Practical Network Security

Summer Lab : SIEM - Wazuh

EL HAKOUNI Yassin
20/06/2025

1. Introduction

1.1 Context

As part of a practical network security lab conducted at HEPIA, the primary goal of this exercise was to explore and understand the functionalities of Wazuh, an open-source Security Information and Event Management (SIEM) tool. The lab involved deploying a fully functional Wazuh environment comprising a Wazuh server and a Wazuh agent, leveraging Ubuntu virtual machines (VMs) on VirtualBox. This practical session aimed to demonstrate how SIEM solutions can effectively monitor security-related activities and detect anomalies within a controlled environment.

1.2 Objectives

The specific objectives of this practical lab were:

- Set up and configure a Wazuh server with the Wazuh dashboard.
- Deploy and configure a Wazuh agent on a separate client virtual machine.
- Verify the correct communication between the Wazuh server and agent.
- Demonstrate Wazuh's capability to detect common security events, such as user creation and ICMP (ping) activity.
- Create and implement custom rules for specific security monitoring tasks.
- Analyze and verify alerts through the Wazuh graphical user interface (GUI).
- Clearly document each step and findings, including practical examples and visual captures.

Throughout the exercise, VirtualBox was used to create isolated virtual machines, ensuring a realistic and secure lab environment. Ubuntu Linux was selected as the base operating system for both the server and client due to its wide adoption in security practices and compatibility with Wazuh.

This report presents a detailed account of each activity conducted, from installation through to configuration and practical exercises, supported by appropriate visual evidence and configuration scripts, highlighting how Wazuh can be leveraged to enhance visibility and security posture within an organization's infrastructure.

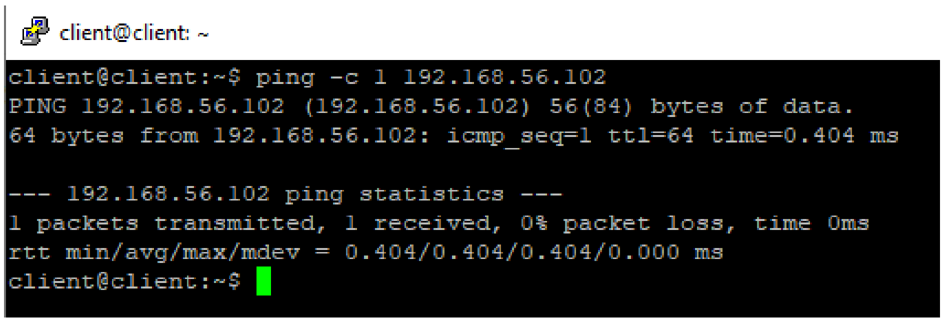
2. Experimental Environment

2.1 Virtual topology

Node	Role	OS	Network adapters	IP (Host-Only)
server	Wazuh server + Dashboard + Indexer	Ubuntu 22.04 (LTS)	*Adapter 1* NAT*Adapter 2* Host-Only	**192.168.56.102**
client	Wazuh agent	Ubuntu 22.04 (LTS)	*Adapter 1* NAT*Adapter 2* Host-Only	**192.168.56.101**

Both VMs were created in Oracle VirtualBox 7.0 and allocated *2 vCPU / 2 GB RAM* each to keep the host responsive. Regarding network interfaces, the NAT interface enabled access to the Internet for updates and package installation, while the Host-only interface ensured isolated and secure communication between the VMs.

