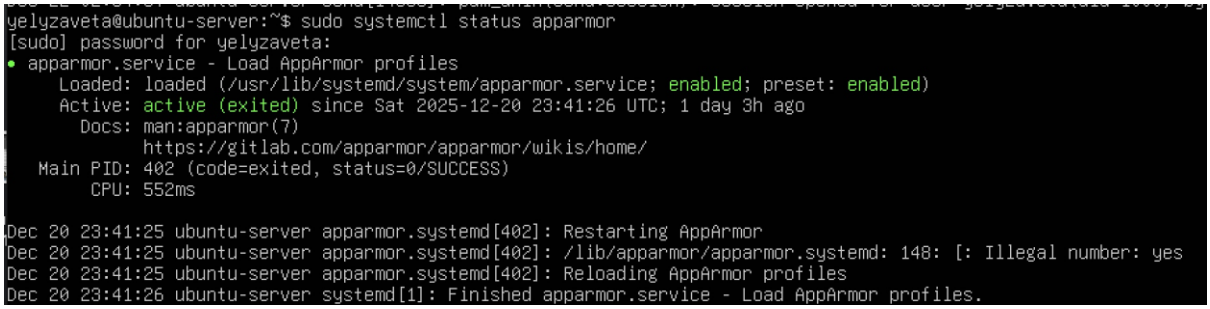
Week 9 – Assessment week 5

Access Control Using AppArmor

AppArmor was used to implement mandatory access control on the Ubuntu server. AppArmor restricts program capabilities by enforcing security profiles, reducing the impact of potential vulnerabilities and unauthorised access. Ubuntu uses AppArmor by default, making it a suitable and stable choice for access control enforcement in this environment.

Step 1



The status of AppArmor was checked to confirm that the service is installed and active.

Step 2

```
   qutebrowser
   rootlesskit
   rpm
   rssguard
   runc
   sbuild
   sbuild-abort
   sbuild-adduser
   sbuild-apt
   sbuild-checkpackages
   sbuild-clean
   sbuild-createchroot
   sbuild-destroychroot
   sbuild-distupgrade
   sbuild-hold
   sbuild-shell
   sbuild-unhold
   sbuild-update
   sbuild-upgrade
   scide
   signal-desktop
   slack
   slirp4netns
   steam
   stress-ng
   surfshark
   systemd-coredump
   thunderbird
   toybox
   trinity
   tup
   tuxedo-control-center
   userbindmount
   uwsgi-core
   vdens
   virtiofsd
   vivaldi-bin
   vpnns
   vscode
   wike
   wpcom
1 processes have profiles defined.
1 processes are in enforce mode.
   /usr/sbin/rsyslogd (5270) rsyslogd
0 processes are in complain mode.
0 processes are in prompt mode.
0 processes are in kill mode.
0 processes are unconfined but have a profile defined.
0 processes are in mixed mode.
yelyzaveta@ubuntu-server:~$
```

Loaded AppArmor profiles and their enforcement modes were displayed.

To ensure that AppArmor is actively enforcing access control policies, profiles were verified to be running in enforce mode.

Step 3

```
yelyzaveta@ubuntu-server:~$ ls /etc/apparmor.d
1password      element-desktop  local           obsidian            runc                signal-desktop        unix-chkpwd
abi            epiphany         loupe           opam                sbuild              slack                 unprivileged_userns
abstractions   evolution        lsb_release     opera               sbuild-abort        slirp4netns           userbindmount
balena-etcher  firefox          lxc-attach      pageedit            sbuild-adduser      steam                 usr.bin.man
brave          flatpak          lxc-create      plasmashell         sbuild-apt          stress-ng             usr.bin.tcpdump
buildah        foliate          lxc-destroy     podman              sbuild-checkpackages surfshark            usr.lib.snapd.snap-confine.real
busybox        force-complain   lxc-execute     polypane            sbuild-clean        systemd-coredump      usr.sbin.rsyslogd
cam            geary            lxc-stop        privacybrowser      sbuild-createchroot thunderbird           uwsgi-core
ch-checkns     github-desktop   lxc-unshare     qcam                sbuild-destroychroot toybox               vdens
chrome         goldendict       lxc-usernsexec  qmapshack           sbuild-distupgrade  transmission          virtiofsd
ch-run         ipa_verify       mmdebstrap      QtWebEngineProcess  sbuild-hold         trinity               vivaldi-bin
code           kchmviewer       MongoDB_Compass qutebrowser         sbuild-shell        tunables              vpnns
crun           keybase          msedge          rootlesskit         sbuild-unhold       tup                   wike
devhelp        lc-compliance    nautilus        rpm                 sbuild-update       tuxedo-control-center wpcom
disable        libcamerify      notepadqq       rssguard            sbuild-upgrade      ubuntu_pro_apt_news
Discord        linux-sandbox    nvidia_modprobe rsyslog.d           scide               ubuntu_pro_esm_cache
```

The AppArmor profiles directory was checked to confirm that security profiles are present on the system.

