments.

Ubuntu and Windows 11 Pro on Oracle VM Virtual Box

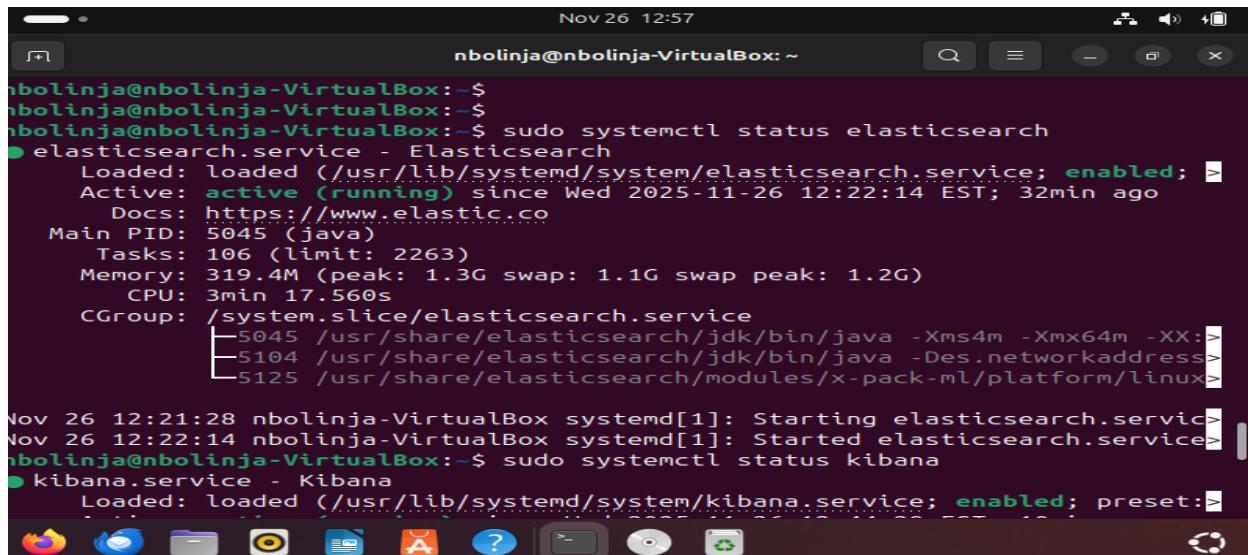
Ubuntu VM and Windows 11 Pro VM in Oracle VM VirtualBox; used to build a mini-SIEM for detection and log monitoring



Guest Additions: Installed on ubuntu.

A screenshot of a terminal window titled 'nborinja@nborinja-VirtualBox:~'. The terminal shows the output of several commands: `lsmod | grep -E 'vbox(guest|sf|video)'` (listing kernel modules), `modinfo vboxguest 2>/dev/null | head` (showing module information), `systemctl status vboxadd-service 2>/dev/null || true` (status of the service), and `lsmod | grep -E 'vbox(guest|sf|video)'` again. It also shows the configuration of the `vboxsf` and `vboxguest` modules, including their filenames, versions, authors, and descriptions. Finally, it shows the logs from `systemctl` for the `vboxadd-service` service, indicating it was loaded and failed to start.

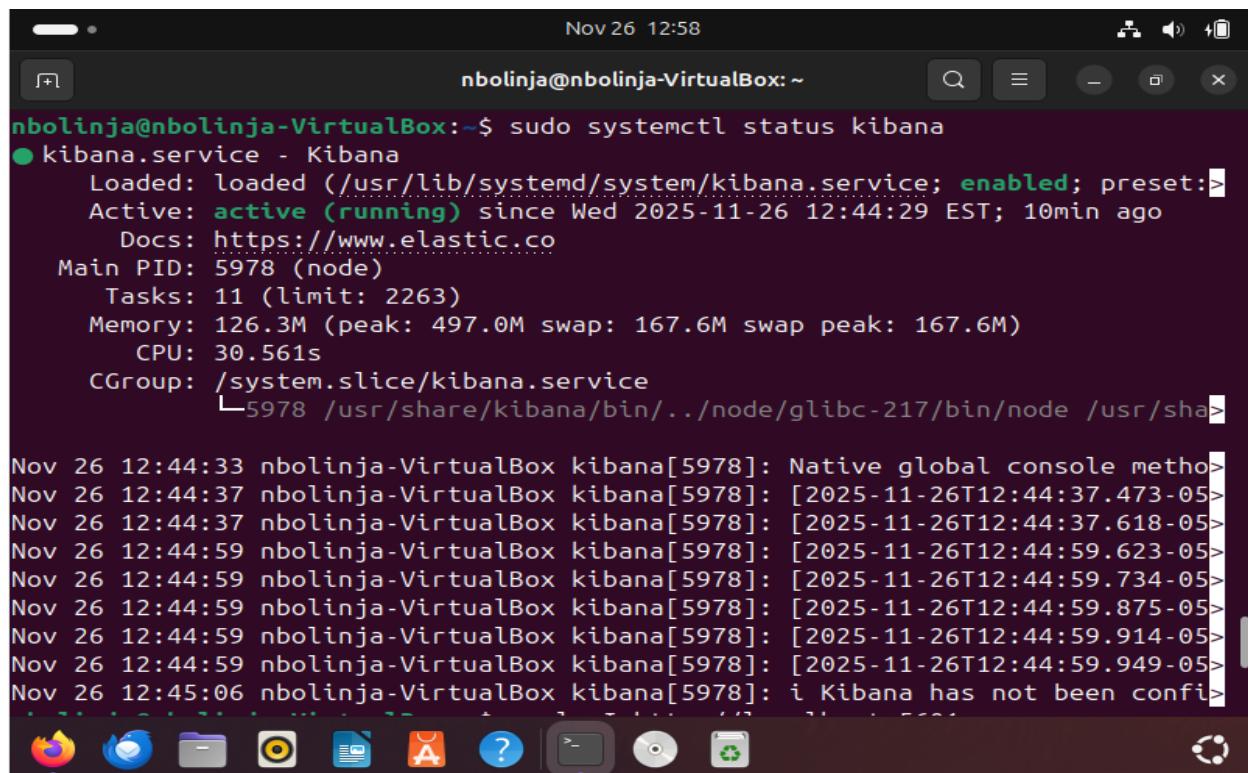
Elasticsearch: service is installed and running (active; running) on Ubuntu.



