Configuring Wazuh Email Alerts with Gmail SMTP

Created By: Amir Raza

Follow Me: www.linkedin.com/in/amirsoc

Configuring Wazuh Email Alerts with Gmail SMTP

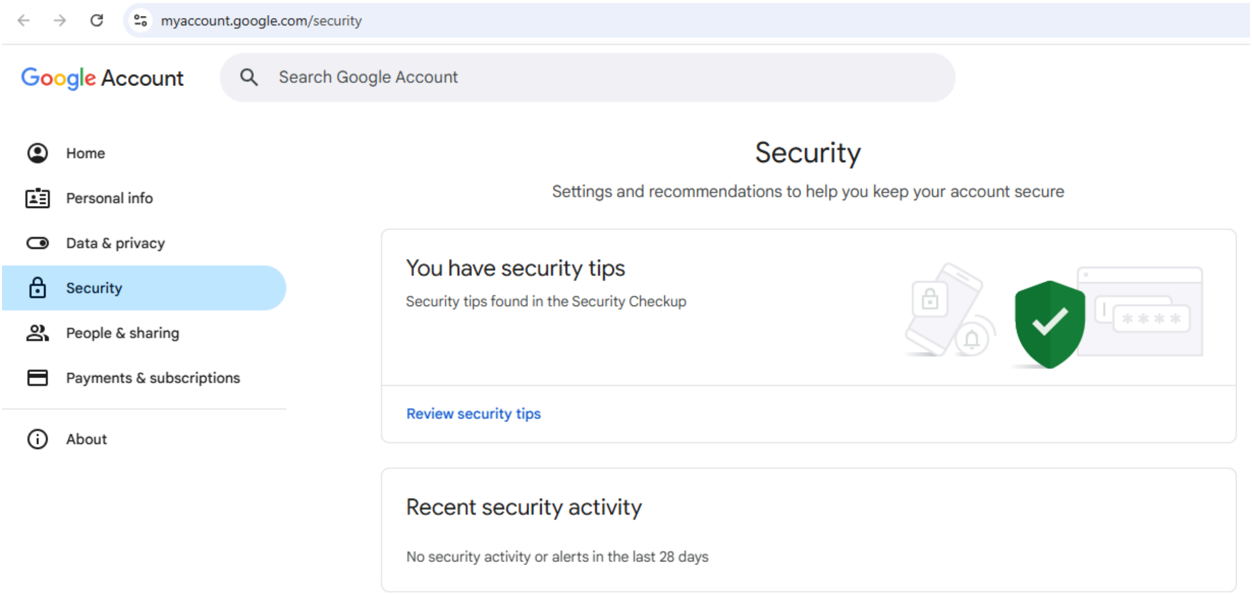
This guide will walk you through setting up email alerts in Wazuh using Gmail SMTP with Postfix on Ubuntu. You'll learn how to install required packages, set up Gmail SMTP, test email sending, and trigger real alerts.

Prerequisites

- Wazuh Manager (e.g., on Kali Linux in VirtualBox or Ubuntu)
- Gmail account
- App password from Gmail (we will create this)
- Internet access on Wazuh server

Step 1: Generate Gmail App Password

1. Log in to your Gmail account (usually from Windows browser).
2. Go to https://myaccount.google.com/security.



Security

Settings and recommendations to help you keep your account secure

You have security tips

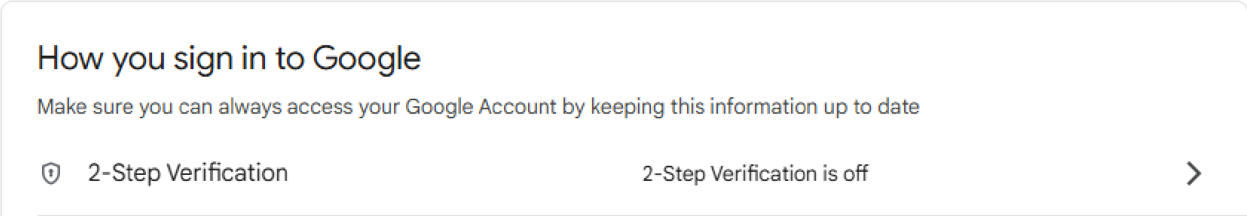
Security tips found in the Security Checkup

Recent security activity

No security activity or alerts in the last 28 days

3. Enable **2-Step Verification** if not already.

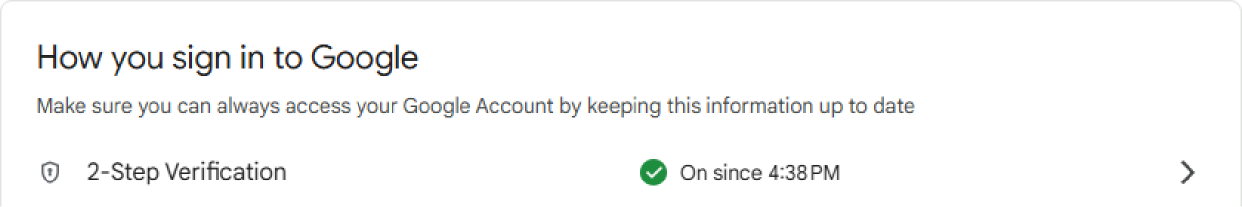
   Turn on 2-Step Verification (if not already).



How you sign in to Google

Make sure you can always access your Google Account by keeping this information up to date

2-Step Verification — 2-Step Verification is off

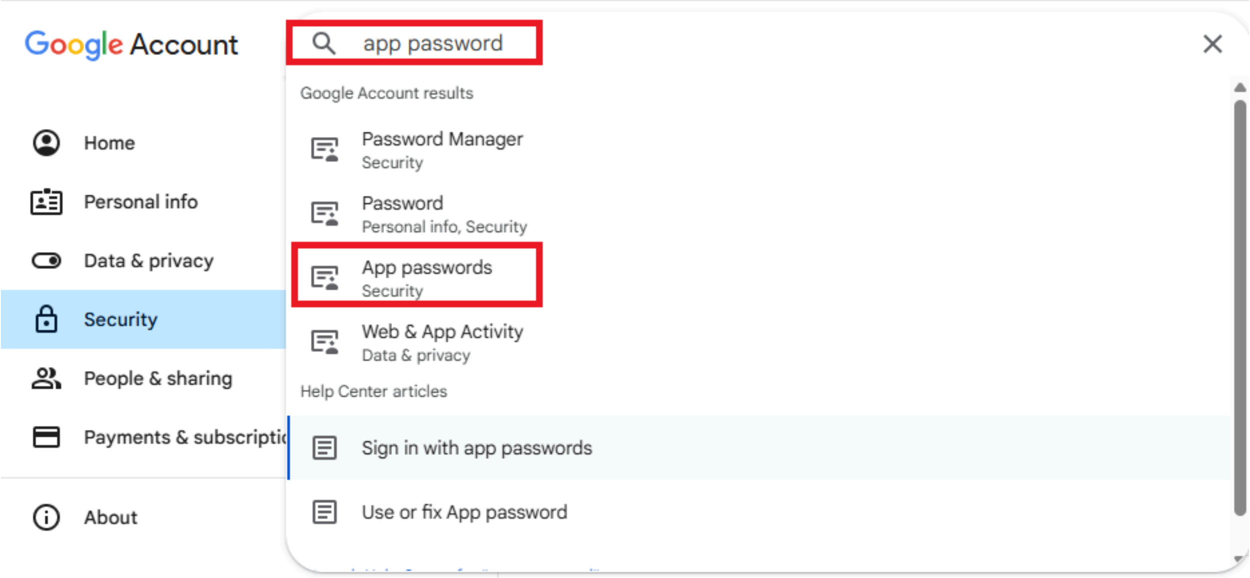


How you sign in to Google

Make sure you can always access your Google Account by keeping this information up to date