Outcome

AppArmor is enabled and actively enforcing access control policies on the server. Security profiles are loaded and running in enforce mode, ensuring that applications operate within defined security boundaries. This configuration enhances system security by limiting the potential impact of compromised services.

Task 2

Automatic security updates were configured on the Ubuntu server to ensure that critical security patches are installed without manual intervention.
This reduces the risk of vulnerabilities caused by outdated packages and improves overall system security. Ubuntu provides the unattended-upgrades mechanism to automatically apply security updates.

Step 1

```
yelyzaveta@ubuntu-server:~$ sudo apt update
[sudo] password for yelyzaveta:
Hit:1 http://ports.ubuntu.com/ubuntu-ports noble InRelease
Get:2 http://ports.ubuntu.com/ubuntu-ports noble-updates InRelease [126 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports noble-backports InRelease [126 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports noble-security InRelease [126 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 Components [172 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports noble-updates/restricted arm64 Components [212 B]
Get:7 http://ports.ubuntu.com/ubuntu-ports noble-updates/universe arm64 Components [376 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports noble-updates/multiverse arm64 Components [212 B]
Get:9 http://ports.ubuntu.com/ubuntu-ports noble-backports/main arm64 Components [3,556 B]
Get:10 http://ports.ubuntu.com/ubuntu-ports noble-backports/restricted arm64 Components [216 B]
Get:11 http://ports.ubuntu.com/ubuntu-ports noble-backports/universe arm64 Components [10.5 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports noble-backports/multiverse arm64 Components [212 B]
Get:13 http://ports.ubuntu.com/ubuntu-ports noble-security/main arm64 Components [18.4 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports noble-security/restricted arm64 Components [208 B]
Get:15 http://ports.ubuntu.com/ubuntu-ports noble-security/universe arm64 Components [71.4 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports noble-security/multiverse arm64 Components [208 B]
Fetched 1,032 kB in 0s (2,135 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
yelyzaveta@ubuntu-server:~$ sudo apt install unattended-upgrades -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
unattended-upgrades is already the newest version (2.9.1+nmu4ubuntu1).
unattended-upgrades set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

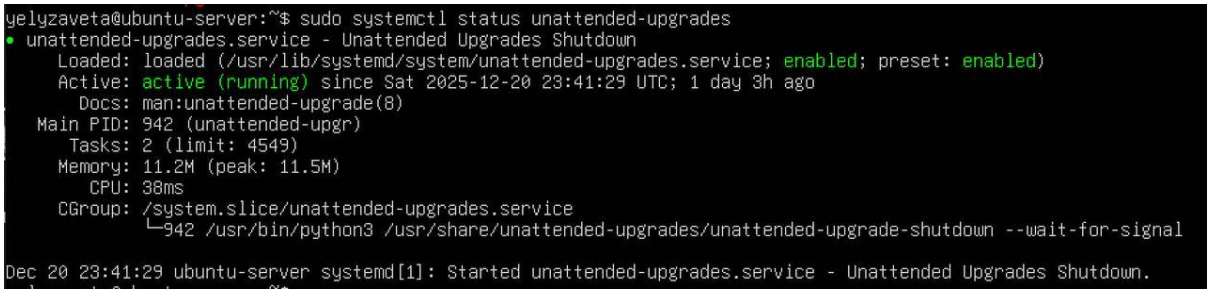
The required packages for automatic updates were installed.

Step 2

```
yelyzaveta@ubuntu-server:~$ cat /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

The configuration file was checked to confirm that security updates are enabled.

Step 3



The unattended-upgrades service status was checked to ensure it is active.

Outcome

Automatic security updates were successfully configured on the server. The system is now set to regularly install security updates without administrator intervention, ensuring timely patching of critical vulnerabilities.

Task 3 – fail2ban

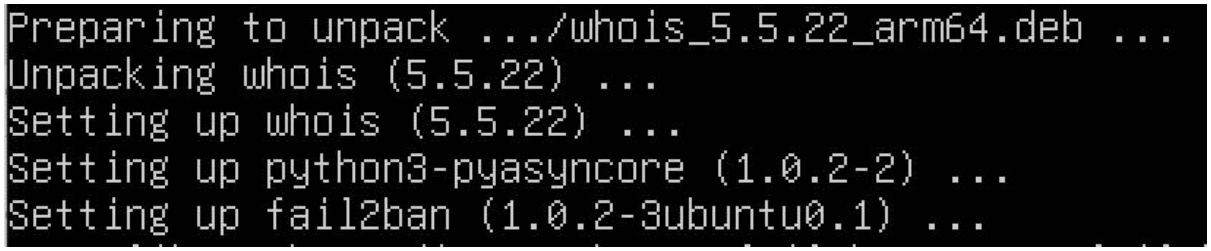
fail2ban was implemented on the Ubuntu server to protect against brute-force attacks and unauthorised access attempts. fail2ban monitors system log files and automatically blocks IP addresses that exhibit suspicious behaviour, such as repeated failed login attempts.