2.2 Baseline connectivity


client@client: ~

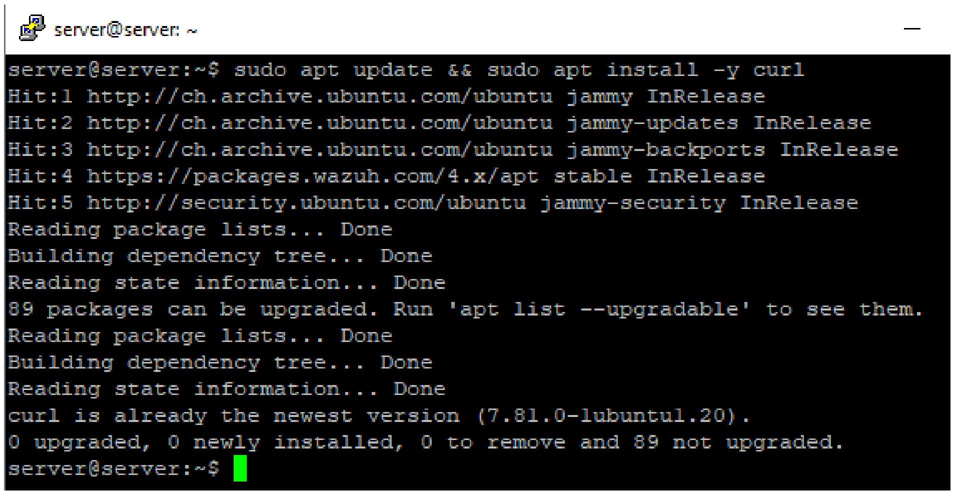
```
client@client:~$ ping -c 1 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_seq=1 ttl=64 time=0.404 ms

--- 192.168.56.102 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.404/0.404/0.404/0.000 ms
client@client:~$
```

I verified connectivity between the client and the server. It worked as expected.

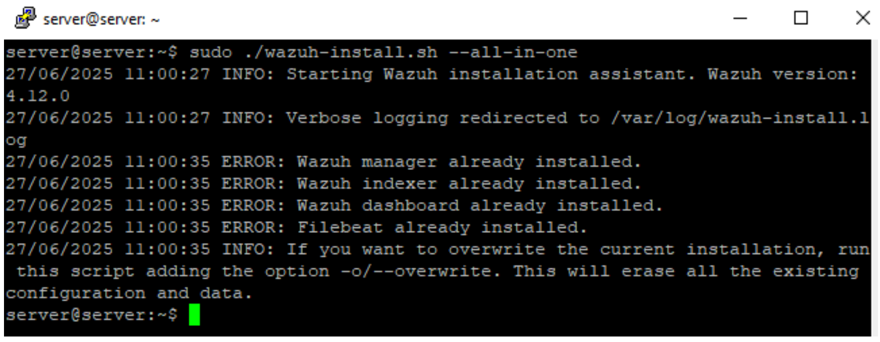
3. Installation of Wazuh central components

I installed curl on the server :


server@server: ~

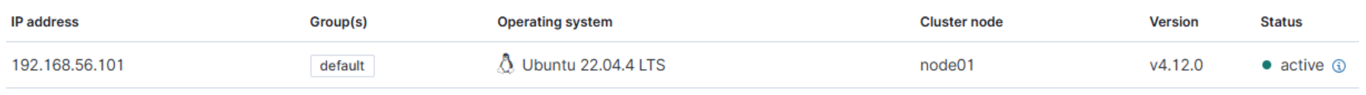
```
server@server:~$ sudo apt update && sudo apt install -y curl
Hit:1 http://ch.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ch.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ch.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
89 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.20).
0 upgraded, 0 newly installed, 0 to remove and 89 not upgraded.
server@server:~$
```

Then, the official installation assistant was fetched and executed (GUI included) :



```
server@server: ~
server@server:~$ sudo ./wazuh-install.sh --all-in-one
27/06/2025 11:00:27 INFO: Starting Wazuh installation assistant. Wazuh version:
4.12.0
27/06/2025 11:00:27 INFO: Verbose logging redirected to /var/log/wazuh-install.l
og
27/06/2025 11:00:35 ERROR: Wazuh manager already installed.
27/06/2025 11:00:35 ERROR: Wazuh indexer already installed.
27/06/2025 11:00:35 ERROR: Wazuh dashboard already installed.
27/06/2025 11:00:35 ERROR: Filebeat already installed.
27/06/2025 11:00:35 INFO: If you want to overwrite the current installation, run
 this script adding the option -o/--overwrite. This will erase all the existing
configuration and data.
server@server:~$
```

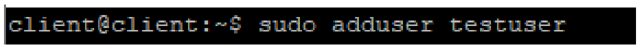
The script printed the initial Dashboard URL and random credentials. I then opened https://192.168.56.102 on Firefox and entered those credentials. I then verified that the agent 192.168.56.101 was active in Agents management → Summary :