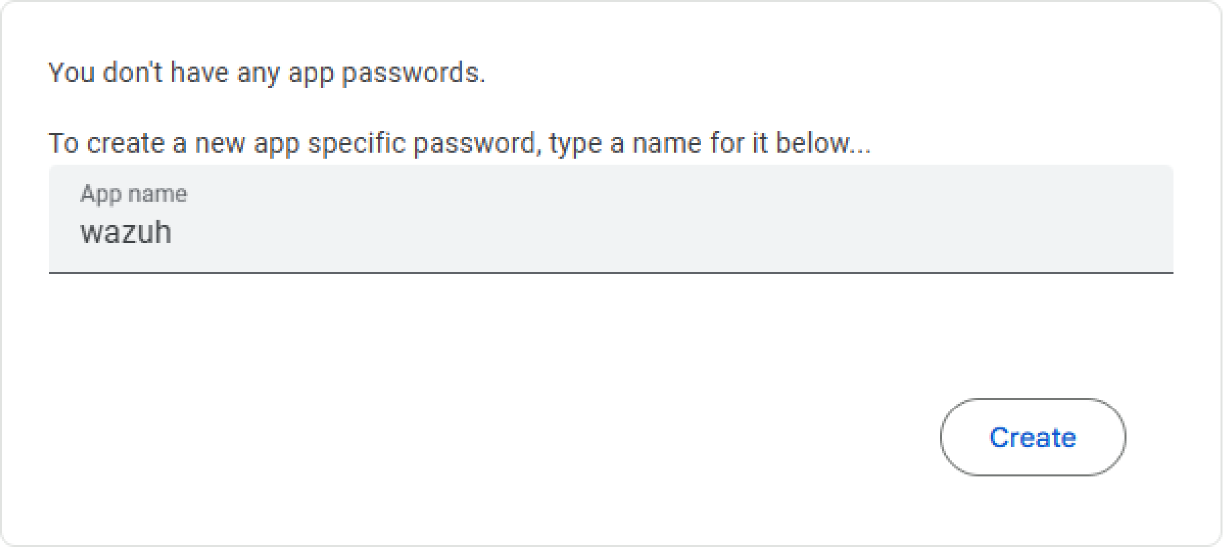
2-Step Verification — On since 4:38 PM

4. After enabling, find **App passwords** below.



app password

5. Create a new app password:
   - App: Other (Custom name → e.g., `wazuh`)
   - Device: linux server

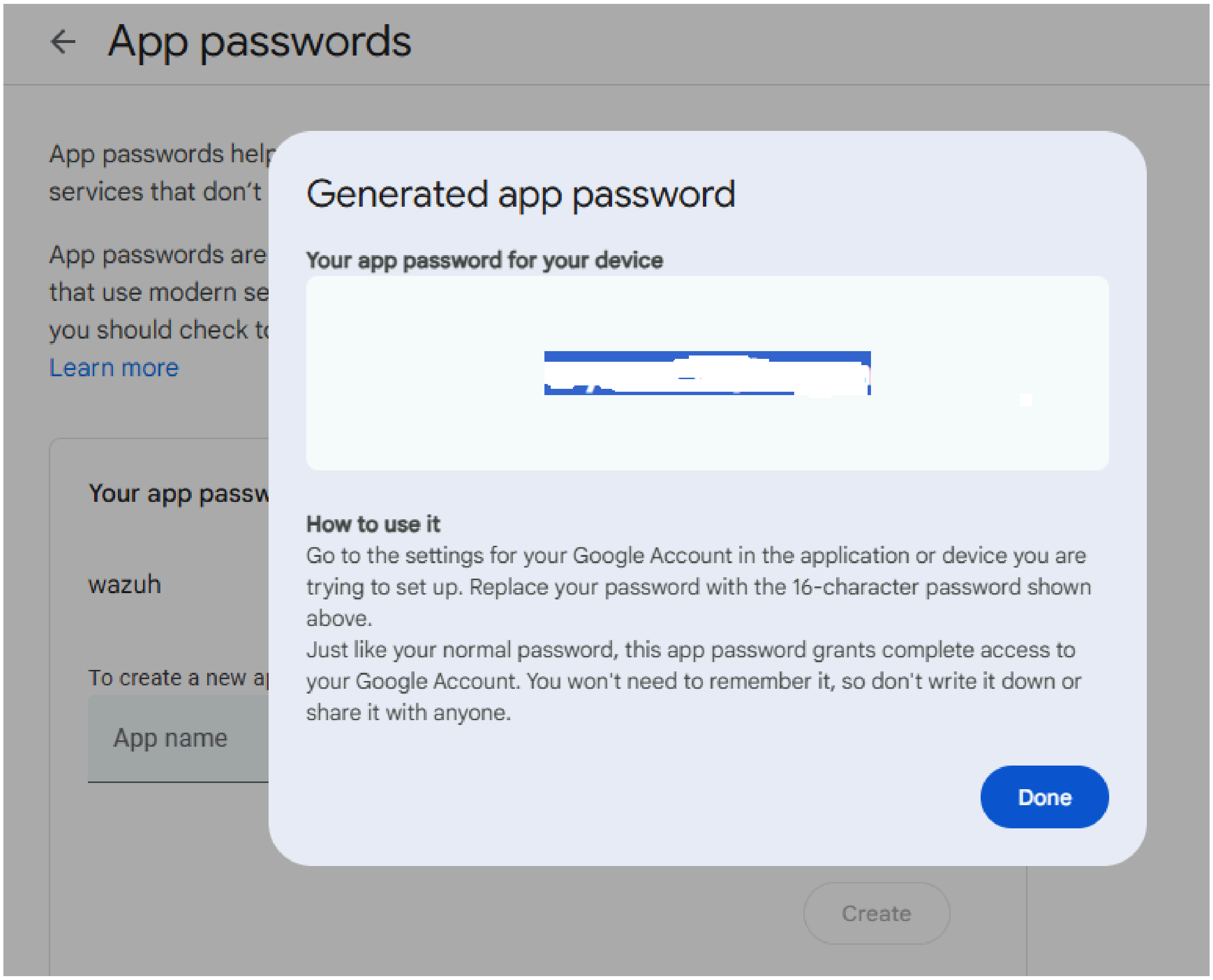


You don't have any app passwords.

To create a new app specific password, type a name for it below...