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays the output of several commands run by the user 'nbolinja'. The commands include navigating to the root directory (~), checking the status of the Elasticsearch service using 'sudo systemctl status elasticsearch', and then checking the status of the Kibana service using 'sudo systemctl status kibana'. The terminal shows that the Elasticsearch service is active and running, and the Kibana service is also active and running. The desktop interface includes a dock at the bottom with icons for various applications like a browser, file manager, and system tools.

```
nbolinja@nbolinja-VirtualBox:~$  
nbolinja@nbolinja-VirtualBox:~$ sudo systemctl status elasticsearch  
● elasticsearch.service - Elasticsearch  
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; preset:)  
   Active: active (running) since Wed 2025-11-26 12:22:14 EST; 32min ago  
     Docs: https://www.elastic.co  
 Main PID: 5045 (java)  
    Tasks: 106 (limit: 2263)  
   Memory: 319.4M (peak: 1.3G swap: 1.1G swap peak: 1.2G)  
     CPU: 3min 17.560s  
    CGroup: /system.slice/elasticsearch.service  
           ├─5045 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:  
           ├─5104 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress:  
           └─5125 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux  
  
Nov 26 12:21:28 nbolinja-VirtualBox systemd[1]: Starting elasticsearch.servic>  
Nov 26 12:22:14 nbolinja-VirtualBox systemd[1]: Started elasticsearch.servic>  
nbolinja@nbolinja-VirtualBox:~$ sudo systemctl status kibana  
● kibana.service - Kibana  
   Loaded: loaded (/usr/lib/systemd/system/kibana.service; enabled; preset:)  
   Active: active (running) since Wed 2025-11-26 12:44:29 EST; 10min ago  
     Docs: https://www.elastic.co  
 Main PID: 5978 (node)  
    Tasks: 11 (limit: 2263)  
   Memory: 126.3M (peak: 497.0M swap: 167.6M swap peak: 167.6M)  
     CPU: 30.561s  
    CGroup: /system.slice/kibana.service  
           └─5978 /usr/share/kibana/bin/../node/glibc-217/bin/node /usr/sha
```

Kibana validation: service is running (active; running) on Ubuntu.



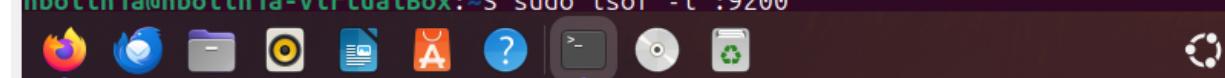
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays the output of several commands run by the user 'nbolinja'. The commands include navigating to the root directory (~), checking the status of the Kibana service using 'sudo systemctl status kibana', and then checking the status of the Elasticsearch service using 'sudo systemctl status elasticsearch'. The terminal shows that the Kibana service is active and running, and the Elasticsearch service is also active and running. The desktop interface includes a dock at the bottom with icons for various applications like a browser, file manager, and system tools.

```
nbolinja@nbolinja-VirtualBox:~$ sudo systemctl status kibana  
● kibana.service - Kibana  
   Loaded: loaded (/usr/lib/systemd/system/kibana.service; enabled; preset:)  
   Active: active (running) since Wed 2025-11-26 12:44:29 EST; 10min ago  
     Docs: https://www.elastic.co  
 Main PID: 5978 (node)  
    Tasks: 11 (limit: 2263)  
   Memory: 126.3M (peak: 497.0M swap: 167.6M swap peak: 167.6M)  
     CPU: 30.561s  
    CGroup: /system.slice/kibana.service  
           └─5978 /usr/share/kibana/bin/../node/glibc-217/bin/node /usr/sha  
  
Nov 26 12:44:33 nbolinja-VirtualBox kibana[5978]: Native global console method>  
Nov 26 12:44:37 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:37.473-05>  
Nov 26 12:44:37 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:37.618-05>  
Nov 26 12:44:59 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:59.623-05>  
Nov 26 12:44:59 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:59.734-05>  
Nov 26 12:44:59 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:59.875-05>  
Nov 26 12:44:59 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:59.914-05>  
Nov 26 12:44:59 nbolinja-VirtualBox kibana[5978]: [2025-11-26T12:44:59.949-05>  
Nov 26 12:45:06 nbolinja-VirtualBox kibana[5978]: i Kibana has not been config>
```