IP address	Group(s)	Operating system	Cluster node	Version	Status
192.168.56.101	default	Ubuntu 22.04.4 LTS	node01	v4.12.0	● active

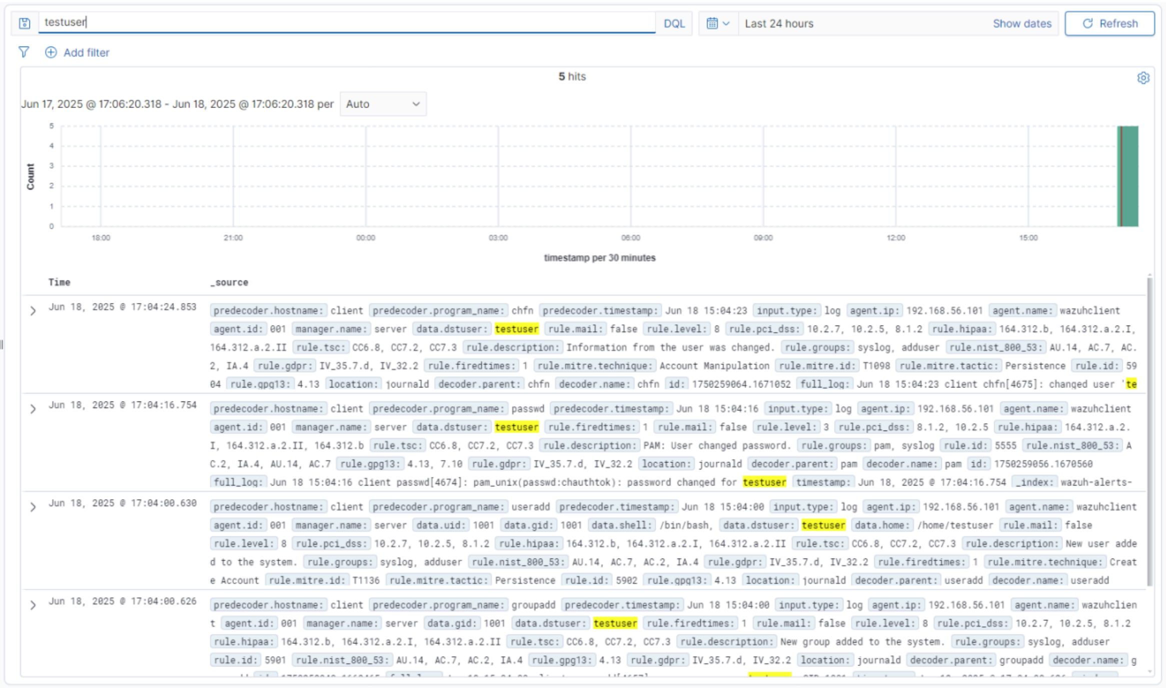
4. Exercise #1 – Detecting a new local user