App name: wazuh

6. Click **Generate** and copy the **16-character password**.



App passwords

App passwords help
services that don't

App passwords are
that use modern se
you should check t
Learn more

Your app passw

wazuh

To create a new a

Generated app password

Your app password for your device

[redacted]

How to use it
Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above.
Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

7. Save it somewhere secure — you will use it in Linux config.

Store Gmail App Password for Postfix

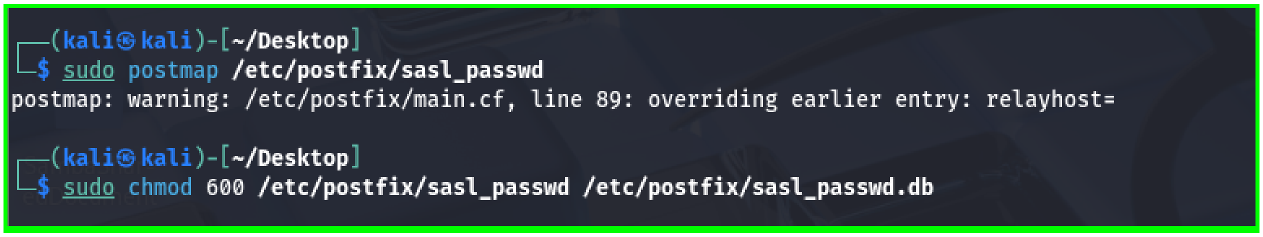
In terminal:

```
echo "[smtp.gmail.com]:587 yourgmail@gmail.com:your_app_password" | sudo tee /etc/postfix/sasl_passwd
```

Replace `yourgmail@gmail.com` and `your_app_password` with your real Gmail and the 16-character app password (no spaces).

Then secure it:

```
sudo postmap /etc/postfix/sasl_passwd
sudo chmod 600 /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
```



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo postmap /etc/postfix/sasl_passwd
postmap: warning: /etc/postfix/main.cf, line 89: overriding earlier entry: relayhost=

┌──(kali㉿kali)-[~/Desktop]
└─$ sudo chmod 600 /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
```

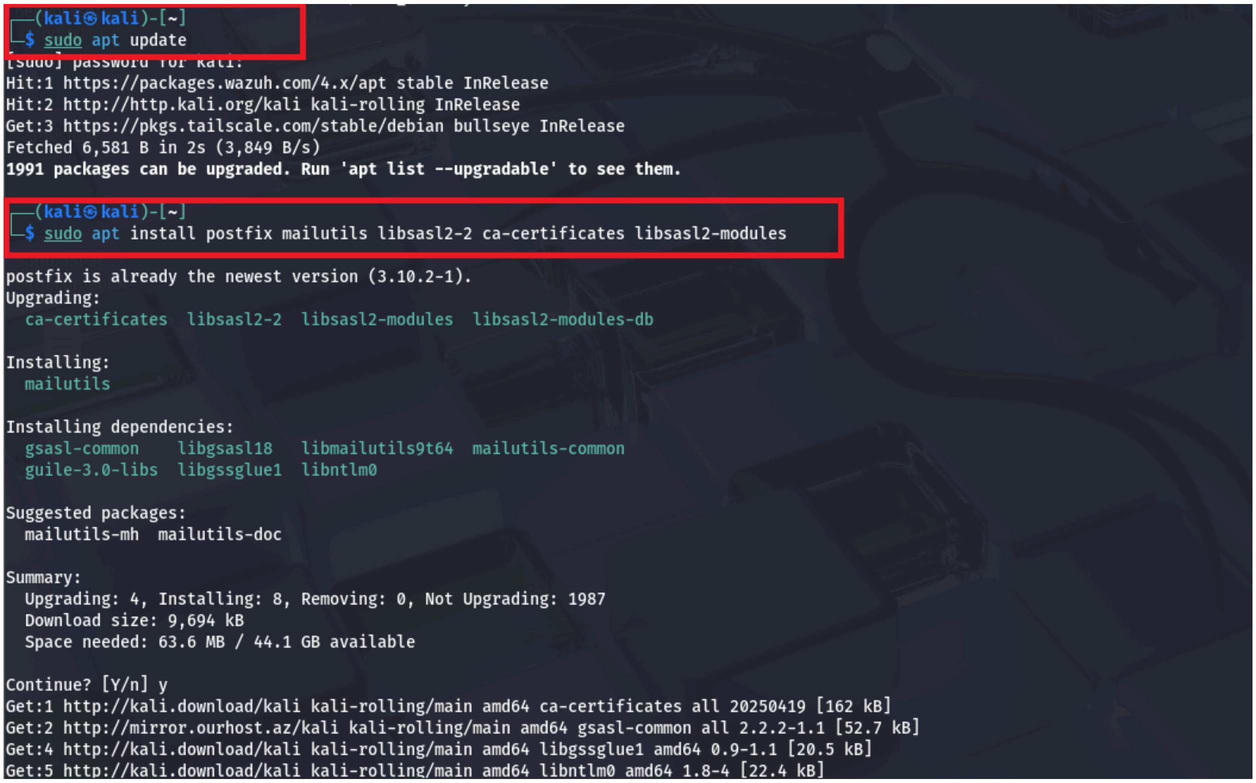
This step lets your server send emails without needing your real Gmail password.

Step 2: Install Required Packages

On your Wazuh server (Kali Linux or Ubuntu):

```
sudo apt update
```

```
sudo apt install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules
```



```
┌──(kali㉿kali)-[~]
└─$ sudo apt update
[sudo] password for kali:
Hit:1 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:2 http://http.kali.org/kali kali-rolling InRelease
Get:3 https://pkgs.tailscale.com/stable/debian bullseye InRelease
Fetched 6,581 B in 2s (3,849 B/s)
1991 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(kali㉿kali)-[~]
└─$ sudo apt install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules

postfix is already the newest version (3.10.2-1).
Upgrading:
  ca-certificates  libsasl2-2  libsasl2-modules  libsasl2-modules-db

Installing:
  mailutils

Installing dependencies:
  gsasl-common    libgsasl18   libmailutils9t64  mailutils-common
  guile-3.0-libs  libgssglue1  libntlm0

Suggested packages:
  mailutils-mh  mailutils-doc

Summary:
  Upgrading: 4, Installing: 8, Removing: 0, Not Upgrading: 1987
  Download size: 9,694 kB
  Space needed: 63.6 MB / 44.1 GB available

Continue? [Y/n] y
Get:1 http://kali.download/kali kali-rolling/main amd64 ca-certificates all 20250419 [162 kB]
Get:2 http://mirror.ourhost.az/kali kali-rolling/main amd64 gsasl-common all 2.2.2-1.1 [52.7 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 libgssglue1 amd64 0.9-1.1 [20.5 kB]
Get:5 http://kali.download/kali kali-rolling/main amd64 libntlm0 amd64 1.8-4 [22.4 kB]
```

During install, if asked, choose “No configuration.”

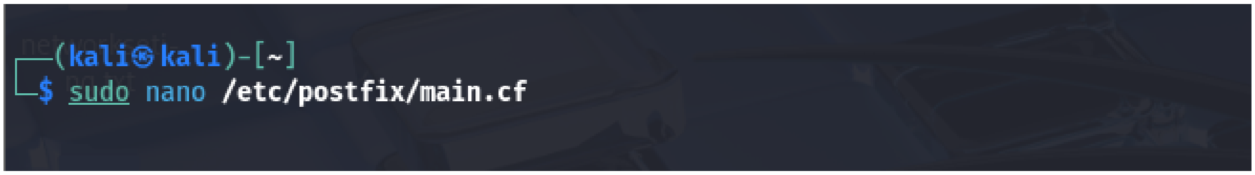
What each package does:

- postfix: the mail server to send emails.
- mailutils: test emails with `mail` command.
- libsasl2, ca-certificates: secure authentication with Gmail.

Step 3: Configure Postfix to Use Gmail SMTP

Open postfix config:

sudo nano /etc/postfix/main.cf



