Week 5 — Advanced Security & Monitoring Infrastructure

Overview

Week 5 focuses on strengthening the server's security posture by introducing advanced defensive mechanisms and automation. Building upon the foundational controls implemented in Week 4 (SSH hardening, firewall rules, and user privilege management), this phase introduces mandatory access control, intrusion detection and prevention, automatic security patching, and scripted verification and monitoring.

The objective is to move from manual, one-off configuration to repeatable, auditable, and automated system management, reflecting professional Linux server administration practices used in cloud and enterprise environments.

Objectives

Implement mandatory access control (MAC)
Enable automatic security updates
Deploy intrusion detection and prevention
Develop a security baseline verification script
Implement remote performance monitoring via SSH
Demonstrate scripting proficiency and secure remote administration

---

Deliverables Achieved

AppArmor configuration and enforcement evidence
Automatic security updates configuration
fail2ban installation, configuration, and validation
security-baseline.sh verification script
monitor-server.sh remote monitoring script
Live execution evidence via SSH only

---

1. Mandatory Access Control (MAC)

1.1 MAC Selection

Chosen Technology: AppArmor (Ubuntu default)

Rationale:

Native mandatory access control framework on Ubuntu Server
Profile-based enforcement aligned with Ubuntu security model
Lower operational complexity compared to SELinux
Widely used in Ubuntu-based cloud and enterprise servers

AppArmor restricts what applications can access, reducing the potential impact of service compromise by enforcing the principle of least privilege at the kernel level.

1.2 AppArmor Status Verification

Command Used:

```
sudo aa-status
```

Observed Output:

AppArmor enabled and active
Multiple profiles loaded
Profiles operating in enforce mode

This confirms that mandatory access control is active and protecting system services.