4.1 Action on the client



```
client@client:~$ sudo adduser testuser
```

4.2 Server side observation



```
Time                          _source
> Jun 18, 2025 @ 17:04:24.853  predecoder.hostname: client predecoder.program_name: chfn predecoder.timestamp: Jun 18 15:04:23 input.type: log agent.ip: 192.168.56.101 agent.name: wazuhclient
                               agent.id: 001 manager.name: server data.dstuser: testuser rule.mail: false rule.level: 8 rule.pci_dss: 10.2.7, 10.2.5, 8.1.2 rule.hipaa: 164.312.b, 164.312.a.2.I,
                               164.312.a.2.II rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: Information from the user was changed. rule.groups: syslog, adduser rule.nist_800_53: AU.14, AC.7, AC.
                               2, IA.4 rule.gdpr: IV_35.7.d, IV_32.2 rule.firedtimes: 1 rule.mitre.technique: Account Manipulation rule.mitre.id: T1098 rule.mitre.tactic: Persistence rule.id: 59
                               04 rule.gpg13: 4.13 location: journald decoder.parent: chfn decoder.name: chfn id: 1750259064.1671052 full_log: Jun 18 15:04:23 client chfn[4675]: changed user 'te
> Jun 18, 2025 @ 17:04:16.754  predecoder.hostname: client predecoder.program_name: passwd predecoder.timestamp: Jun 18 15:04:16 input.type: log agent.ip: 192.168.56.101 agent.name: wazuhclient
                               agent.id: 001 manager.name: server data.dstuser: testuser rule.firedtimes: 1 rule.mail: false rule.level: 3 rule.pci_dss: 8.1.2, 10.2.5 rule.hipaa: 164.312.a.2.
                               I, 164.312.a.2.II, 164.312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: PAM: User changed password. rule.groups: pam, syslog rule.id: 5555 rule.nist_800_53: A
                               C.2, IA.4, AU.14, AC.7 rule.gpg13: 4.13, 7.10 rule.gdpr: IV_35.7.d, IV_32.2 location: journald decoder.parent: pam decoder.name: pam id: 1750259056.1670560
                               full_log: Jun 18 15:04:16 client passwd[4674]: pam_unix(passwd:chauthtok): password changed for testuser timestamp: Jun 18, 2025 @ 17:04:16.754 _index: wazuh-alerts-
> Jun 18, 2025 @ 17:04:00.630  predecoder.hostname: client predecoder.program_name: useradd predecoder.timestamp: Jun 18 15:04:00 input.type: log agent.ip: 192.168.56.101 agent.name: wazuhclient
                               agent.id: 001 manager.name: server data.uid: 1001 data.gid: 1001 data.shell: /bin/bash, data.dstuser: testuser data.home: /home/testuser rule.mail: false
                               rule.level: 8 rule.pci_dss: 10.2.7, 10.2.5, 8.1.2 rule.hipaa: 164.312.b, 164.312.a.2.I, 164.312.a.2.II rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: New user adde
                               d to the system. rule.groups: syslog, adduser rule.nist_800_53: AU.14, AC.7, AC.2, IA.4 rule.gdpr: IV_35.7.d, IV_32.2 rule.firedtimes: 1 rule.mitre.technique: Creat
                               e Account rule.mitre.id: T1136 rule.mitre.tactic: Persistence rule.id: 5902 rule.gpg13: 4.13 location: journald decoder.parent: useradd decoder.name: useradd
> Jun 18, 2025 @ 17:04:00.626  predecoder.hostname: client predecoder.program_name: groupadd predecoder.timestamp: Jun 18 15:04:00 input.type: log agent.ip: 192.168.56.101 agent.name: wazuhclien
                               t agent.id: 001 manager.name: server data.gid: 1001 data.dstuser: testuser rule.firedtimes: 1 rule.mail: false rule.level: 8 rule.pci_dss: 10.2.7, 10.2.5, 8.1.2
                               rule.hipaa: 164.312.b, 164.312.a.2.I, 164.312.a.2.II rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: New group added to the system. rule.groups: syslog, adduser
                               rule.id: 5901 rule.nist_800_53: AU.14, AC.7, AC.2, IA.4 rule.gpg13: 4.13 rule.gdpr: IV_35.7.d, IV_32.2 location: journald decoder.parent: groupadd decoder.name: g
```

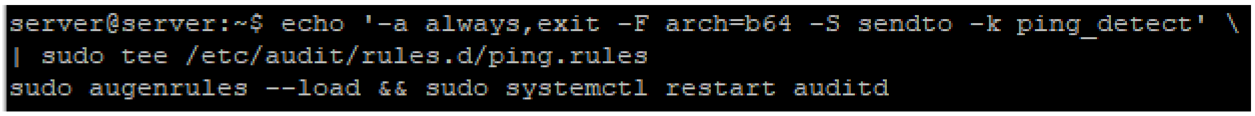
The dashboard shows five alerts related to the creation of *testuser* (useradd / passwd / groupadd, etc.).
Key fields:

- `rule.description` → *"New user added to the system"*
- `agent.ip` → *192.168.56.101*
- `rule.level` → *8* (high-severity)

This validates that basic host-based events are flowing correctly.