```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/postfix/main.cf
```

At the end, add:

relayhost = [smtp.gmail.com]:587

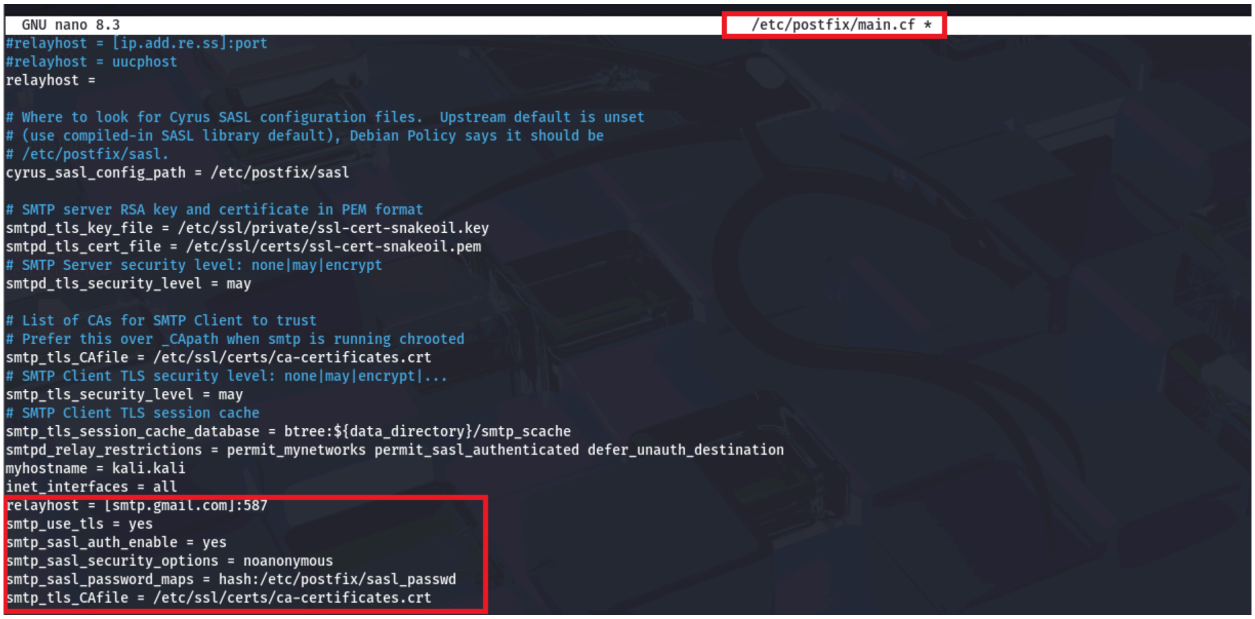
smtp_use_tls = yes

smtp_sasl_auth_enable = yes

smtp_sasl_security_options = noanonymous

smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd

smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt

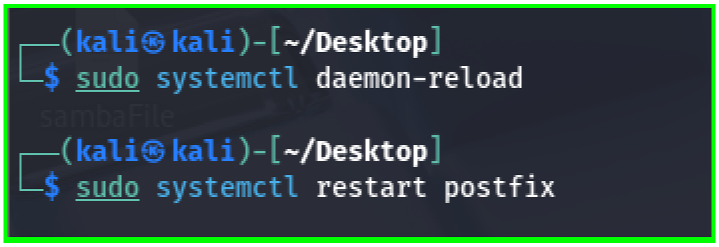


```
GNU nano 8.3                                                  /etc/postfix/main.cf *
#relayhost = [ip.add.re.ss]:port
#relayhost = uucphost
relayhost =

# Where to look for Cyrus SASL configuration files.  Upstream default is unset
# (use compiled-in SASL library default), Debian Policy says it should be
# /etc/postfix/sasl.
cyrus_sasl_config_path = /etc/postfix/sasl

# SMTP server RSA key and certificate in PEM format
smtpd_tls_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
# SMTP Server security level: none|may|encrypt
smtpd_tls_security_level = may

# List of CAs for SMTP Client to trust
# Prefer this over _CApath when smtp is running chrooted
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
# SMTP Client TLS security level: none|may|encrypt|...
smtp_tls_security_level = may
# SMTP Client TLS session cache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = kali.kali
inet_interfaces = all
relayhost = [smtp.gmail.com]:587
smtp_use_tls = yes
smtp_sasl_auth_enable = yes
smtp_sasl_security_options = noanonymous
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

This tells postfix to use Gmail SMTP with encryption.

Step 4: Restart Postfix & Test Email

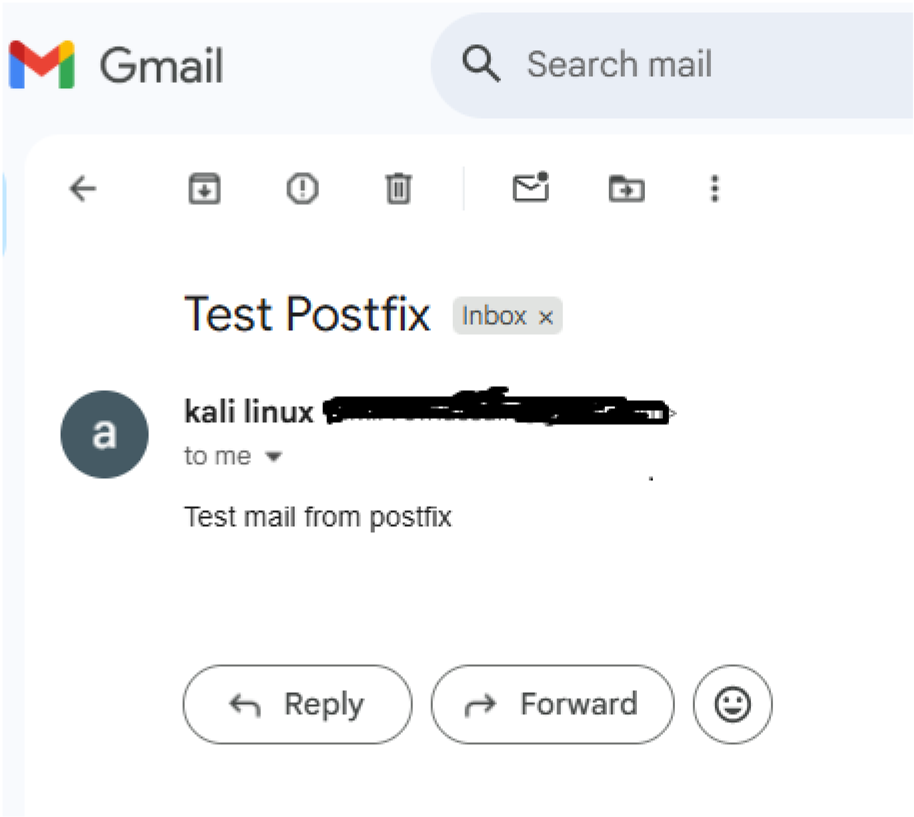
sudo systemctl daemon-reload
sudo systemctl restart postfix



