Cyberprotection Systems

Laboratory Work 1. Event Gathering and Correlation

(MEMORY)

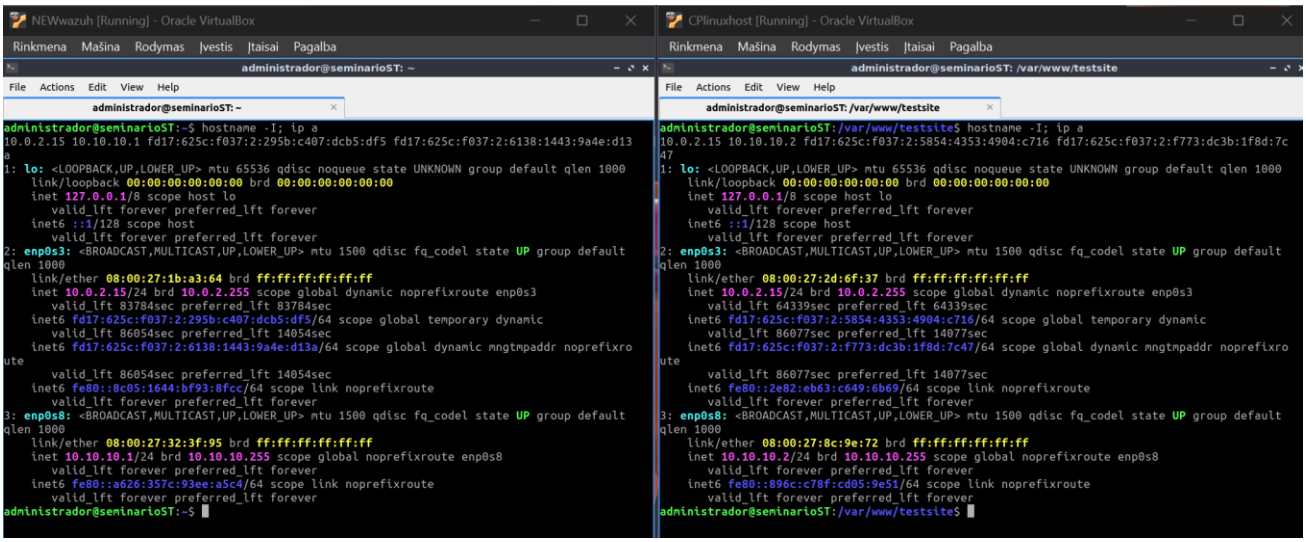
NAME AND SURNAME: Austėja Bauraitė

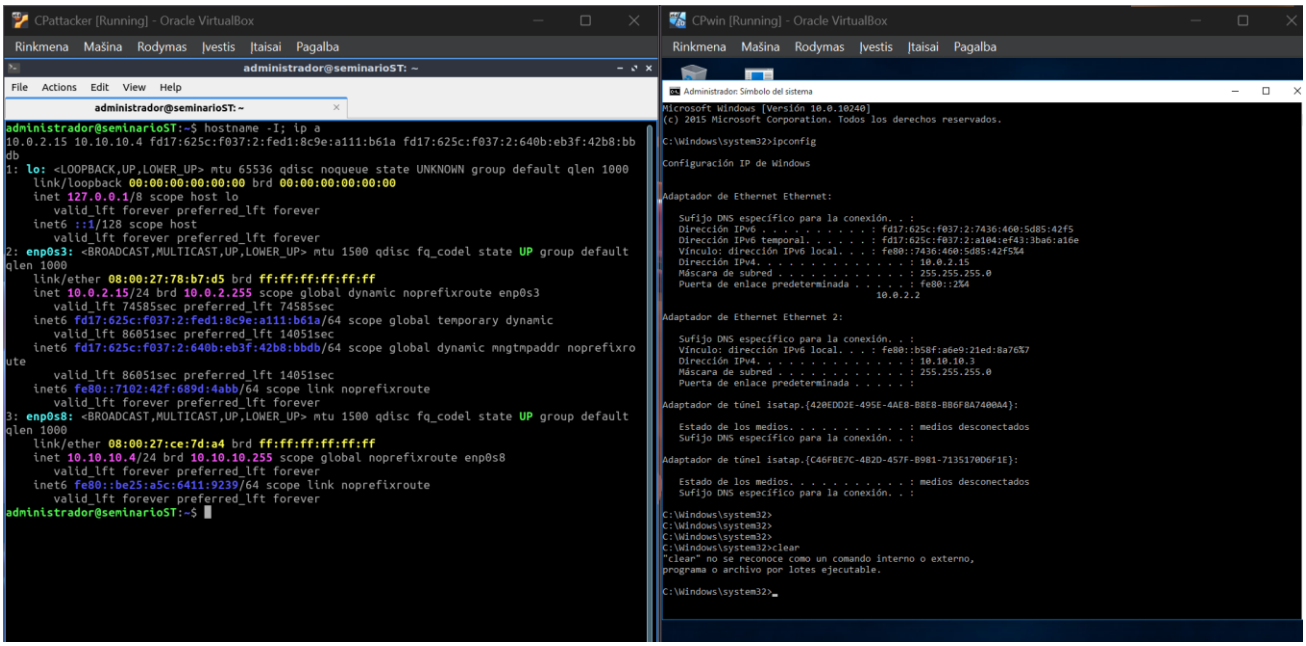
1. Topology configuration

Describe the configured topology (machines, O.S), as well as the technologies used (VirtualBox, Dockers, etc).