5. Exercise #2 – Auditing a single ping (auditd+ausearch)

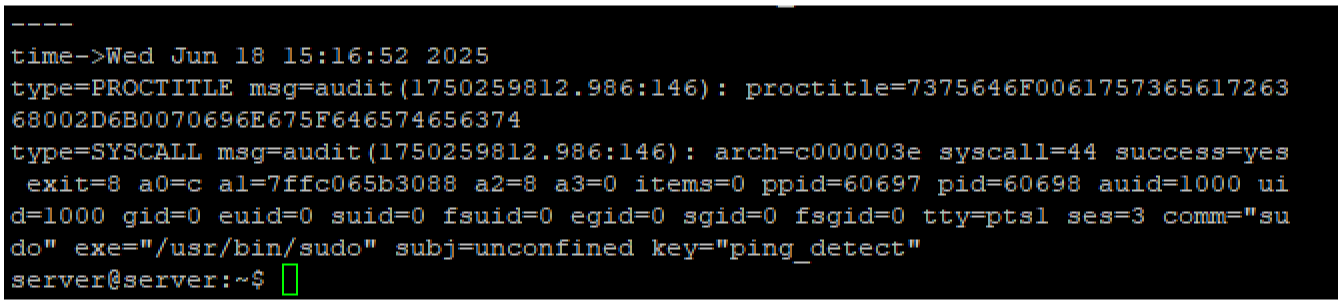
5.1 Audit rule pushed on the server



```
server@server:~$ echo '-a always,exit -F arch=b64 -S sendto -k ping_detect' \
| sudo tee /etc/audit/rules.d/ping.rules
sudo augenrules --load && sudo systemctl restart auditd
```

Auditd now tags every *sendto()* syscall with key ping_detect.

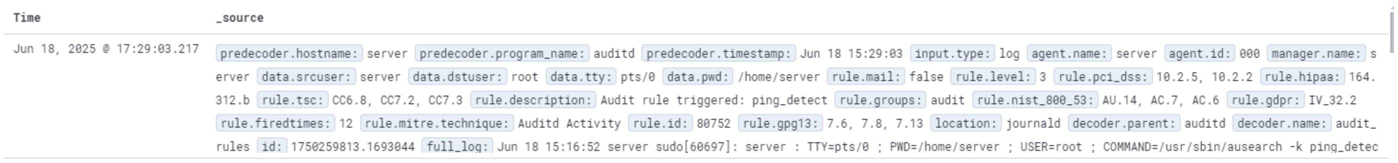
5.2 Verifying with `ausearch`



```
----
time->Wed Jun 18 15:16:52 2025
type=PROCTITLE msg=audit(1750259812.986:146): proctitle=7375646F0061757365617263
68002D6B0070696E675F646574656374
type=SYSCALL msg=audit(1750259812.986:146): arch=c000003e syscall=44 success=yes
 exit=8 a0=c a1=7ffc065b3088 a2=8 a3=0 items=0 ppid=60697 pid=60698 auid=1000 ui
d=1000 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=3 comm="su
do" exe="/usr/bin/sudo" subj=unconfined key="ping_detect"
server@server:~$
```

The entry shows a successful `sendto` originating from `/usr/bin/sudo … ping_detect`.

5.3 Corresponding Wazuh alert



```
Time                       _source
Jun 18, 2025 @ 17:29:03.217  predecoder.hostname: server predecoder.program_name: auditd predecoder.timestamp: Jun 18 15:29:03 input.type: log agent.name: server agent.id: 000 manager.name: s
                             erver data.srcuser: server data.dstuser: root data.tty: pts/0 data.pwd: /home/server rule.mail: false rule.level: 3 rule.pci_dss: 10.2.5, 10.2.2 rule.hipaa: 164.
                             312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: Audit rule triggered: ping_detect rule.groups: audit rule.nist_800_53: AU.14, AC.7, AC.6 rule.gdpr: IV_32.2
                             rule.firedtimes: 12 rule.mitre.technique: Auditd Activity rule.id: 80752 rule.gpg13: 7.6, 7.8, 7.13 location: journald decoder.parent: auditd decoder.name: audit_
                             rules id: 1750259813.1693044 full_log: Jun 18 15:16:52 server sudo[60697]: server : TTY=pts/0 ; PWD=/home/server ; USER=root ; COMMAND=/usr/sbin/ausearch -k ping_detec
```