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo systemctl daemon-reload

┌──(kali㉿kali)-[~/Desktop]
└─$ sudo systemctl restart postfix
```

echo "Test mail from postfix" | mail -s "Test Postfix" -r "yourgmail@gmail.com" yourgmail@gmail.com

Check Gmail inbox → you should see an email titled "Test Postfix."



Test Postfix

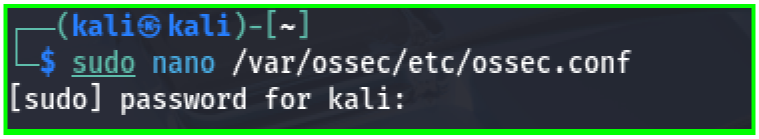
kali linux [redacted]
to me

Test mail from postfix

Step 5: Configure Wazuh to Send Emails

Open Wazuh config:

sudo nano /var/ossec/etc/ossec.conf



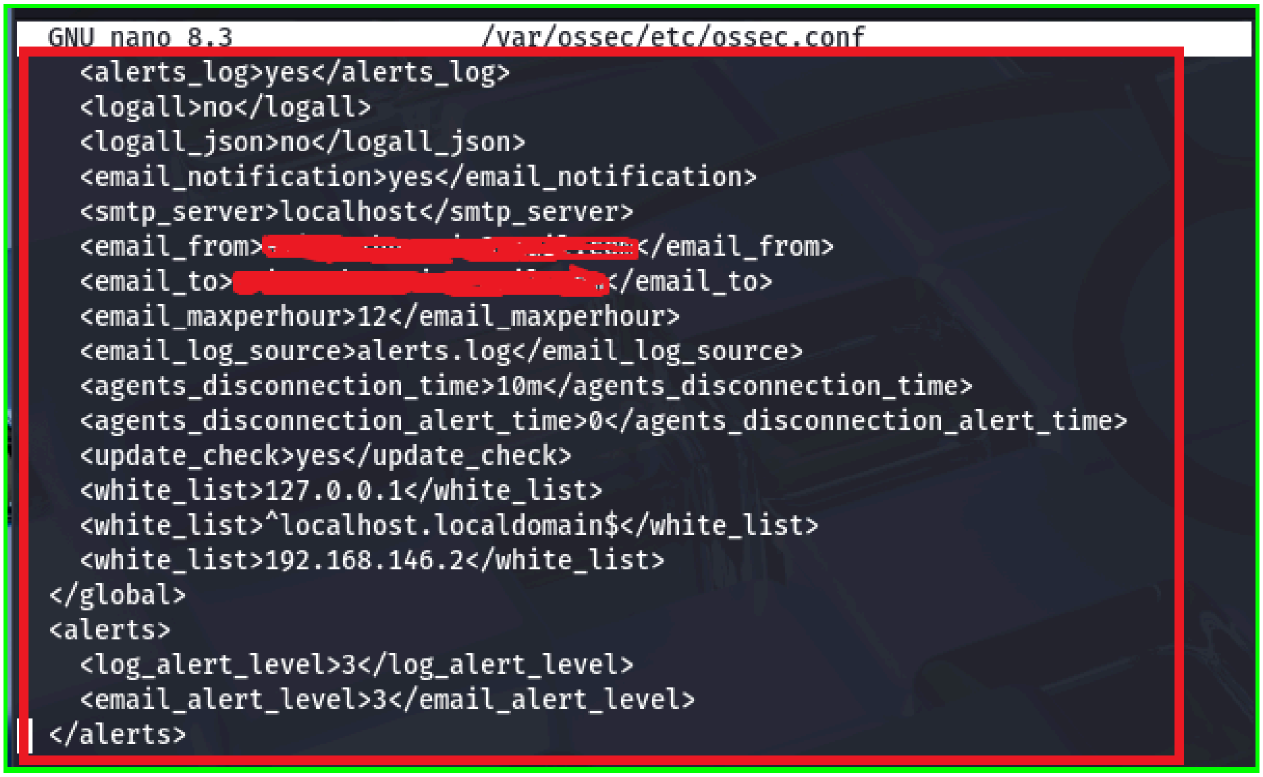
```
┌──(kali㉿kali)-[~]
└─$ sudo nano /var/ossec/etc/ossec.conf
[sudo] password for kali:
```

Inside `<global>` add:

```
<global>
 <email_notification>yes</email_notification>
 <email_to>yourgmail@gmail.com</email_to>
 <email_from>yourgmail@gmail.com</email_from>
 <smtp_server>localhost</smtp_server>
 <email_maxperhour>12</email_maxperhour>
</global>
```

Inside `<alerts>` add:

```
<alerts>
 <log_alert_level>3</log_alert_level>
 <email_alert_level>3</email_alert_level>
</alerts>
```

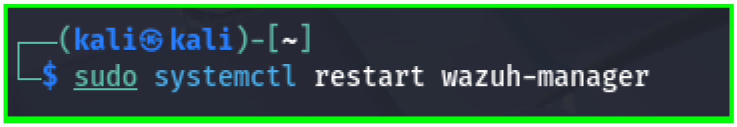


```
  GNU nano 8.3                    /var/ossec/etc/ossec.conf
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <logall_json>no</logall_json>
    <email_notification>yes</email_notification>
    <smtp_server>localhost</smtp_server>
    <email_from>[redacted]</email_from>
    <email_to>[redacted]</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
    <white_list>127.0.0.1</white_list>
    <white_list>^localhost.localdomain$</white_list>
    <white_list>192.168.146.2</white_list>
  </global>
  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>3</email_alert_level>
  </alerts>
```

This means Wazuh sends alerts through postfix (running on localhost).

Step 6: Restart Wazuh Manager