The lab runs a simple, flat network with four virtual machines, a Wazuh manager , a Linux endpoint where Suricata is installed alongside the Wazuh agent, plus two additional VMs to play the roles of user/attacker and generate traffic (e.g., nmap) toward LinuxHost. Suricata inspects the network interface on LinuxHost and logs alerts to /var/log/suricata/eve.json, which the Wazuh agent forwards to the manager for correlation and visualization, configuration points include setting HOME_NET, default-rule-path, and enabling rule-files in suricata.yaml. The whole setup is virtualized (in VirtualBox) and uses Suricata with the ET Open ruleset and Wazuh for SIEM/alerting, same as the lab guide.

Put a screenshot (or some) to show them running.

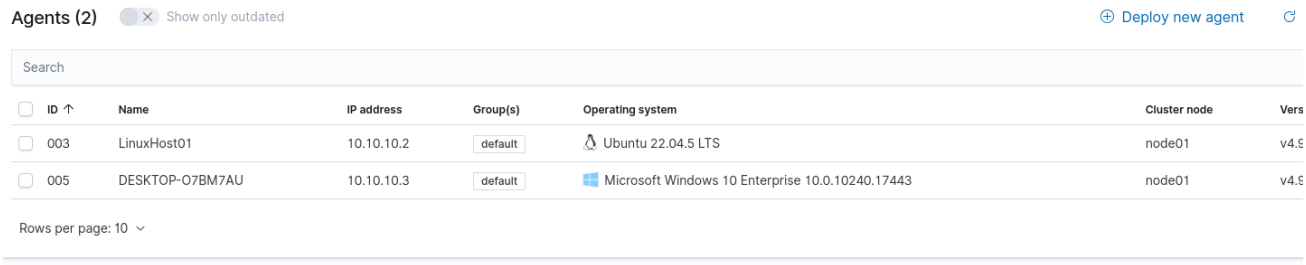




2. Wazuh setting

Show a screenshot (or some) where it can be seen the deployed agents in Wazuh. Explain it.

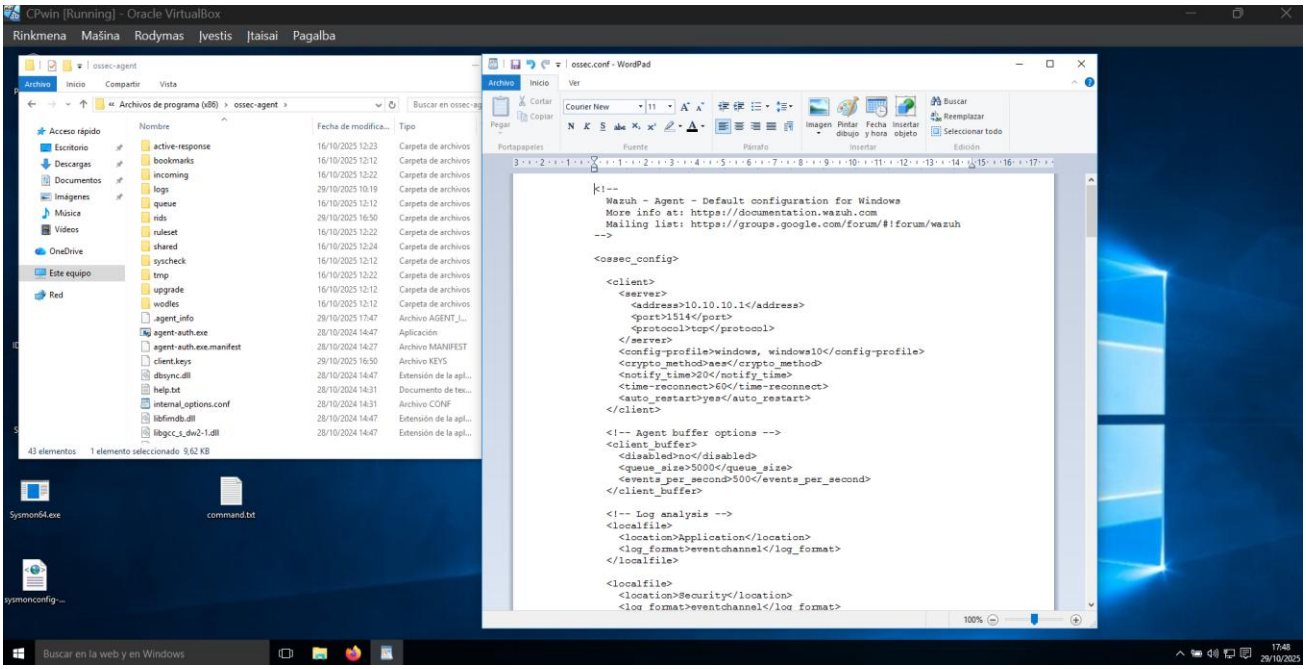
This Wazuh “Agents” view shows the two endpoints that are actively enrolled with the manager. Agent **003 (LinuxHost01)** at **10.10.10.2** is an **Ubuntu 22.04.5 LTS** host in the **default** group, and agent **005 (DESKTOP-O7BM7AU)** at **10.10.10.3** is a **Windows 10 Enterprise** host, also in **default**. Both are reporting to cluster node **node01** with the same agent major version, registered, and communicating so their logs and security events can be collected and shown in dashboards.



3. Wazuh agents

Show a screenshot (or some) with the configuration files of the agents.

