LAB — SOLUTION

Securing a Database Server Environment

Systems & Database Administration

Debian 12 · UFW · Fail2ban · PostgreSQL

Part 1 — Virtual Machine Setup

Create the VM in VirtualBox with the following specifications, then boot from the Debian 12 ISO.

Setting	Required value
RAM	2 048 MB (2 GB minimum)
CPU	2 virtual cores
Disk	20 GB, dynamic allocation
Network adapter	Bridged
Boot image	Debian 12 (Bookworm) minimal ISO

Tasks 1 & 2 — Create VM and boot installer

In VirtualBox: New → Type: Linux → Version: Debian (64-bit). Attach the Debian ISO to the optical drive and start the installation wizard.

Part 2 — Debian Installation & Hardening

Perform a minimal installation (no GUI), then apply hardening tasks.

Installation guidelines

- Hostname: db-server
- Non-root admin account: dbadmin
- Separate partitions: /, /var, /home, /tmp, swap
- Install SSH server + standard utilities only

Task 3 — Full system update

```
apt update && apt upgrade -y
apt install -y sudo curl wget gnupg2
usermod -aG sudo dbadmin
```

Task 4 — Disable unnecessary services

```
# List active services
systemctl list-units --type=service --state=running

# Disable services not needed (examples)
systemctl disable --now bluetooth avahi-daemon cups 2>/dev/null || true
```

Task 5 — Harden SSH

Edit the SSH daemon configuration file:

```
nano /etc/ssh/sshd_config
```

Apply these settings:

```
Port 2222                       # Custom port
```

```
PermitRootLogin no            # Disable root login
AllowUsers dbadmin            # Restrict to dbadmin only
PasswordAuthentication no     # Key-based auth only
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
X11Forwarding no
MaxAuthTries 3
```

Generate and copy the key from the host machine:

```
# On HOST machine
ssh-keygen -t ed25519 -C "dbadmin@db-server"
ssh-copy-id -p 2222 dbadmin@<IP_VM>

# On SERVER — restart SSH
systemctl restart sshd
```

Task 6 — Enable automatic security updates

> **Enable automatic updates BEFORE configuring the firewall.**

```
apt install -y unattended-upgrades apt-listchanges
dpkg-reconfigure -plow unattended-upgrades   # Answer: Yes

# Verify
cat /etc/apt/apt.conf.d/20auto-upgrades
```

Expected content:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

Task 7 — Verify SSH on new port

```
# From HOST machine
ssh -p 2222 dbadmin@<IP_VM>
```

Part 3 — Firewall Configuration (UFW)

> **Always add the SSH rule BEFORE enabling UFW to avoid locking yourself out.**

Task 8 — Install UFW

```
apt install -y ufw
```

Task 9 — Default policy

```
ufw default deny incoming
ufw default allow outgoing
```

Task 10 — Allow SSH with rate-limit

```
ufw limit 2222/tcp comment 'SSH custom port with rate-limit'
```

Task 11 — PostgreSQL from localhost only

```
ufw allow from 127.0.0.1 to any port 5432 proto tcp comment 'PostgreSQL localhost
only'
```

Task 12 — Enable UFW and verify rules

```
ufw enable
ufw status verbose
```

Expected output:

```
Status: active
To                         Action      From
--                         ------      ----
2222/tcp                   LIMIT IN    Anywhere
5432/tcp                   ALLOW IN    127.0.0.1
```

Task 13 — Port scan from host

```
# On HOST machine
nmap -p 5432 <IP_VM>
```

Expected result:

```
5432/tcp  filtered  postgresql
```

Task 14 — Enable UFW logging

```
ufw logging on
# Logs appear in /var/log/ufw.log
```

Part 4 — Brute-Force Protection (Fail2ban)

Fail2ban monitors log files and bans IPs showing repeated authentication failures.

Task 15 — Install Fail2ban

```
apt install -y fail2ban
systemctl enable --now fail2ban
```

Task 16 — Create local configuration file

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Never edit jail.conf directly — it is overwritten on package upgrades.

Task 17 — Configure SSH jail

Edit /etc/fail2ban/jail.local — find or add the [sshd] section:

```
nano /etc/fail2ban/jail.local
```

```
[sshd]
enabled   = true
port      = 2222
filter    = sshd
logpath   = /var/log/auth.log
maxretry  = 3
bantime   = 7200    # 2 hours in seconds
findtime  = 600
```

Task 18 — Configure PostgreSQL jail

Create the custom filter:

```
nano /etc/fail2ban/filter.d/postgresql.conf
```

```
[Definition]
failregex = ^.*FATAL:  password authentication failed for user.*$
            ^.*FATAL:  no pg_hba.conf entry for host "<HOST>".*$
ignoreregex =
```

Add the PostgreSQL jail in jail.local:

```
[postgresql]
enabled   = true
port      = 5432
filter    = postgresql
logpath   = /var/log/postgresql/postgresql-*.log
maxretry  = 3
bantime   = 7200
findtime  = 600
```

Task 19 — Restart and verify

```
systemctl restart fail2ban
fail2ban-client status
fail2ban-client status sshd
fail2ban-client status postgresql
```