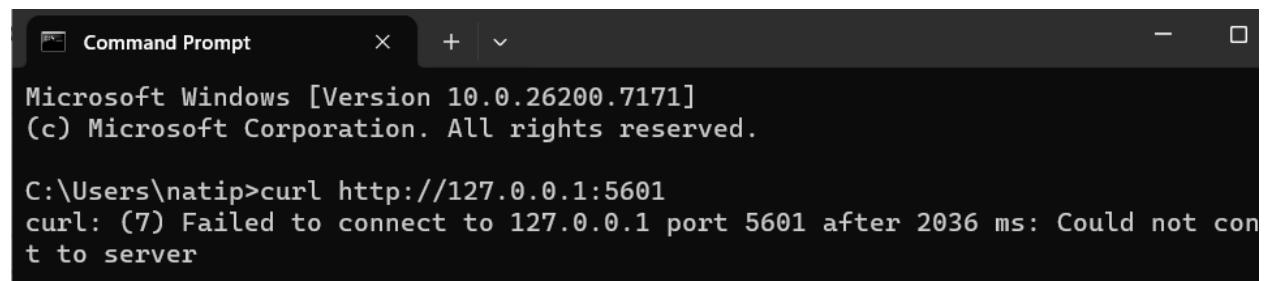
Kibana local reachability: localhost:5601 responds successfully; confirms the UI endpoint is up on the Ubuntu VM.

```
nbolinja@nbolinja-VirtualBox:~$ curl -I http://localhost:5601
HTTP/1.1 200 OK
x-content-type-options: nosniff
referrer-policy: strict-origin-when-cross-origin
permissions-policy: camera=(), display-capture=(), fullscreen=(self), geolocation=(), microphone=(), web-share=()
cross-origin-opener-policy: same-origin
content-security-policy: script-src 'report-sample' 'self'; worker-src 'report-sample' 'self' blob;; style-src 'report-sample' 'self' 'unsafe-inline'; object-src 'report-sample' 'none'
content-security-policy-report-only: form-action 'report-sample' 'self'
kbn-name: nbolinja-VirtualBox
content-type: text/html; charset=utf-8
cache-control: private, no-cache, no-store, must-revalidate
content-length: 38441
vary: accept-encoding
Date: Wed, 26 Nov 2025 17:55:47 GMT
Connection: keep-alive
Keep-Alive: timeout=120

nbolinja@nbolinja-VirtualBox:~$ sudo lsof -i :9200
```



Connectivity issue: NAT did not allow Windows 11 to reach Kibana; access failed from the Windows VM.

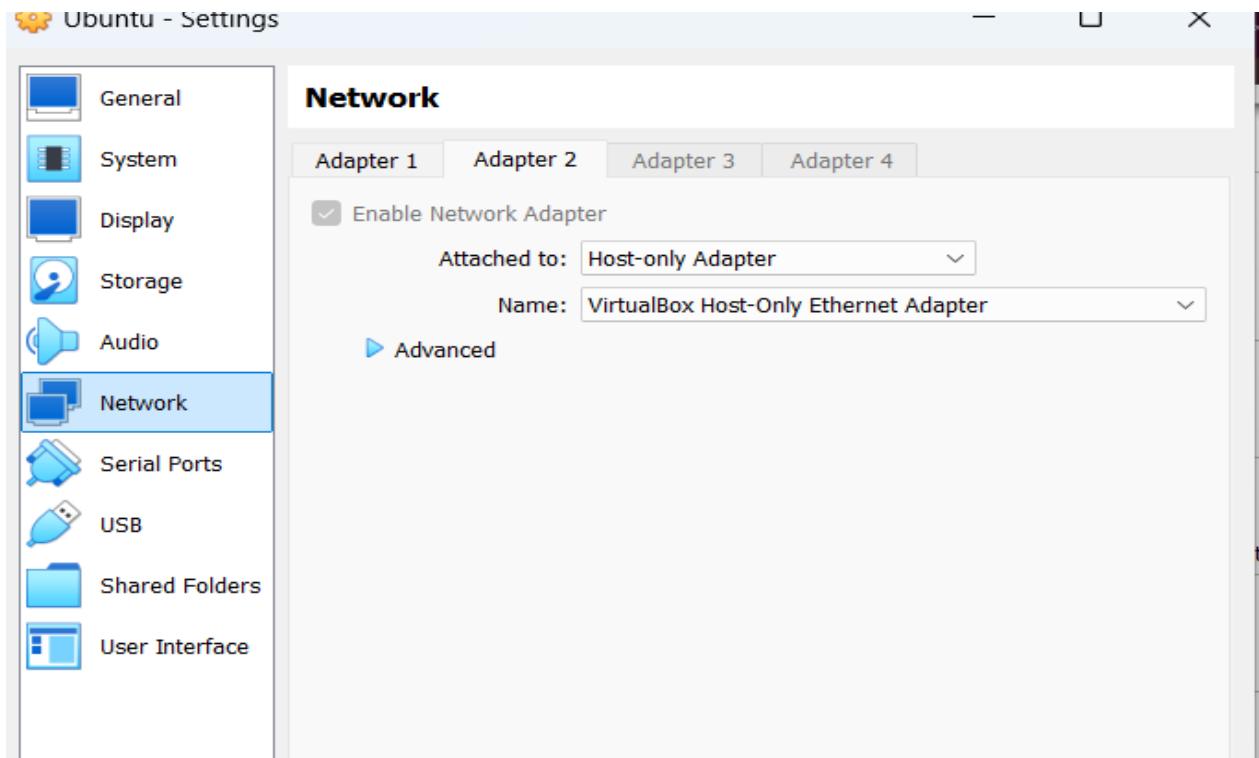


```
Command Prompt
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\natip>curl http://127.0.0.1:5601
curl: (7) Failed to connect to 127.0.0.1 port 5601 after 2036 ms: Could not connect to server
```