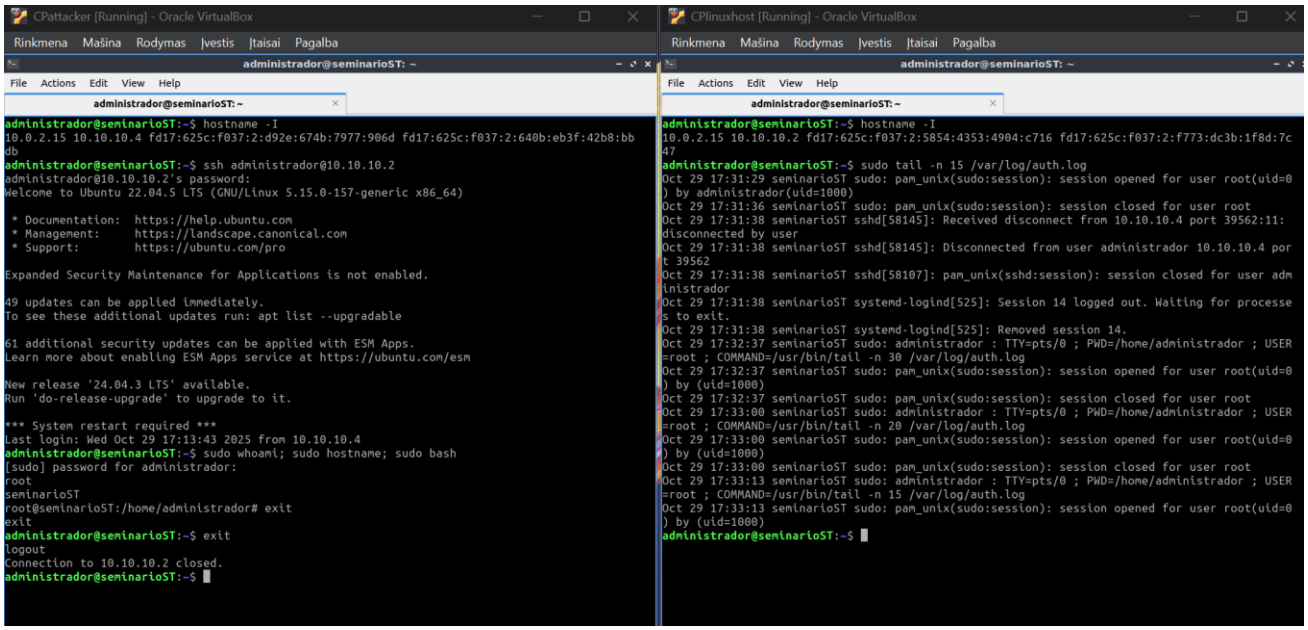
```
administrador@seminarioST:~$ sudo cat  /var/ossec/etc/ossec.conf
<!--
  Wazuh - Agent - Configuration for Ubuntu 22.04
  This version collects classic syslog files (auth.log + syslog) and avoids
  duplicate ingestion from journald. It also keeps FIM, SCA, and Syscollector enabled.
  Replace only the content of /var/ossec/etc/ossec.conf with this block.
-->

<ossec_config>

  <!-- ========================
       Agent -> Manager settings
       ======================== -->
  <client>
    <server>
      <address>10.10.10.1</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>

    <!-- Profile tags are fine to keep -->
    <config-profile>ubuntu, ubuntu22, ubuntu22.04</config-profile>

    <notify_time>10</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>

    <!-- Enrollment (leave enabled until the agent is enrolled) -->
    <enrollment>
      <enabled>yes</enabled>
      <agent_name>LinuxHost01</agent_name>
      <!-- Absolute path is clearer -->
      <authorization_pass_path>/var/ossec/etc/authd.pass</authorization_pass_path>
    </enrollment>
  </client>

  <client_buffer>
```



4. Event generation

a) Show a screeshot (or some) in which the SSH connection task (from Practical part 1.1) is shown; i.e. present the SSH connection, the actions performed, the log events produced, and the generated events in Wazuh. Explain what it is shown.

An attacker from 10.10.10.2 logged into **LinuxHost01** over SSH as administrador, then ran a few sudo commands (things like whoami). Those actions show up in the host's /var/log/auth.log as SSH accept messages and sudo entries, and the Wazuh agent picked them up and turned them into alerts, you can see the SSH login, the PAM session open/close, and the successful sudo to ROOT executed events in the Wazuh console, complete with the exact commands, the user, the source IP and rule IDs for quick reference.



With the KQL filter:

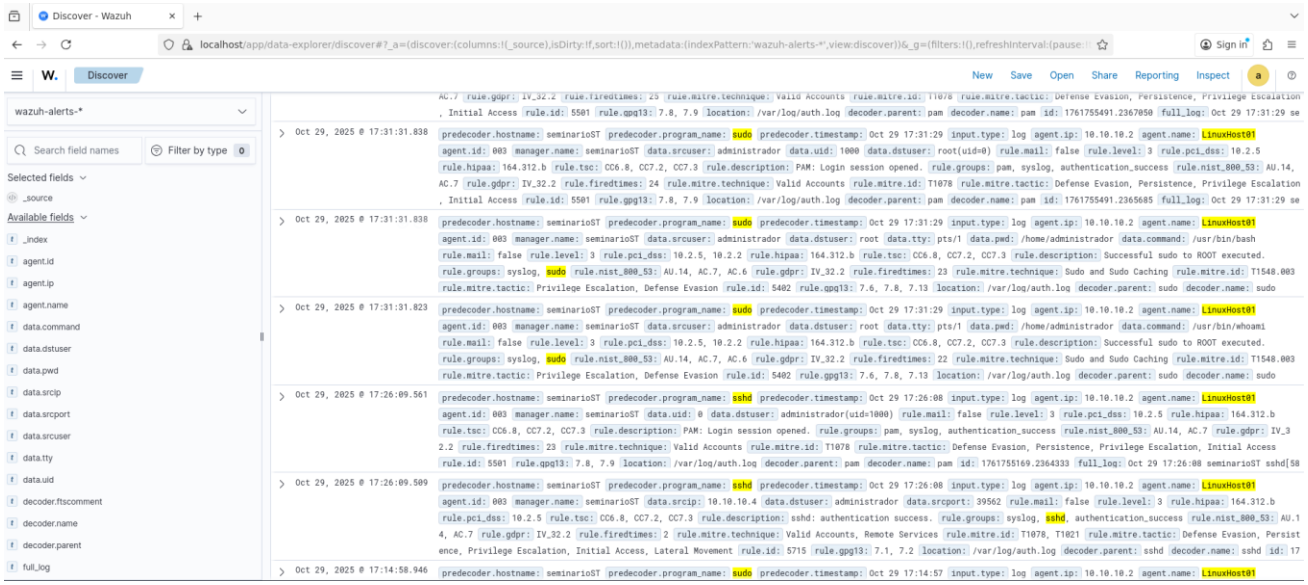
agent.name:"LinuxHost01" and (

rule.groups : ("sshd" or "sudo")

or predecoder.program_name : (sshd or sudo)

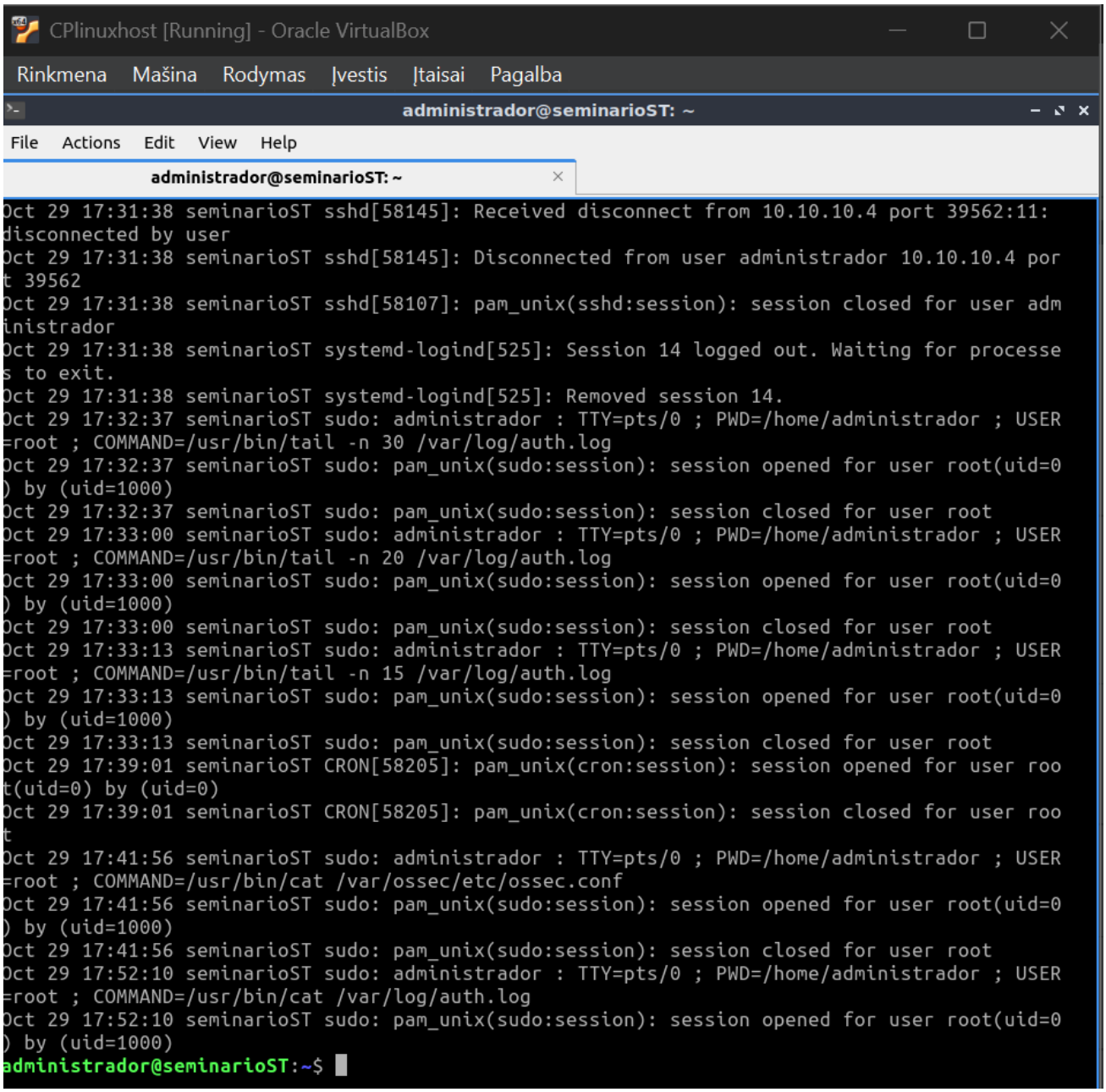
)

and not agent.id:"000"



b) Relate at least three events from the dashboard to the corresponding lines in the auth.log file.

In the auth.log file, we can clearly see the SSH session and the actions that followed, which match the alerts captured in Wazuh. At **17:26:08**, the log records the SSH authentication success “sshd: authentication success” for user *administrador* coming from 10.10.10.2. This corresponds directly to the Wazuh event with **rule ID 5715**, where the agent LinuxHost01 reports a successful SSH login. Just after that, the log shows the PAM session opening message (“pam_unix(sshd:session): session opened for user administrador”), which matches the Wazuh alert with **rule ID 5501**, describing the start of an SSH session. Later, at **17:31:38**, the log shows the session closing (“pam_unix(sshd:session): session closed for user administrador”) this is reflected in Wazuh by the **rule ID 5502** event that marks the termination of that session. Together, these correlated entries show the full SSH activity chain: a successful login, session initiation, and proper logout, all detected by Wazuh and tied back to the same timestamps and user actions recorded in /var/log/auth.log.