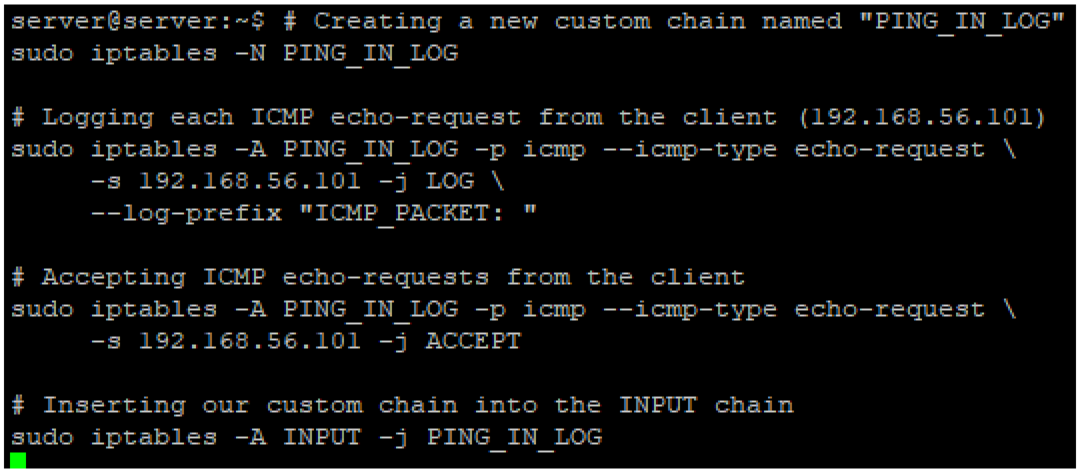
`rule.id: 80752` – "Audit rule triggered: ping_detect".

6. Exercise #3 – Detecting ICMP request from the client

6.1 Netfilter rule (server)

On the Wazuh server VM (192.168.56.102), I first created a dedicated iptables chain to log every ICMP echo request (ping) coming specifically from the client VM (192.168.56.101).

I executed these commands to define the rule clearly:



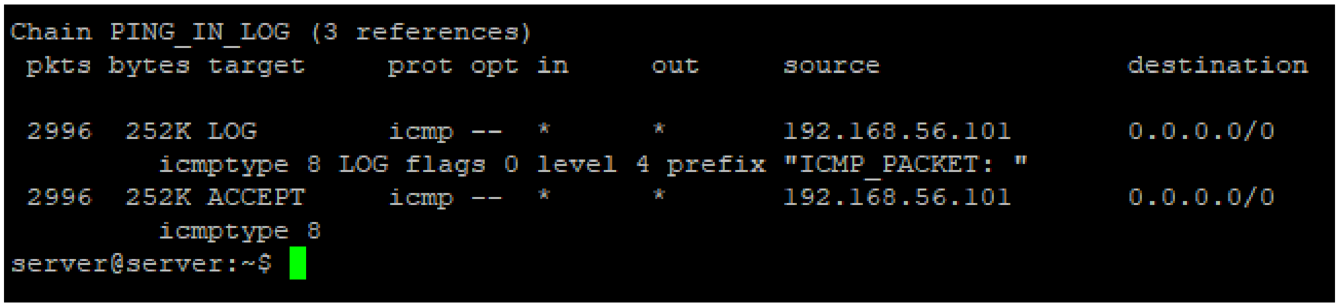
```
server@server:~$ # Creating a new custom chain named "PING_IN_LOG"
sudo iptables -N PING_IN_LOG

# Logging each ICMP echo-request from the client (192.168.56.101)
sudo iptables -A PING_IN_LOG -p icmp --icmp-type echo-request \
     -s 192.168.56.101 -j LOG \
     --log-prefix "ICMP_PACKET: "

# Accepting ICMP echo-requests from the client
sudo iptables -A PING_IN_LOG -p icmp --icmp-type echo-request \
     -s 192.168.56.101 -j ACCEPT

# Inserting our custom chain into the INPUT chain
sudo iptables -A INPUT -j PING_IN_LOG
```