Fix applied: Kibana became accessible from Windows 11 Pro after switching to a Host-only adapter.

Host-only networking: Adapter 2 set to Host-only for direct Ubuntu - Windows communication; 192.168.56.0/24; Kibana reachable at 192.168.56.101 or 192.168.56.10.

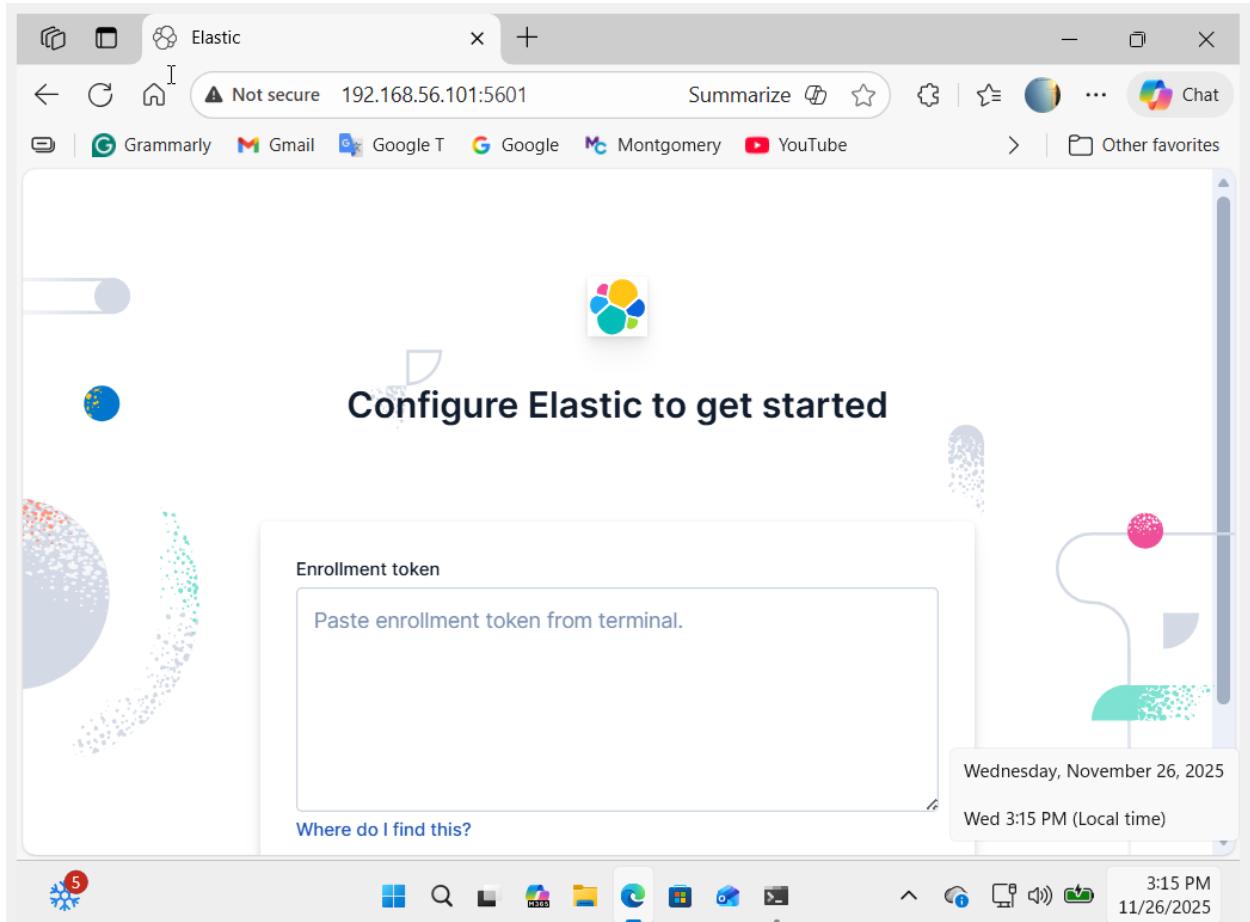


Fleet connectivity: Elastic Agent is running and connected through Fleet.