CPlinuxhost [Running] - Oracle VirtualBox
Rinkmena Mašina Rodymas Įvestis Įtaisai Pagalba
administrador@seminarioST: ~
File Actions Edit View Help
administrador@seminarioST: ~
Oct 29 17:31:38 seminarioST sshd[58145]: Received disconnect from 10.10.10.4 port 39562:11: disconnected by user
Oct 29 17:31:38 seminarioST sshd[58145]: Disconnected from user administrador 10.10.10.4 port 39562
Oct 29 17:31:38 seminarioST sshd[58107]: pam_unix(sshd:session): session closed for user administrador
Oct 29 17:31:38 seminarioST systemd-logind[525]: Session 14 logged out. Waiting for processes to exit.
Oct 29 17:31:38 seminarioST systemd-logind[525]: Removed session 14.
Oct 29 17:32:37 seminarioST sudo: administrador : TTY=pts/0 ; PWD=/home/administrador ; USER=root ; COMMAND=/usr/bin/tail -n 30 /var/log/auth.log
Oct 29 17:32:37 seminarioST sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
Oct 29 17:32:37 seminarioST sudo: pam_unix(sudo:session): session closed for user root
Oct 29 17:33:00 seminarioST sudo: administrador : TTY=pts/0 ; PWD=/home/administrador ; USER=root ; COMMAND=/usr/bin/tail -n 20 /var/log/auth.log
Oct 29 17:33:00 seminarioST sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
Oct 29 17:33:00 seminarioST sudo: pam_unix(sudo:session): session closed for user root
Oct 29 17:33:13 seminarioST sudo: administrador : TTY=pts/0 ; PWD=/home/administrador ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/auth.log
Oct 29 17:33:13 seminarioST sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
Oct 29 17:33:13 seminarioST sudo: pam_unix(sudo:session): session closed for user root
Oct 29 17:39:01 seminarioST CRON[58205]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Oct 29 17:39:01 seminarioST CRON[58205]: pam_unix(cron:session): session closed for user root
Oct 29 17:41:56 seminarioST sudo: administrador : TTY=pts/0 ; PWD=/home/administrador ; USER=root ; COMMAND=/usr/bin/cat /var/ossec/etc/ossec.conf
Oct 29 17:41:56 seminarioST sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
Oct 29 17:41:56 seminarioST sudo: pam_unix(sudo:session): session closed for user root
Oct 29 17:52:10 seminarioST sudo: administrador : TTY=pts/0 ; PWD=/home/administrador ; USER=root ; COMMAND=/usr/bin/cat /var/log/auth.log
Oct 29 17:52:10 seminarioST sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
administrador@seminarioST:~$