```
sudo systemctl restart wazuh-manager
```

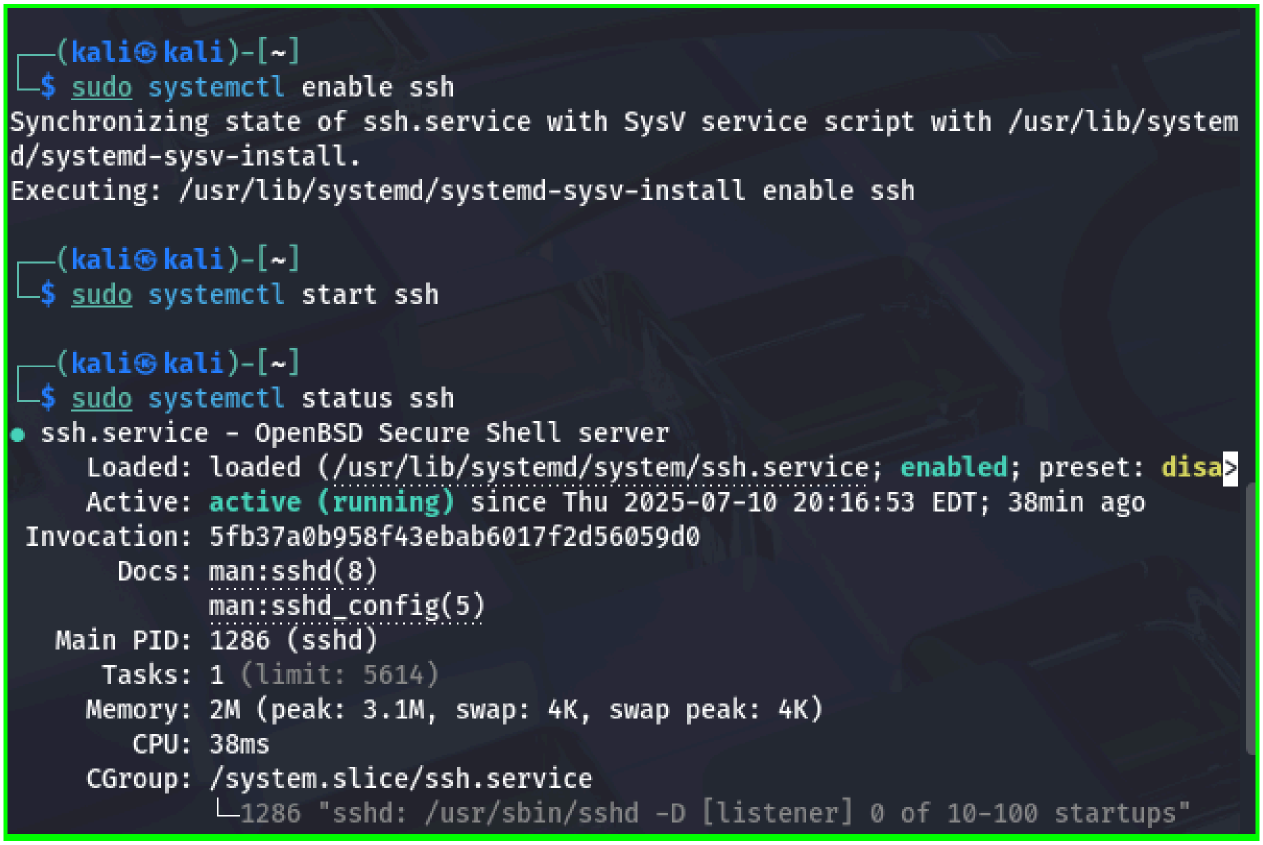


```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart wazuh-manager
```

Step 7 : Simulate a Security Alert (Fake SSH Login)

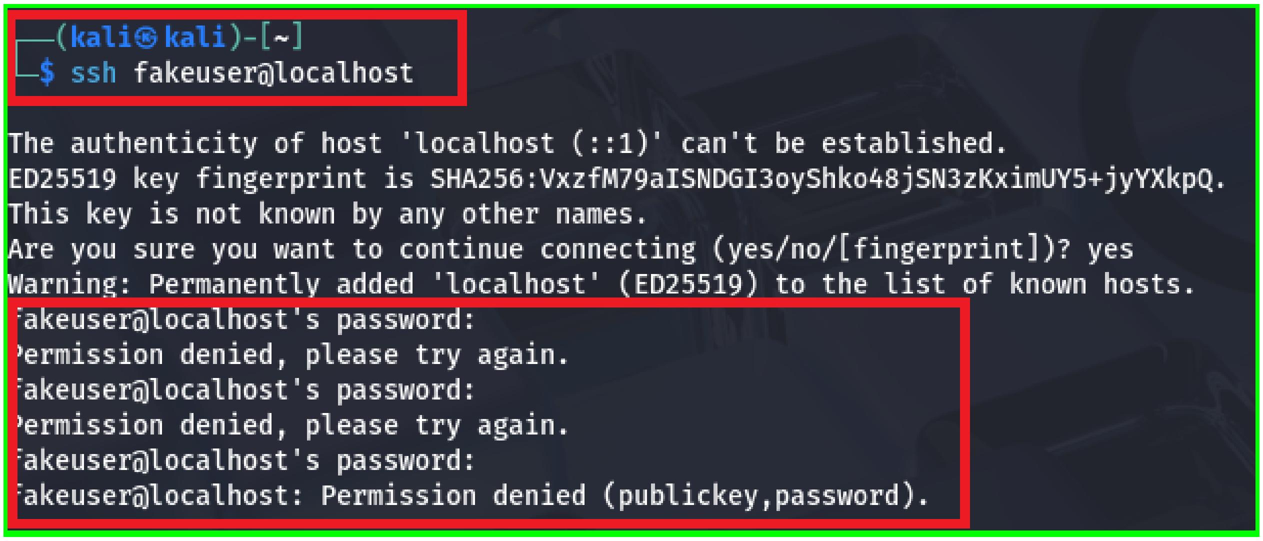
Make sure SSH is running:

```
sudo apt install openssh-server
sudo systemctl enable ssh
sudo systemctl start ssh
sudo systemctl status ssh
```



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /usr/lib/system
d/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh

┌──(kali㉿kali)-[~]
└─$ sudo systemctl start ssh

┌──(kali㉿kali)-[~]
└─$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: disa>
     Active: active (running) since Thu 2025-07-10 20:16:53 EDT; 38min ago
 Invocation: 5fb37a0b958f43ebab6017f2d56059d0
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 1286 (sshd)
      Tasks: 1 (limit: 5614)
     Memory: 2M (peak: 3.1M, swap: 4K, swap peak: 4K)
        CPU: 38ms
     CGroup: /system.slice/ssh.service
             └─1286 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

Attempt a fake login:
ssh fakeuser@localhost



```
┌──(kali㉿kali)-[~]
└─$ ssh fakeuser@localhost

The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:VxzfM79aISNDGI3oyShko48jSN3zKximUY5+jyYXkpQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
fakeuser@localhost's password:
Permission denied, please try again.
fakeuser@localhost's password:
Permission denied, please try again.
fakeuser@localhost's password:
fakeuser@localhost: Permission denied (publickey,password).
```

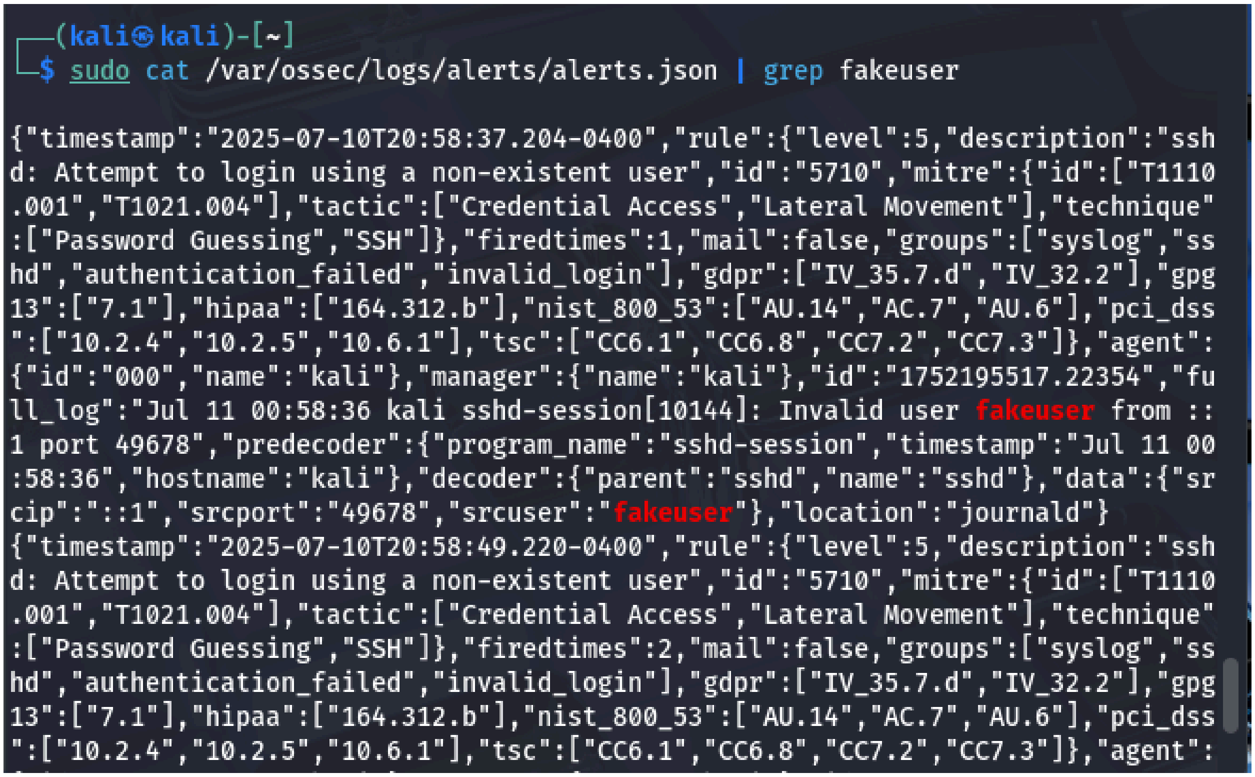
When it asks for a password, type any wrong password several times (3–4 times is enough).

Expected result:
Since the user `fakeuser` does not exist, the system will register failed login attempts.

Find the alerts log

Run the following command:

sudo cat /var/ossec/logs/alerts/alerts.json | grep fakeuser