The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell". The command PS C:\WINDOWS\system32> Start-Service "Elastic Agent" is run, followed by Start-Sleep -Seconds 5. Then PS C:\WINDOWS\system32> Start-Service "Elastic Agent" is run again. The output shows the status of the fleet and elastic-agent services. The fleet status starts as (STARTING) and then becomes (HEALTHY) Running. The elastic-agent status starts as (HEALTHY) Running and then becomes (HEALTHY) Connected. Finally, the status is checked again, showing the fleet as (HEALTHY) Connected and the elastic-agent as (HEALTHY) Running. A watermark for "Activate Windows" is visible in the bottom right corner of the window.

```
PS C:\WINDOWS\system32> Start-Service "Elastic Agent"
PS C:\WINDOWS\system32> Start-Sleep -Seconds 5
PS C:\WINDOWS\system32> Start-Service "Elastic Agent"
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> $EA = "$env:ProgramFiles\Elastic\Agent\elastic-agent.exe"
>> & $EA status
  fleet
    └ status: (STARTING)
      elastic-agent
        └ status: (HEALTHY) Running
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> & $EA status
  fleet
    └ status: (STARTING)
      elastic-agent
        └ status: (HEALTHY) Running
PS C:\WINDOWS\system32> & $EA status
  fleet
    └ status: (HEALTHY) Connected
      elastic-agent
        └ status: (HEALTHY) Running
PS C:\WINDOWS\system32> & $EA status
  fleet
    └ status: (HEALTHY) Connected
      elastic-agent
        └ status: (HEALTHY) Running
PS C:\WINDOWS\system32>
```

Enrollment token: token used to enroll/register the Elastic Agent in Fleet from the Elastic website workflow.

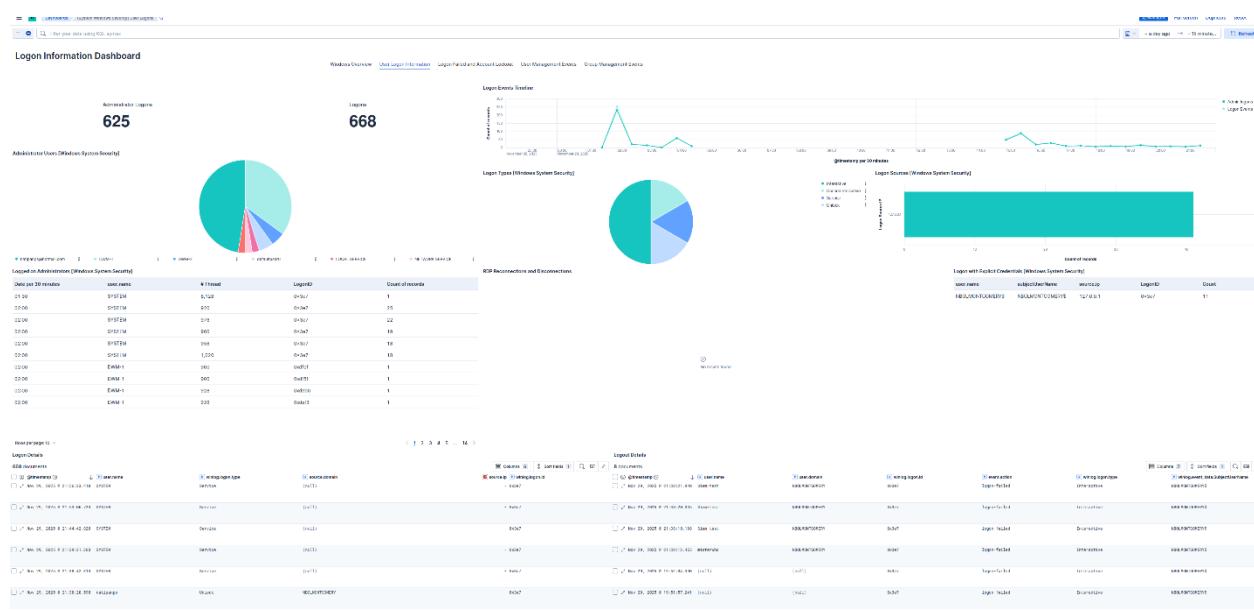
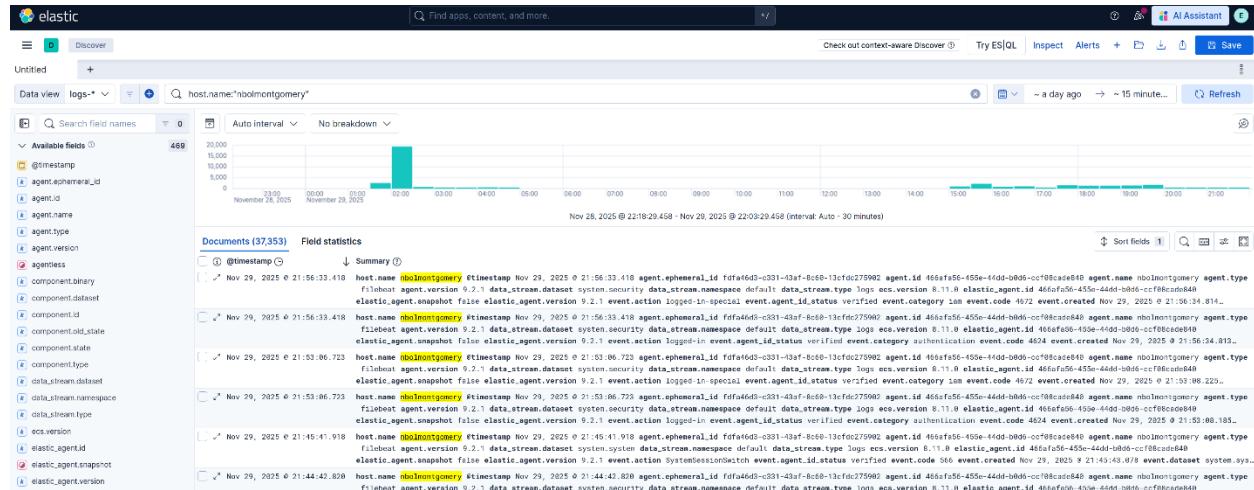