Time	_source
Oct 29, 2025 @ 17:31:39.835	predecoder.hostname: seminarioST predecoder.program_name: sshd predecoder.timestamp: Oct 29 17:31:38 input.type: log agent.ip: 10.10.10.2 agent.name: LinuxHost01 agent.id: 003 manager.name: seminarioST data.dstuser: administrador rule.firedtimes: 26 rule.mail: false rule.level: 3 rule.pci_dss: 10.2.5 rule.hipaa: 164.312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: PAM: Login session closed. rule.groups: pam, syslog rule.id: 5502 rule.nist_800_53: AU.14, AC.7 rule.gpg13: 7.8, 7.9 rule.gdpr: IV_32.2 location: /var/log/auth.log decoder.parent: pam decoder.name: pam id: 1761755499.2369261 full_log: Oct 29 17:31:38 seminarioST sshd[58107]: pam_unix(sshd:session): session closed for user administrador timestamp: Oct 29, 2025 @ 17:31:39.835 _index: wazuh-alerts-4.x-2025.10.29
Oct 29, 2025 @ 17:26:09.561	predecoder.hostname: seminarioST predecoder.program_name: sshd predecoder.timestamp: Oct 29 17:26:08 input.type: log agent.ip: 10.10.10.2 agent.name: LinuxHost01 agent.id: 003 manager.name: seminarioST data.uid: 0 data.dstuser: administrador(uid=1000) rule.mail: false rule.level: 3 rule.pci_dss: 10.2.5 rule.hipaa: 164.312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: PAM: Login session opened. rule.groups: pam, syslog, authentication_success rule.nist_800_53: AU.14, AC.7 rule.gdpr: IV_32.2 rule.firedtimes: 23 rule.mitre.technique: Valid Accounts rule.mitre.id: T1078 rule.mitre.tactic: Defense Evasion, Persistence, Privilege Escalation, Initial Access rule.id: 5501 rule.gpg13: 7.8, 7.9 location: /var/log/auth.log decoder.parent: pam decoder.name: pam id: 1761755169.2364333 full_log: Oct 29 17:26:08 seminarioST sshd[58
Oct 29, 2025 @ 17:26:09.509	predecoder.hostname: seminarioST predecoder.program_name: sshd predecoder.timestamp: Oct 29 17:26:08 input.type: log agent.ip: 10.10.10.2 agent.name: LinuxHost01 agent.id: 003 manager.name: seminarioST data.srcip: 10.10.10.4 data.dstuser: administrador data.srcport: 39562 rule.mail: false rule.level: 3 rule.hipaa: 164.312.b rule.pci_dss: 10.2.5 rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: sshd: authentication success. rule.groups: syslog, sshd, authentication_success rule.nist_800_53: AU.14, AC.7 rule.gdpr: IV_32.2 rule.firedtimes: 2 rule.mitre.technique: Valid Accounts, Remote Services rule.mitre.id: T1078, T1021 rule.mitre.tactic: Defense Evasion, Persistence, Privilege Escalation, Initial Access, Lateral Movement rule.id: 5715 rule.gpg13: 7.1, 7.2 location: /var/log/auth.log decoder.parent: sshd decoder.name: sshd id: 17
Oct 29, 2025 @ 17:14:38.931	predecoder.hostname: seminarioST predecoder.program_name: sshd predecoder.timestamp: Oct 29 17:14:37 input.type: log agent.ip: 10.10.10.2 agent.name: LinuxHost01 agent.id: 003 manager.name: seminarioST data.dstuser: administrador rule.firedtimes: 19 rule.mail: false rule.level: 3 rule.pci_dss: 10.2.5 rule.hipaa: 164.312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: PAM: Login session closed. rule.groups: pam, syslog rule.id: 5502 rule.nist_800_53: AU.14, AC.7 rule.gpg13: 7.8, 7.9 rule.gdpr: IV_32.2 location: /var/log/auth.log decoder.parent: pam decoder.name: pam id: 1761754478.2357196 full_log: Oct 29 17:14:37 seminarioST sshd[57953]: pam_unix(sshd:session): session closed for user administrador timestamp: Oct 29, 2025 @ 17:14:38.931 _index: wazuh-alerts-4.x-2025.10.29
Oct 29, 2025 @ 17:13:44.876	predecoder.hostname: seminarioST predecoder.program_name: sshd predecoder.timestamp: Oct 29 17:13:43 input.type: log agent.ip: 10.10.10.2 agent.name: LinuxHost01 agent.id: 003 manager.name: seminarioST data.uid: 0 data.dstuser: administrador(uid=1000) rule.mail: false rule.level: 3 rule.pci_dss: 10.2.5 rule.hipaa: 164.312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: PAM: Login session opened. rule.groups: pam, syslog, authentication_success rule.nist_800_53: AU.14, AC.7 rule.gdpr: IV_32.2 rule.firedtimes: 19 rule.mitre.technique: Valid Accounts rule.mitre.id: T1078 rule.mitre.tactic: Defense Evasion, Persistence, Privilege Escalation, Initial Access rule.id: 5501 rule.gpg13: 7.8, 7.9 location: /var/log/auth.log decoder.parent: pam decoder.name: pam id: 1761754424.2356728 full_log: Oct 29 17:13:43 seminarioST sshd[57
Oct 29, 2025 @ 17:13:44.831	predecoder.hostname: seminarioST predecoder.program_name: sshd predecoder.timestamp: Oct 29 17:13:43 input.type: log agent.ip: 10.10.10.2 agent.name: LinuxHost01

c) Analyze whether the dashboard displays additional information that was not present in the original logs. Investigate how Wazuh is able to provide enriched data and document your findings.

When I compare the raw Linux logs to what Wazuh shows in its alerts, Wazuh is clearly adding extra context that the original logs don't have. For example, /var/log/auth.log just says things like "Accepted password for administrador from 10.10.10.4" or "sudo: administrador : USER=root ; COMMAND=/usr/bin/hostname," but in alerts.json that same activity is turned into security event that includes a rule ID, a severity level, tags like "Privilege Escalation," the exact MITRE ATT&CK technique (for example T1548.003 for sudo abuse, T1078 for valid accounts over SSH), compliance mappings (PCI, HIPAA, etc.), and clean parsed fields like srcip, dstuser, and the exact command the user ran. So the dashboard/alerts aren't just reprinting logs, they're basically telling who did what, from where, with which command, and why that might matter from a security/compliance point of view.

d) Identify three different decoders used and the rule sets involved:
   - Decoders used: Identify the decoders applied to the logs to normalize the information.
   - Rule sets involved: Indicate the rules that were triggered and their severity level.

From the alerts pulled, we can clearly see multiple decoders, and which rules they triggered along with their severities. The sshd decoder handled SSH login lines like "Accepted password for administrador…," and that triggered rule 5715 ("sshd: authentication success.") with severity level 3, which Wazuh tags to MITRE techniques like Valid Accounts and Remote Services. The sudo decoder parsed sudo activity such as USER=root ; COMMAND=/usr/bin/hostname, and that fired rule 5402 ("Successful sudo to ROOT executed."), also level 3, which Wazuh classifies as privilege escalation (MITRE T1548.003). The pam decoder processed session events like pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000) and triggered rules 5501 ("PAM: Login session opened.") and 5502 ("PAM: Login session closed."), again level 3, used to track session start/stop for both SSH and sudo. We also saw the dpkg-decoder, which parsed /var/log/dpkg.log entries about packages being installed or half-configured and raised rules 2902 ("New dpkg installed") and 2904

("Dpkg half configured") with severity level 7, marking those as configuration changes. So in short: sshd -> rule 5715 (level 3), sudo -> rule 5402 (level 3), pam -> rules 5501/5502 (level 3), and dpkg-decoder -> rules 2902/2904 (level 7).

5. Suricata installation

Show a screenshot (or some) where it can be seen the Suricata folders and configuration file (/etc/suricata/suricata.yaml) contents.