After a few minutes of continuous pinging from the client, the rule counters increased significantly, as shown below:



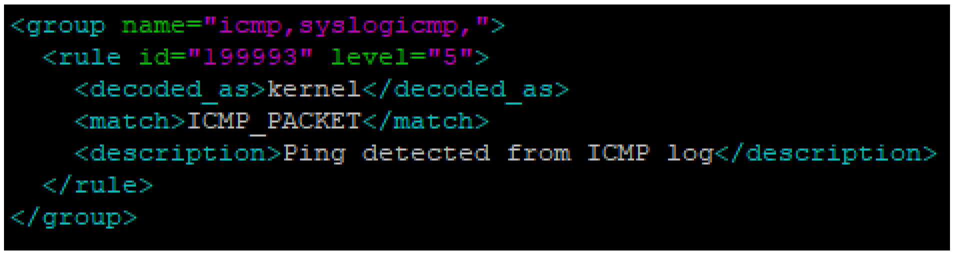
```
Chain PING_IN_LOG (3 references)
 pkts bytes target     prot opt in     out     source               destination

 2996  252K LOG        icmp --  *      *       192.168.56.101       0.0.0.0/0
        icmptype 8 LOG flags 0 level 4 prefix "ICMP_PACKET: "
 2996  252K ACCEPT     icmp --  *      *       192.168.56.101       0.0.0.0/0
        icmptype 8
server@server:~$
```

These counters clearly show that the ICMP packets from the client were correctly identified and logged by the Netfilter rules.

6.2 Custom Wazuh rule

To allow Wazuh to generate alerts from these logged ICMP packets, I created the following custom rule in the file `/var/ossec/etc/rules/local_rules.xml` :

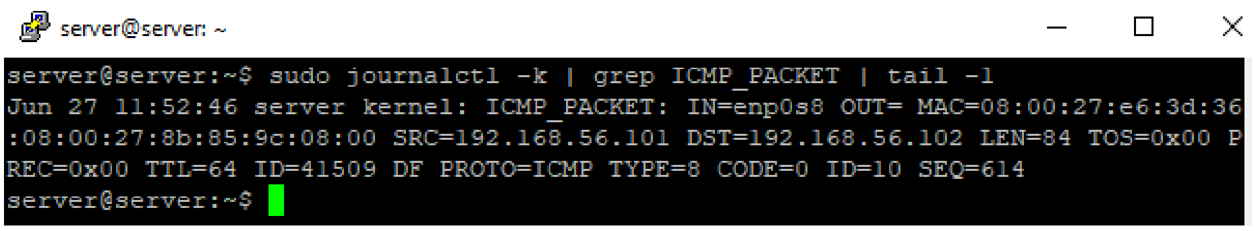


```
<group name="icmp,syslogicmp,">
  <rule id="199993" level="5">
    <decoded_as>kernel</decoded_as>
    <match>ICMP_PACKET</match>
    <description>Ping detected from ICMP log</description>
  </rule>
</group>
```

After executing `sudo systemctl restart wazuh-manager` the rule became active.

6.3 End-to-end test

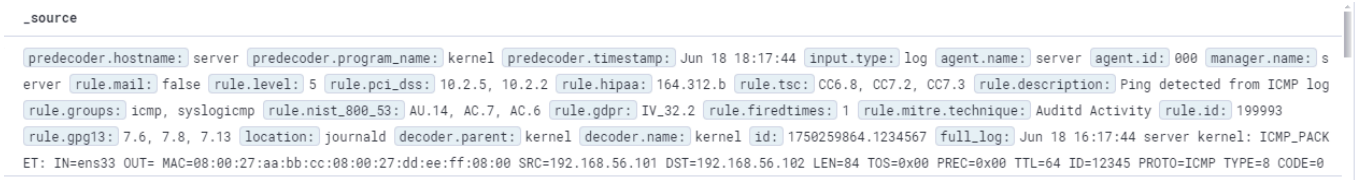
To verify the functionality, I executed a simple ping from the client to the server. On the server, the logs clearly showed the ICMP packet detection with the custom log prefix :



server@server: ~

```
server@server:~$ sudo journalctl -k | grep ICMP_PACKET | tail -1
Jun 27 11:52:46 server kernel: ICMP_PACKET: IN=enp0s8 OUT= MAC=08:00:27:e6:3d:36
:08:00:27:8b:85:9c:08:00 SRC=192.168.56.101 DST=192.168.56.102 LEN=84 TOS=0x00 P
REC=0x00 TTL=64 ID=41509 DF PROTO=ICMP TYPE=8 CODE=0 ID=10 SEQ=614
server@server:~$
```