Fleet agents inventory: both endpoints are listed and Healthy; Windows host uses “Windows-Logs” policy; Ubuntu host uses the Fleet Server policy.

The screenshot shows the Elastic Fleet Agents page for a Windows host. The top navigation bar includes the elastic logo, a search bar, and tabs for 'Fleet' and 'Agents'. Below the search bar are filters for 'Status' (5), 'Tags' (1), 'Agent policy' (2), and an 'Upgrade available' button. The main table displays two healthy agents: 'nbolmontgomery' (Windows-Logs policy) and 'nbolinja-VirtualBox' (Fleet Server policy). Both agents are marked as healthy with a green status indicator. The table columns include Status, Host, Agent policy, CPU, Memory, and Last active. A message at the bottom right encourages activating Windows. The bottom navigation bar includes the elastic logo, a search bar, and tabs for 'Fleet' and 'Agents'. The taskbar at the bottom shows various pinned icons and the system clock at 4:12 PM on 11/29/2025.

The screenshot shows the Elastic Fleet Agents page for an Ubuntu host. The interface is identical to the Windows host's, with the same navigation bar, filters, and table structure. The two agents listed are 'nbolmontgomery' (Windows-Logs policy) and 'nbolinja-VirtualBox' (Fleet Server policy). Both are marked as healthy. The table columns and bottom message are identical. The bottom navigation bar and taskbar are also present, showing the elastic logo, search bar, and pinned icons, along with the system clock at 4:12 PM on 11/29/2025.

Windows logon dashboard: (System Windows Security) User Logons summarizes Windows logon events received from host nbolmontgomery.



Detection alert: alert created for event.code 4672 to demonstrate privileged-elevation detection in the mini-SIEM.

Edit rule settings

Rule preview

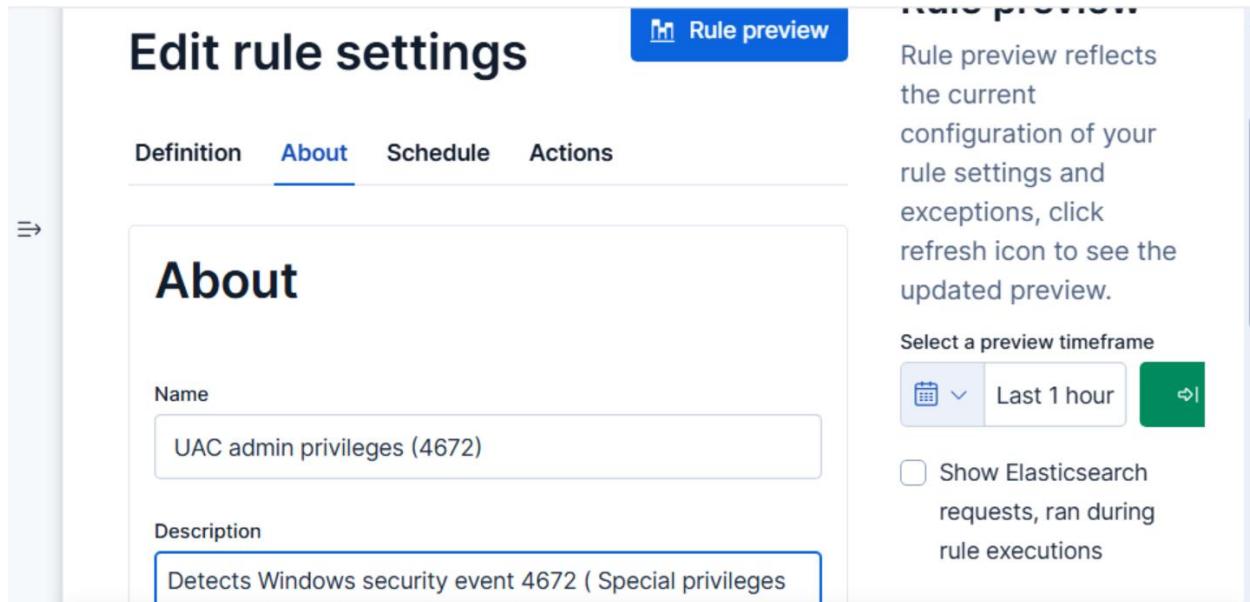
Rule preview reflects the current configuration of your rule settings and exceptions, click refresh icon to see the updated preview.