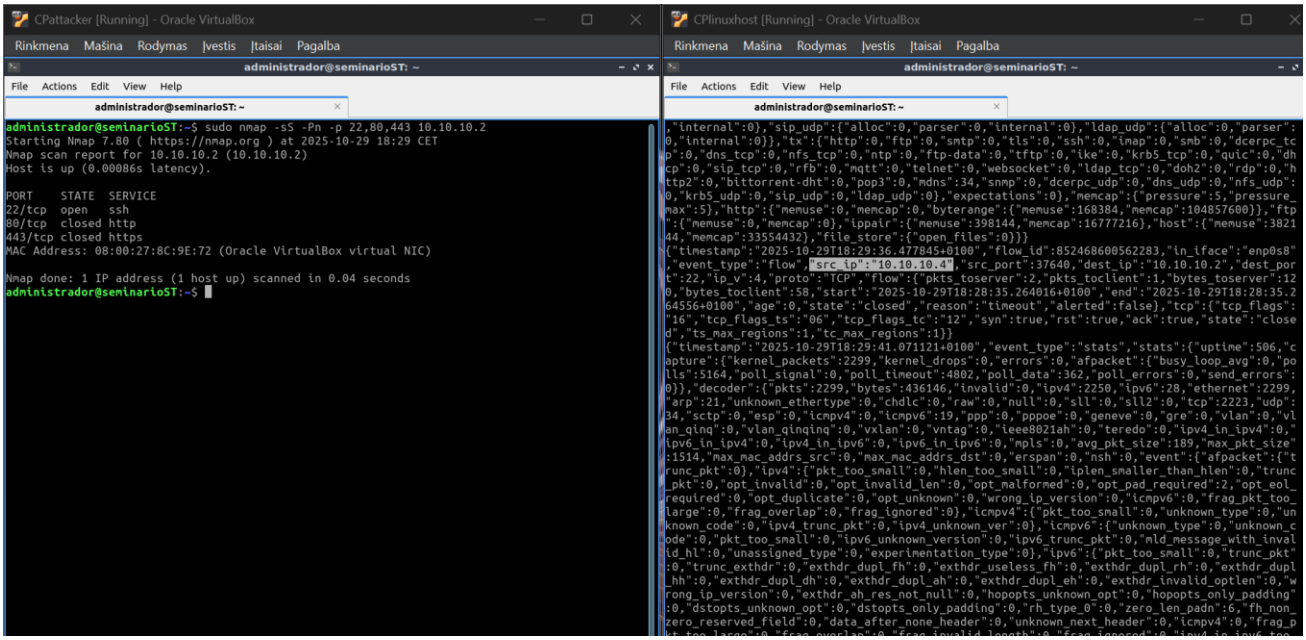
```
administrador@seminarioST:~$ sudo suricata --build-info
sudo systemctl status suricata
[sudo] password for administrador:
This is Suricata version 8.0.1 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1.1 HAVE_HTP_URI_NO
RMALIZE_HOOK PCRE_JIT HAVE_NSS HTTP2_DECOMPRESSION HAVE_LUA HAVE_JA3 HAVE_JA4 HAVE_LIBJANSSO
N TLS TLS_C11 MAGIC RUST POPCNT64
SIMD support: SSE_2
Atomic intrinsics: 1 2 4 8 byte(s)
64-bits, little-endian architecture
GCC version 11.4.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v8.0.1

Suricata Configuration:
  AF_PACKET support:                       yes
  AF_XDP support:                          no
  DPDK support:                            no
  eBPF support:                            no
  XDP support:                             no
  PF_RING support:                         no
  NFQueue support:                         yes
  NFLOG support:                           no
  IPFW support:                            no
  Netmap support:                          no
  DAG enabled:                             no
  Napatech enabled:                        no
  WinDivert enabled:                       no
  Npcap support:

  Unix socket enabled:                     yes
  Detection enabled:                       yes

  Libmagic support:                        yes
  libjansson support:                      yes
  hiredis support:                         yes
  hiredis async with libevent:             yes
```

```
ls: cannot open directory '/var/log/suricata': Permission denied
administrador@seminarioST:~$ sudo ls -R /var/log/suricata
/var/log/suricata:
certs  core  eve.json  fast.log  files  stats.log  suricata.log

/var/log/suricata/certs:

/var/log/suricata/core:

/var/log/suricata/files:
administrador@seminarioST:~$ sudo  ls -R /etc/suricata
/etc/suricata:
classification.config  local.rules  reference.config  rules  sid-msg.map  suricata.yaml  suricata.yaml.broken.1761034437  suricata.yaml.broken.1761034535  threshold.config

/etc/suricata/rules:
botcc.portgrouped.rules         emerging-chat.rules            emerging-games.rules          emerging-misc.rules            emerging-scada.rules        emerging-voip.rules
botcc.rules                     emerging-coinminer.rules       emerging-hunting.rules        emerging-mobile_malware.rules  emerging-scan.rules         emerging-web_client.rules
clarny.rules                    emerging-current_events.rules  emerging-icmp_info.rules      emerging-netbios.rules         emerging-shellcode.rules    emerging-web_server.rules
compromised.rules               emerging-deleted.rules         emerging-icmp.rules           emerging-p2p.rules             emerging-smtp.rules         emerging-web_specific_apps.rules
drop.rules                      emerging-dns.rules             emerging-imap.rules           emerging-phishing.rules        emerging-snmp.rules         emerging-worm.rules
dshield.rules                   emerging-dos.rules             emerging-inappropriate.rules  emerging-policy.rules          emerging-sql.rules          suricata.rules
emerging-activex.rules          emerging-exploit_kit.rules     emerging-info.rules           emerging-pop3.rules            emerging-telnet.rules       threatview_CS_c2.rules
emerging-adware_pup.rules       emerging-exploit.rules         emerging-ja3.rules            emerging-retired.rules         emerging-tftp.rules         tor.rules
emerging-attack_response.rules  emerging-ftp.rules             emerging-malware.rules        emerging-rpc.rules             emerging-user_agents.rules
administrador@seminarioST:~$
```

```
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://docs.suricata.io/en/latest/configuration/suricata-yaml.html

# This configuration file was generated by Suricata 8.0.1.
suricata-version: "8.0"

##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "10.10.10.2"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
    DC_SERVERS: "$HOME_NET"
    DNP3_SERVER: "$HOME_NET"
    DNP3_CLIENT: "$HOME_NET"
    MODBUS_CLIENT: "$HOME_NET"
    MODBUS_SERVER: "$HOME_NET"
    ENIP_CLIENT: "$HOME_NET"
```