Step 1

Commands executed:

sudo apt update

sudo apt install fail2ban -y



The fail2ban package was installed from the official Ubuntu repositories.

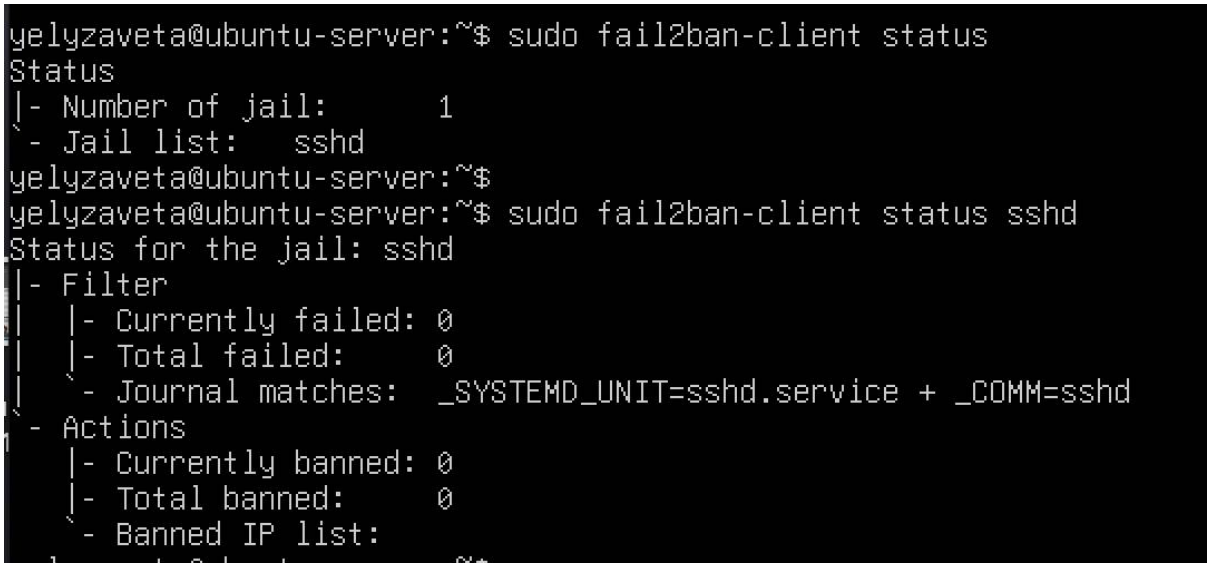
Step 2

The fail2ban service was enabled and started to ensure it runs automatically on system boot.

Step 3

The status of the fail2ban service was checked to confirm it is running correctly.

Step 4



The active fail2ban jails were checked to confirm that protection is enabled for SSH.

The SSH jail configuration was specifically checked to confirm that SSH is protected.

Outcome

fail2ban is successfully installed and actively monitoring the system for suspicious activity. The SSH service is protected by fail2ban, providing automated intrusion detection and response by blocking malicious IP addresses after repeated failed login attempts.

Task 4 - security-baseline.sh

Automatic security updates were configured on the Ubuntu server using the unattended-upgrades mechanism. This ensures that important security patches are installed automatically without manual intervention.

The unattended-upgrades package was installed and enabled using the system configuration tool. Configuration verification confirmed that automatic security updates are active and operational.