Select a preview timeframe

Name: UAC admin privileges (4672)

Description: Detects Windows security event 4672 (Special privileges)

Show Elasticsearch requests, ran during rule executions



Detection rules (SIEM) - Kibana

Not secure 192.168.56.10:5601/app/security/rules/id/ff169...

elasticsearch Find apps, content, and more. AI Assistant

ML job settings Add integrations Data view Security solution default

Filter your data using KQL syntax Today

UAC admin privileg...

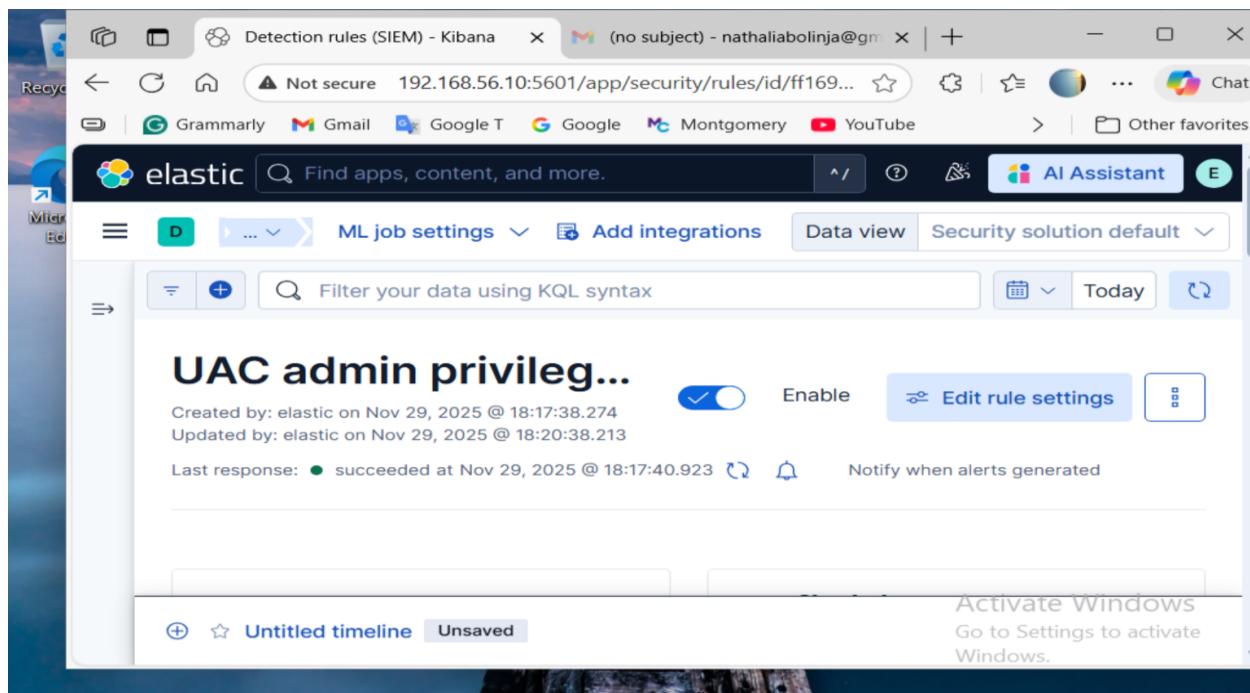
Created by: elastic on Nov 29, 2025 @ 18:17:38.274
Updated by: elastic on Nov 29, 2025 @ 18:20:38.213

Last response: succeeded at Nov 29, 2025 @ 18:17:40.923 Notify when alerts generated

Enable Edit rule settings

Untitled timeline Unsaved

Activate Windows
Go to Settings to activate Windows.



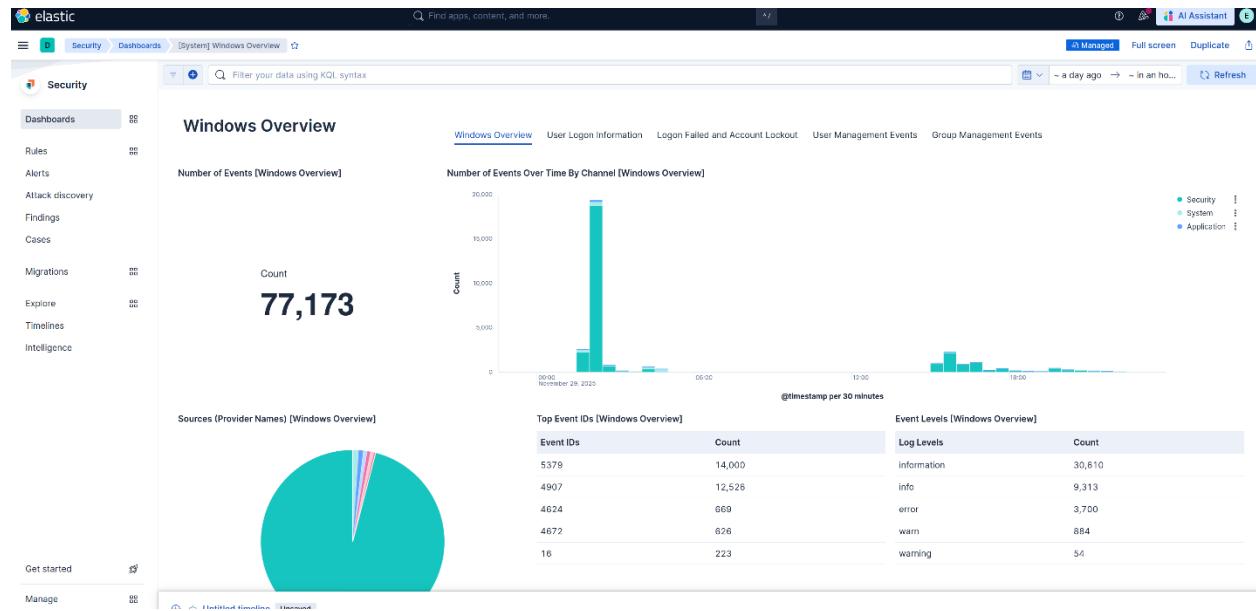
Hosts view: Elastic Security summary of observed hosts, unique IPs, and log-volume over time for the selected period.