6. Suricata events

Generate some events and show a screenshot (or some) where it can be seen the contents of Suricata events json file.



7. Suricata rule for “HTTP Attack Detected”

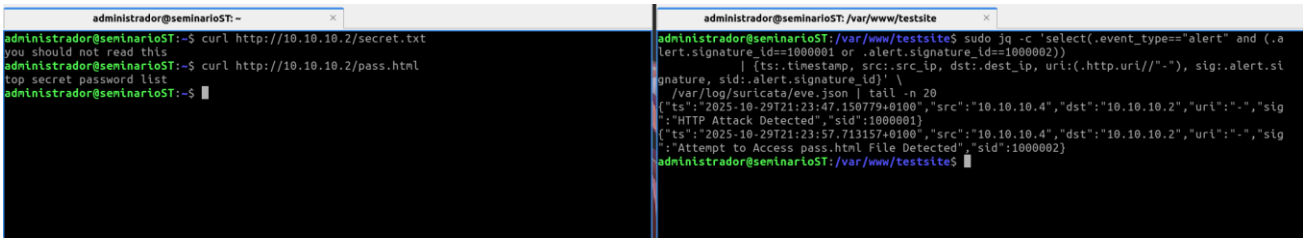
Show with screenshots the created rule. Access the protected file and show the event detection in Suricata (json file), and in Wazuh (generated event).

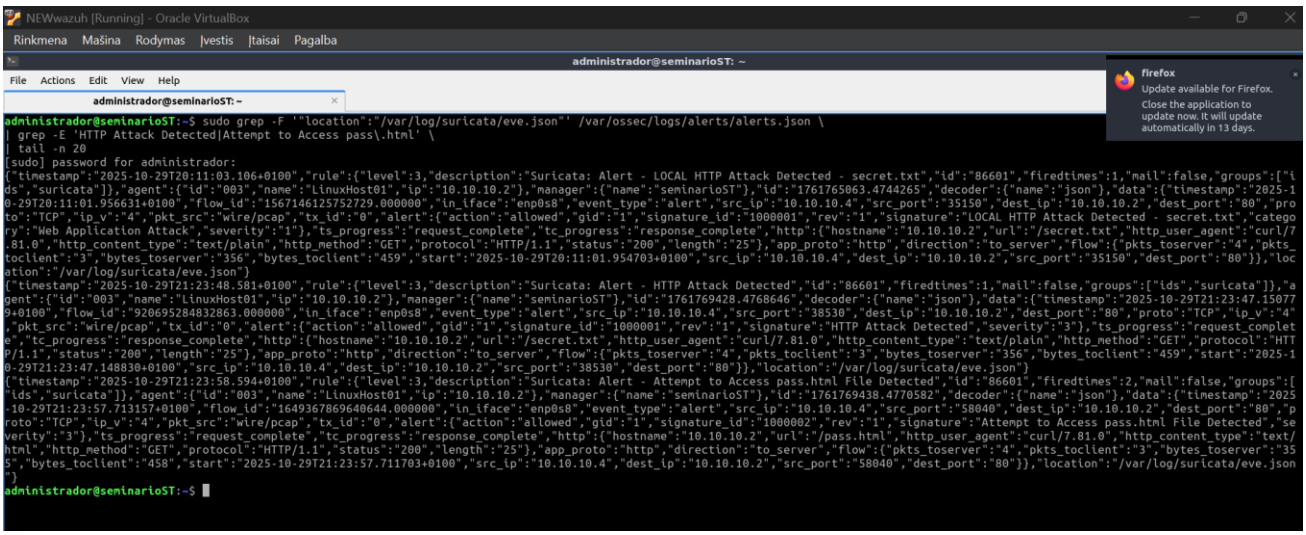
```
default-rule-path: /var/lib/suricata/rules

rule-files:
  - suricata.rules
  - /etc/suricata/rules/local.rules
##
## Auxiliary configuration files.
##
```

```
administrador@seminarioST:/var/www/testsite$ sudo cat /etc/suricata/rules/local.rules
alert http any any -> $HOME_NET any (msg:"HTTP Attack Detected"; http.method; content:"GET";
 nocase; http.uri; content:"/secret.txt"; nocase; sid:1000001; rev:1;)
alert http any any -> $HOME_NET any (msg:"Attempt to Access pass.html File Detected"; http.m
ethod; content:"GET"; nocase; http.uri; content:"/pass.html"; nocase; sid:1000002; rev:1;)
administrador@seminarioST:/var/www/testsite$
```

8. Suricata rule for “Attempt to Access pass.html File Detected”

Show with screenshots the created rule. Access the protected file and show the event detection in Suricata (json file), and in Wazuh (generated event).





With KQL:

location:"/var/log/suricata/eve.json" AND (data.alert.signature_id:1000001 OR data.alert.signature_id:1000002)