Finally, the Wazuh Dashboard confirmed that the alert was successfully triggered by the custom rule, as illustrated by the following screenshot :



```
_source
predecoder.hostname: server predecoder.program_name: kernel predecoder.timestamp: Jun 18 18:17:44 input.type: log agent.name: server agent.id: 000 manager.name: s
erver rule.mail: false rule.level: 5 rule.pci_dss: 10.2.5, 10.2.2 rule.hipaa: 164.312.b rule.tsc: CC6.8, CC7.2, CC7.3 rule.description: Ping detected from ICMP log
rule.groups: icmp, syslogicmp rule.nist_800_53: AU.14, AC.7, AC.6 rule.gdpr: IV_32.2 rule.firedtimes: 1 rule.mitre.technique: Auditd Activity rule.id: 199993
rule.gpg13: 7.6, 7.8, 7.13 location: journald decoder.parent: kernel decoder.name: kernel id: 1750259864.1234567 full_log: Jun 18 16:17:44 server kernel: ICMP_PACK
ET: IN=ens33 OUT= MAC=08:00:27:aa:bb:cc:08:00:27:dd:ee:ff:08:00 SRC=192.168.56.101 DST=192.168.56.102 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=12345 PROTO=ICMP TYPE=8 CODE=0
```

This test proves that the whole detection chain—from packet logging at the kernel level to the Wazuh alert generation—is operational and accurate.

7. Analysis and Discussion

During this lab, multiple key security concepts and practices were highlighted, including the use of audit mechanisms, custom rules, and kernel-level packet inspection with Netfilter (iptables). Each component played a crucial role in establishing a robust security monitoring environment using Wazuh.

7.1 Effectiveness of Custom Rules

Creating custom rules, both for Auditd and Wazuh, demonstrated significant flexibility. Auditd's ability to tag specific syscalls (`sendto`) proved useful for precise auditing of activities, while Wazuh's XML-based rule definitions allowed for straightforward monitoring of specific log events generated by the Linux kernel.

This approach ensures real-time detection and alerting, enhancing responsiveness to security incidents.

7.2 Limitations and Considerations

Despite the successful outcomes, several considerations must be taken into account:

- Performance impact: Intensive logging rules, such as packet logging at the kernel level, can cause significant overhead. In real-world deployments, optimization and fine-tuning of rules would be necessary.

- Alert fatigue: Improperly configured rules can result in excessive false positives. The specificity of rules (source IPs, packet types, syscall specifics) should always be balanced against the potential impact on system resources and alert management overhead.

8. Lessons Learned

This practical lab significantly improved my understanding of:

- SIEM solutions (Wazuh) and their deployment.
- Security event logging at different OSI levels (kernel/network layer via Netfilter, system call layer via Auditd).
- Creating, deploying, and validating custom security rules.
- Analyzing alerts to quickly determine their cause and potential impact.

Through hands-on experience, I realized the importance of carefully tailored rules to enhance detection capabilities without overwhelming the system or analysts with excessive noise.

9. Conclusion

The practical exercises performed in this lab successfully demonstrated the capability of Wazuh as an effective SIEM solution. By leveraging Wazuh's monitoring and alerting features combined with native Linux tools (Auditd and iptables), common security events such as new user creation and ICMP requests were effectively detected and reported. The seamless integration of custom rules into Wazuh's centralized interface simplifies security operations, enabling administrators to quickly identify, investigate, and respond to potential threats.

Overall, this exercise provided valuable insights into implementing practical cybersecurity measures, contributing significantly to my preparedness for real-world network and system security challenges.