“Uncommon processes” is simply a list of Windows processes (programs/services) that ran on the system and, within the time range you selected, appeared less frequently compared to what is typical for that host.

Process name	Hosts	Instances	Host names	Last command	Last user
WerFault.exe	1	2	nbolmontgomery	—	—
Registry	1	7	nbolmontgomery	—	—
autochk.exe	1	7	nbolmontgomery	—	—
lsass.exe	1	7	nbolmontgomery	—	—
services.exe	1	7	nbolmontgomery	—	—
wininit.exe	1	7	nbolmontgomery	—	—
winlogon.exe	1	7	nbolmontgomery	—	—
csrss.exe	1	14	nbolmontgomery	—	—
smss.exe	1	21	nbolmontgomery	—	—

Windows Overview: high-level confirmation that Windows logs are being ingested; shows event volume, top event IDs (4624; 4672), and providers before deeper analysis in Discover.



Conclusion

For this project, I built a mini-SIEM lab using two virtual machines in Oracle VM VirtualBox. The Ubuntu VM runs Elasticsearch and Kibana, and the Windows 11 Pro VM generates Windows Security logs so I could confirm that my system can collect data, display it in dashboards (Kibana), and trigger detections.

Elasticsearch is the backend that receives and stores the Windows logs. It indexes the data so users can fast search, filter, and correlate events during an investigation. While Kibana provides the analysis and visualization tool. It connects to Elasticsearch and displays the collected logs in a user-friendly interface, allowing exploration in *Discover*, dashboard creation, and trend monitoring through charts and metrics (for example, highlighting spikes in failed logon attempts). Using Kibana, I was able to visually confirm log activity, including how many times a user entered the wrong password (failed logon attempts) and test basic detections/alerts based on specific Windows Security Event IDs. However, before getting to this point, I had to install Elasticsearch on Ubuntu and confirmed it was working by checking that the service was active (running). I also installed Kibana and verified it was active (running) and reachable locally at localhost:5601, which confirmed the web interface was available for analysis.

During the integration step, I ran into a networking issue: with NAT enabled, my Windows VM could not reach Kibana. To fix this, I added a Host-only adapter so the Ubuntu and Windows VMs could communicate directly on the 192.168.56.0/24 network. After that change, I was able to access Kibana from Windows using the Ubuntu Host-only IP.

The integration phase was challenging due to hardware limitations. My computer wasn't powerful enough to run these heavy tools reliably, so I ended up buying a more powerful machine to finish the project.

After I fixed the earlier issues, I moved on to the integration phase. From the Windows VM, I signed in to Kibana with my credentials and used Fleet to enroll and manage the Elastic Agents. During this step, Kibana generated a ready-to-run command to install and configure the Fleet Server on the Ubuntu VM. After I ran the command on Ubuntu, I checked Fleet and confirmed that two agents were enrolled and both were reporting Healthy. The Windows host (nbolmontgomery) was assigned the Windows-Logs policy, and the Ubuntu host (nbolinja-VirtualBox) was assigned the Fleet Server policy, confirming end-to-end communication between both endpoints and the Fleet Server.

Finally, I validated the results in Elastic Security. The (System Windows Security) User Logons dashboard showed logon activity from the Windows host in the selected time range. The Hosts and Windows Overview pages also confirmed ingestion volume and important Windows event IDs like 4624 (successful logon) and 4672 (special privileges). To demonstrate detection, I created an alert rule for event.code 4672 to flag privileged-elevation activity as part of the mini-SIEM deliverable.

In conclusion, Elasticsearch and Kibana are powerful tools, and they become even more effective when used together. The project was very challenging because there were many extra configurations needed to get the entire system working end to end. Even so, it was absolutely worth it. I also tried to explore more Elasticsearch and Kibana features, but I was limited because some capabilities require a premium subscription. It's impressive how a small set of tools can give you the ability to build a home security monitoring

environment that can collect logs, visualize activity, and detect suspicious behavior on your systems.