```
┌──(kali㉿kali)-[~]
└─$ sudo cat /var/ossec/logs/alerts/alerts.json | grep fakeuser

{"timestamp":"2025-07-10T20:58:37.204-0400","rule":{"level":5,"description":"ssh
d: Attempt to login using a non-existent user","id":"5710","mitre":{"id":["T1110
.001","T1021.004"],"tactic":["Credential Access","Lateral Movement"],"technique"
:["Password Guessing","SSH"]},"firedtimes":1,"mail":false,"groups":["syslog","ss
hd","authentication_failed","invalid_login"],"gdpr":["IV_35.7.d","IV_32.2"],"gpg
13":["7.1"],"hipaa":["164.312.b"],"nist_800_53":["AU.14","AC.7","AU.6"],"pci_dss
":["10.2.4","10.2.5","10.6.1"],"tsc":["CC6.1","CC6.8","CC7.2","CC7.3"]},"agent":
{"id":"000","name":"kali"},"manager":{"name":"kali"},"id":"1752195517.22354","fu
ll_log":"Jul 11 00:58:36 kali sshd-session[10144]: Invalid user fakeuser from ::
1 port 49678","predecoder":{"program_name":"sshd-session","timestamp":"Jul 11 00
:58:36","hostname":"kali"},"decoder":{"parent":"sshd","name":"sshd"},"data":{"sr
cip":"::1","srcport":"49678","srcuser":"fakeuser"},"location":"journald"}
{"timestamp":"2025-07-10T20:58:49.220-0400","rule":{"level":5,"description":"ssh
d: Attempt to login using a non-existent user","id":"5710","mitre":{"id":["T1110
.001","T1021.004"],"tactic":["Credential Access","Lateral Movement"],"technique"
:["Password Guessing","SSH"]},"firedtimes":2,"mail":false,"groups":["syslog","ss
hd","authentication_failed","invalid_login"],"gdpr":["IV_35.7.d","IV_32.2"],"gpg
13":["7.1"],"hipaa":["164.312.b"],"nist_800_53":["AU.14","AC.7","AU.6"],"pci_dss
":["10.2.4","10.2.5","10.6.1"],"tsc":["CC6.1","CC6.8","CC7.2","CC7.3"]},"agent":
```

If Wazuh caught the fake login, you will see something like:

"level":5, "sshd: Attempt to login using a non-existent user"

If yes — it worked!

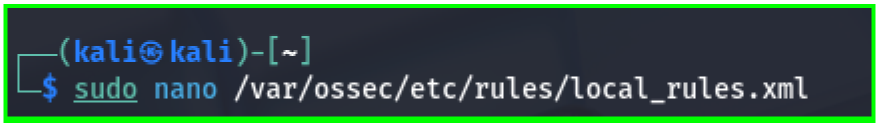
Step 9: Enable Email for This Alert (Custom Rule)

Sometimes, even if Wazuh detects an alert, it doesn't actually send an email.
This usually happens because some rules aren't set to trigger email notifications by default.

So, we'll add our own custom rule to make sure Wazuh sends an email whenever it sees someone trying to log in with a fake username.

Open your local Wazuh rules file:

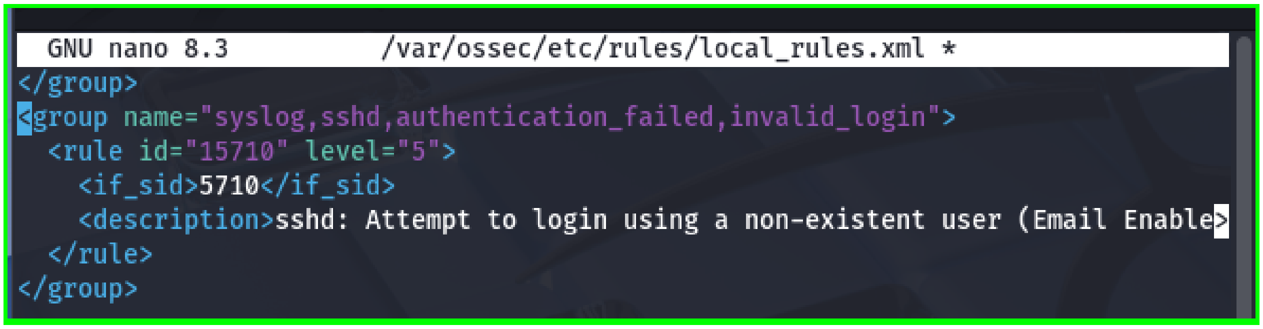
```
sudo nano /var/ossec/etc/rules/local_rules.xml
```



```
┌──(kali㉿kali)-[~]
└─$ sudo nano /var/ossec/etc/rules/local_rules.xml
```

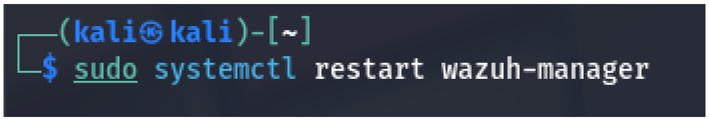
Add this rule

```
<group name="syslog,sshd,authentication_failed,invalid_login">
  <rule id="15710" level="5">
    <if_sid>5710</if_sid>
    <description>sshd: Attempt to login using a non-existent user (Email Enabled)</description>
  </rule>
</group>
```