Task 20 — Simulate SSH brute-force attack

```
# From a TEST machine — run 3+ failed login attempts
ssh -p 2222 root@<IP_VM>   # attempt 1
ssh -p 2222 root@<IP_VM>   # attempt 2
ssh -p 2222 root@<IP_VM>   # attempt 3 → BAN triggered

# On SERVER — verify the ban
fail2ban-client status sshd
```

Task 21 — Unban the test IP

```
fail2ban-client set sshd unbanip <IP_MACHINE_TEST>
fail2ban-client status sshd   # Confirm released
```

Part 5 — PostgreSQL Installation & Security

Task 22 — Install PostgreSQL

```
apt install -y postgresql postgresql-contrib
```

Task 23 — Verify service and version

```
systemctl status postgresql
psql --version
# Or inside psql:
psql -U postgres -c "SELECT version();"
```

Task 24 — Strong password for postgres superuser

```
psql -U postgres
```

```
ALTER USER postgres WITH PASSWORD 'P@ssw0rd_Str0ng!2025';
\q
```

Task 25 — Create application user and database

```
psql -U postgres
```

```
CREATE USER appuser WITH PASSWORD 'AppUser_S3cur3!';
CREATE DATABASE appdb OWNER appuser;
GRANT CONNECT ON DATABASE appdb TO appuser;
\q
```

Task 26 — Configure postgresql.conf

```
nano /etc/postgresql/15/main/postgresql.conf
```

```
listen_addresses = 'localhost'      # Localhost only
ssl = on                            # Enable SSL
log_connections = on
log_disconnections = on
log_failed_connections = on
log_line_prefix = '%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h '
logging_collector = on
log_directory = 'pg_log'
log_filename = 'postgresql-%Y-%m-%d.log'
```

Task 27 — Configure pg_hba.conf

```
nano /etc/postgresql/15/main/pg_hba.conf
```

Replace entire content with:

```
# TYPE  DATABASE    USER      ADDRESS          METHOD
local   all         postgres                   peer
local   all         all                        scram-sha-256
host    appdb       appuser   127.0.0.1/32     scram-sha-256
hostssl appdb       appuser   127.0.0.1/32     scram-sha-256
```

Remove ALL lines with 'trust' or 'md5'. Never use trust authentication in production.

Task 28 — Self-signed SSL certificate

```
cd /etc/postgresql/15/main/

openssl req -new -x509 -days 365 -nodes \
  -out server.crt \
  -keyout server.key \
  -subj "/CN=db-server"

chmod 600 server.key
chown postgres:postgres server.crt server.key
```

Add to postgresql.conf:

```
ssl_cert_file = 'server.crt'
ssl_key_file  = 'server.key'
```

```
systemctl restart postgresql
```

Task 29 — Revoke PUBLIC privileges

```
psql -U postgres -d appdb
```

```
REVOKE ALL ON DATABASE appdb FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM PUBLIC;
GRANT USAGE ON SCHEMA public TO appuser;
\q
```

Task 30 — Test SSL connection

```
psql "host=localhost dbname=appdb user=appuser sslmode=require"

# Inside psql
\conninfo
-- Expected: SSL connection (protocol: TLSv1.3, ...)
```

Task 31 — Simulate failed logins and verify

```
# Trigger 5 failed attempts
for i in $(seq 1 5); do
  PGPASSWORD=WRONG psql -h localhost -U appuser -d appdb 2>/dev/null
done

# Check PostgreSQL logs
tail -f /var/log/postgresql/postgresql-*.log

# Check Fail2ban
fail2ban-client status postgresql
```

Reflection Questions

Q32 — Why change the default SSH port? What are its limits?

Changing port 22 to a non-standard port (e.g. 2222) drastically reduces noise from automated scanners and bots that blindly target port 22. It is a 'security through obscurity' measure.

Limits:

- A determined attacker running a full port scan (nmap -p-) will find the port.
- It offers no real cryptographic protection — it only reduces the attack surface from script-kiddies.
- Must be combined with key-based auth, Fail2ban, and firewall rules for real security.

Q33 — md5 vs scram-sha-256 in pg_hba.conf

Criterion	md5	scram-sha-256
Algorithm	MD5 (obsolete)	SCRAM-SHA-256 (RFC 5802)
Vulnerability	Prone to rainbow table attacks	Resistant to replay attacks
Transmission	MD5 hash sent over network	Challenge-response; password never sent
Recommendation	Not recommended in production	PostgreSQL recommended standard (v10+)

Q34 — listen_addresses = '*' without firewall?

PostgreSQL would listen on ALL network interfaces, exposing port 5432 to every machine on the network or Internet. Without a firewall rule blocking external access, any host could attempt to connect to the database server directly, drastically increasing the attack surface.

Q35 — How does Fail2ban interact with UFW?

Fail2ban watches log files (e.g. /var/log/auth.log, PostgreSQL logs) for patterns matching failed authentications. When the maxretry threshold is exceeded within findtime seconds, Fail2ban calls UFW via iptables/nftables to insert a DROP rule for the offending IP address, blocking all further traffic from it for the bantime duration. The ban is automatically lifted after bantime expires, or manually via fail2ban-client set <jail> unbanip <IP>.