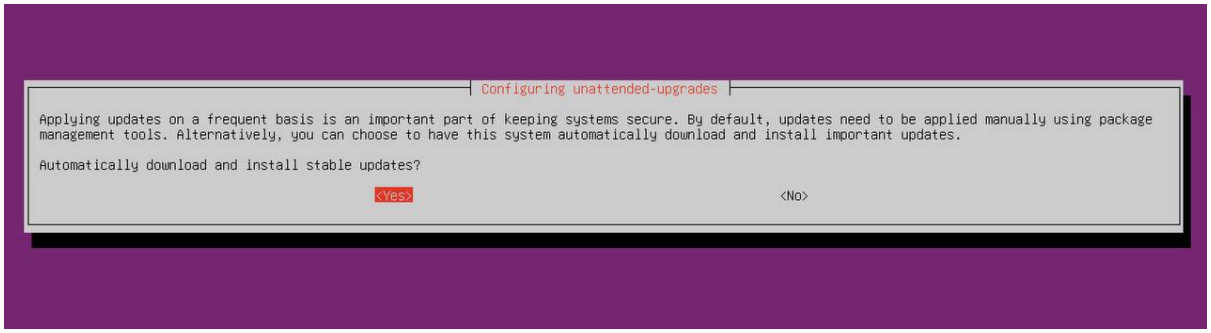
Step 1

sudo apt update

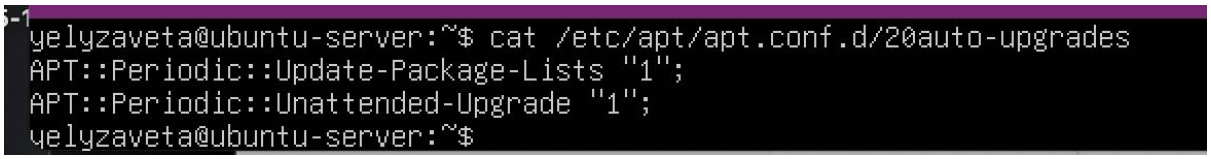
sudo apt install unattended-upgrades –y

Step 2

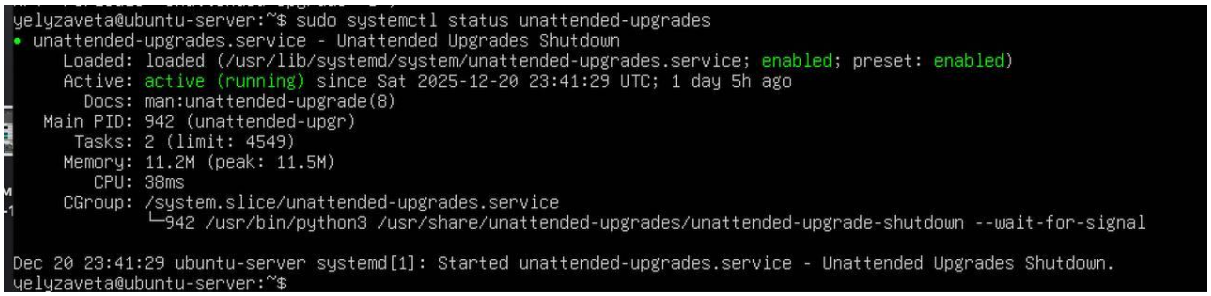
sudo dpkg-reconfigure unattended-upgrades



Step 3



Step 4



security-baseline.sh Verification Script

```
#!/bin/bash

echo "=== Security Baseline Verification ==="

echo "[1] Checking automatic updates (unattended-upgrades)"

systemctl is-enabled unattended-upgrades && systemctl is-active unattended-upgrades
```

```
echo "[2] Checking SSH service status"

systemctl is-active ssh

echo "[3] Checking firewall status (UFW)"

ufw status

echo "[4] Checking SSH root login configuration"

grep "^PermitRootLogin" /etc/ssh/sshd_config

echo "[5] Checking SSH key-based authentication"

grep "^PubkeyAuthentication" /etc/ssh/sshd_config

echo "[6] Checking allowed SSH users"

grep "^AllowUsers" /etc/ssh/sshd_config || echo "AllowUsers not set (default policy)"

echo "=== Verification completed ==="
```

This script validates that all security controls configured in previous phases remain active and correectly enforced. It provides a repeatable method for security baseline verification and supports ongoing system auditing.

Overview

This task focuses on improving system security by enabling automatic installation of security updates on the Ubuntu server. Automatic updates ensure that critical vulnerabilities are patched promptly without requiring manual administrator intervention. By configuring unattended security upgrades, the server remains protected against newly discovered threats while reducing administrative overhead. This approach supports long-term system stability and aligns with best practices for secure server management.

Task 5

Remote monitoring script (monitor-server.sh)

```
#!/bin/bash

SERVER_USER="yelyzaveta"
SERVER_IP="192.168.64.2 "
SSH_KEY="$HOME/.ssh/id_ed25519"

echo "Collecting system performance metrics from server..."
```

```
ssh -i "$SSH_KEY" ${SERVER_USER}@${SERVER_IP} << EOF
echo "===== CPU & Load ====="
uptime

echo "===== Memory Usage ====="
free -h

echo "===== Disk Usage ====="
df -h

echo "===== Disk I/O ====="
iostat -x 1 3

echo "===== Network ====="
ip -s link
EOF
```

Overview

The `monitor-server.sh` script is designed to perform **remote system monitoring** from the administrator's workstation. The script connects securely to the server using **SSH with key-based authentication** and executes standard Linux commands to collect real-time performance metrics.

The script retrieves information related to:

- System load and CPU activity (uptime)
- Memory usage and availability (free –h)
- Disk space utilisation (df –h)
- Disk I/O performance (iostat -x 1 3)
- Network interface statistics (ip -s link)

All commands are executed remotely on the server, ensuring that monitoring is performed without direct access to the system console. This approach reflects real-world administrative practices where servers are managed and monitored remotely.