```
  GNU nano 8.3             /var/ossec/etc/rules/local_rules.xml *
</group>
<group name="syslog,sshd,authentication_failed,invalid_login">
  <rule id="15710" level="5">
    <if_sid>5710</if_sid>
    <description>sshd: Attempt to login using a non-existent user (Email Enable>
  </rule>
</group>
```

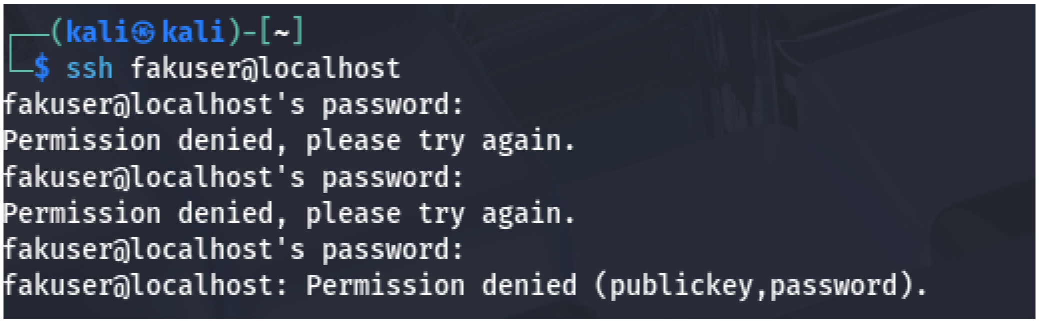
Then, restart wazuh manager.



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart wazuh-manager
```

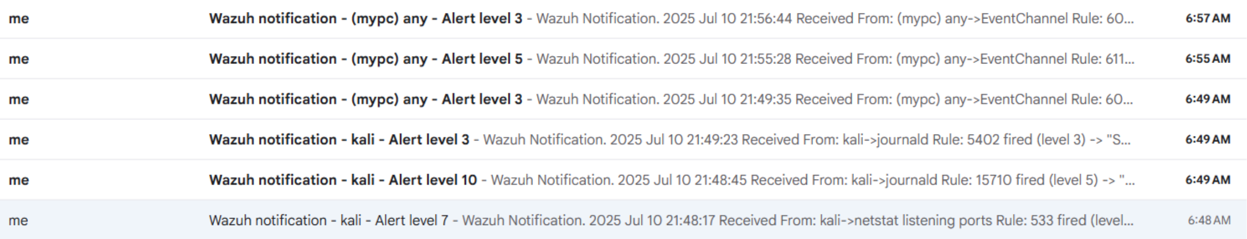
Test Again.

```
ssh fakeuser@localhost
```



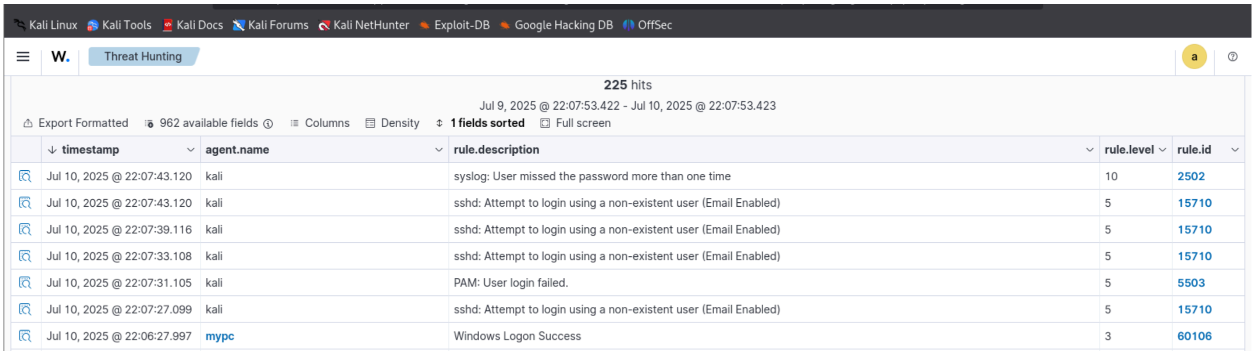
```
┌──(kali㉿kali)-[~]
└─$ ssh fakuser@localhost
fakuser@localhost's password:
Permission denied, please try again.
fakuser@localhost's password:
Permission denied, please try again.
fakuser@localhost's password:
fakuser@localhost: Permission denied (publickey,password).
```

You can verify that email notifications are being sent correctly by reviewing your email inbox or checking the mail logs.



me	**Wazuh notification - (mypc) any - Alert level 3** - Wazuh Notification. 2025 Jul 10 21:56:44 Received From: (mypc) any->EventChannel Rule: 60...	6:57 AM
me	**Wazuh notification - (mypc) any - Alert level 5** - Wazuh Notification. 2025 Jul 10 21:55:28 Received From: (mypc) any->EventChannel Rule: 611...	6:55 AM
me	**Wazuh notification - (mypc) any - Alert level 3** - Wazuh Notification. 2025 Jul 10 21:49:35 Received From: (mypc) any->EventChannel Rule: 60...	6:49 AM
me	**Wazuh notification - kali - Alert level 3** - Wazuh Notification. 2025 Jul 10 21:49:23 Received From: kali->journald Rule: 5402 fired (level 3) -> "S...	6:49 AM
me	**Wazuh notification - kali - Alert level 10** - Wazuh Notification. 2025 Jul 10 21:48:45 Received From: kali->journald Rule: 15710 fired (level 5) -> "...	6:49 AM
me	Wazuh notification - kali - Alert level 7 - Wazuh Notification. 2025 Jul 10 21:48:17 Received From: kali->netstat listening ports Rule: 533 fired (level...	6:48 AM

You also see logs in wazuh manager in events section.



225 hits

Jul 9, 2025 @ 22:07:53.422 - Jul 10, 2025 @ 22:07:53.423

timestamp	agent.name	rule.description	rule.level	rule.id
Jul 10, 2025 @ 22:07:43.120	kali	syslog: User missed the password more than one time	10	2502
Jul 10, 2025 @ 22:07:43.120	kali	sshd: Attempt to login using a non-existent user (Email Enabled)	5	15710
Jul 10, 2025 @ 22:07:39.116	kali	sshd: Attempt to login using a non-existent user (Email Enabled)	5	15710
Jul 10, 2025 @ 22:07:33.108	kali	sshd: Attempt to login using a non-existent user (Email Enabled)	5	15710
Jul 10, 2025 @ 22:07:31.105	kali	PAM: User login failed.	5	5503
Jul 10, 2025 @ 22:07:27.099	kali	sshd: Attempt to login using a non-existent user (Email Enabled)	5	15710
Jul 10, 2025 @ 22:06:27.997	mypc	Windows Logon Success	3	60106

Final Result

Congratulations! The Wazuh server is now fully set up to:

Detect and log real-time security threats (such as failed SSH login attempts)
Automatically send email alerts to Gmail using Postfix as the mail relay
Support further customization by adjusting alert levels or adding new detection rules

Summary

The setup involved:

Installing and configuring Postfix to relay emails through Gmail securely
Creating and protecting the `sasl_passwd` file containing Gmail SMTP credentials
Editing the Wazuh configuration (`ossec.conf`) to enable email notifications and define alert thresholds
Adding a custom rule in `local_rules.xml` to ensure specific SSH login failures trigger an email alert
Testing by simulating failed SSH login attempts and confirming alerts arrived in Gmail
As a result, the system now provides real-time monitoring and immediate alerts for suspicious activities, improving overall security visibility and response.