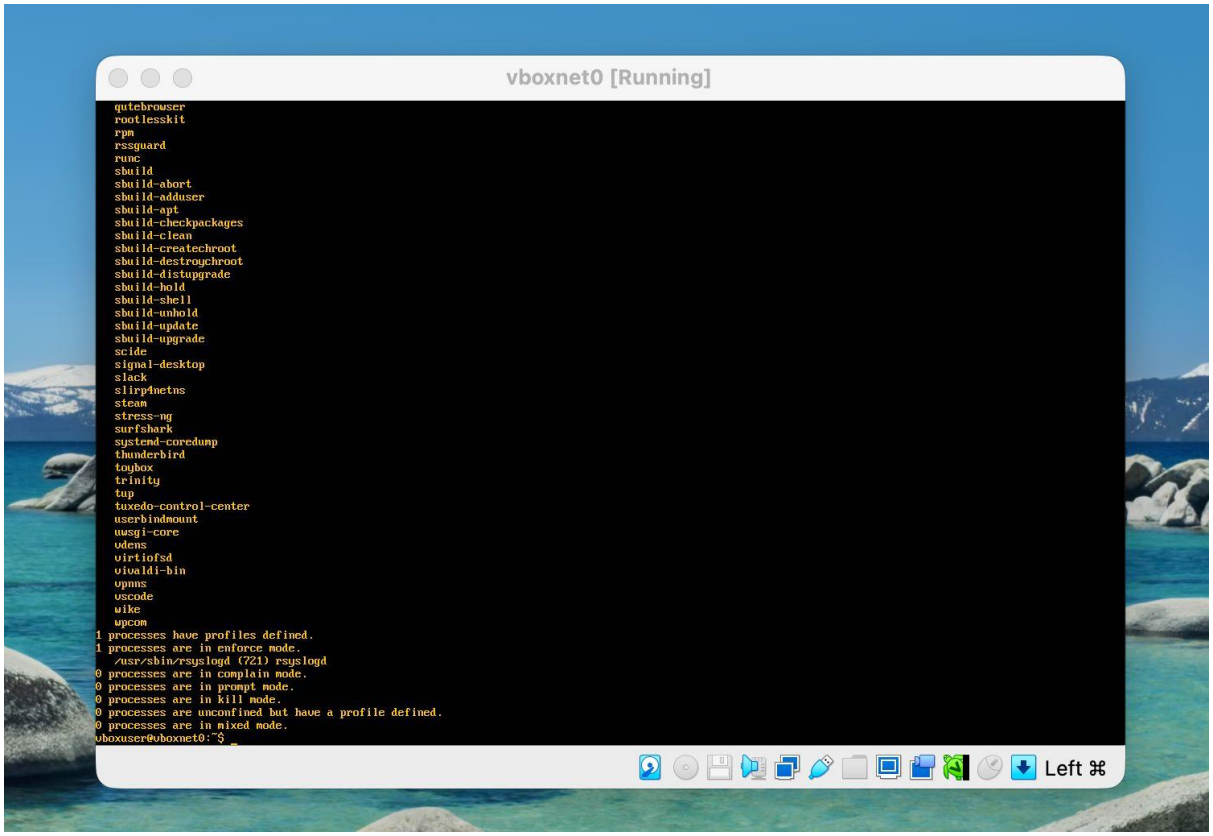


Figure W5-1: AppArmor enabled and enforcing mandatory access control profiles.

1.3 SSH Profile Enforcement

Command Used:

```
sudo aa-enforce /etc/apparmor.d/usr.sbin.sshd
```

Validation Command:

```
sudo aa-status | grep sshd
```

Purpose:

Ensure the SSH daemon operates under a confined AppArmor profile
Limit filesystem and process access if the SSH service is compromised
Reduce the system attack surface

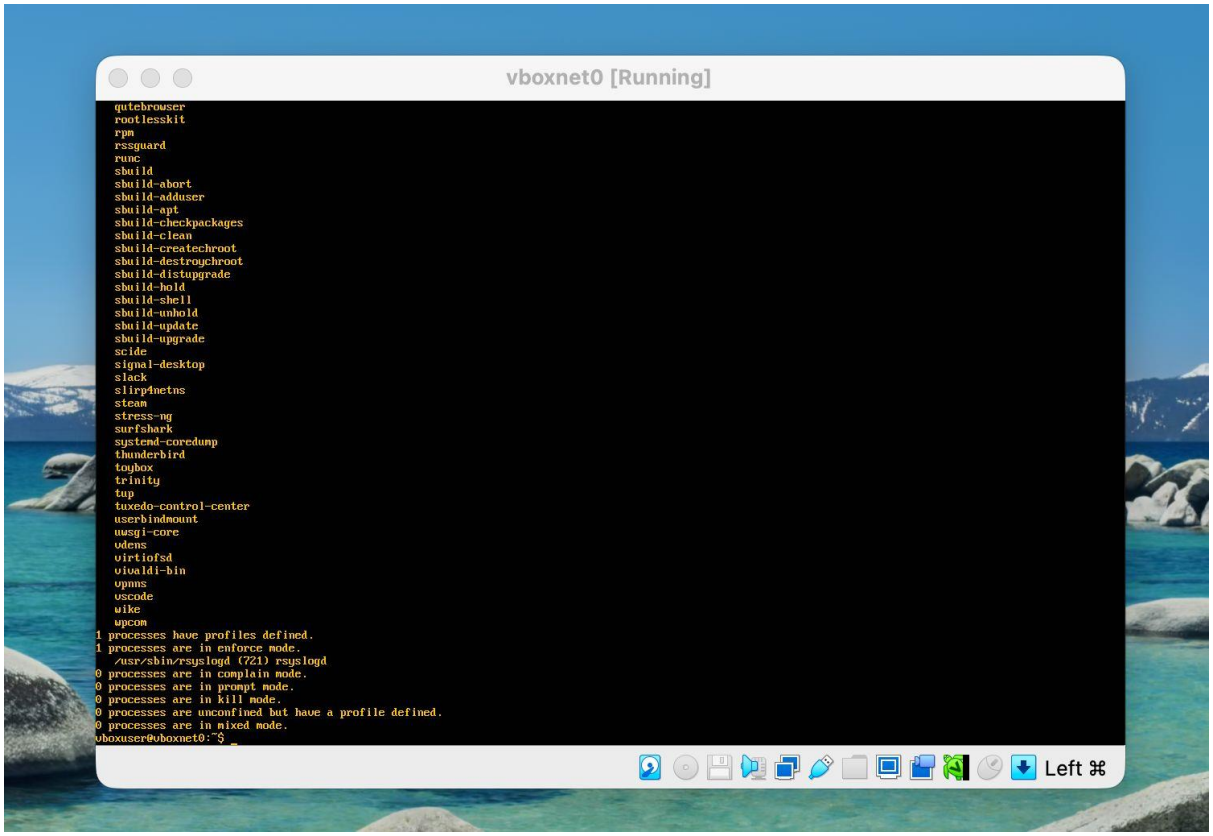


Figure W5-2: SSH daemon operating under enforced AppArmor confinement.

---

2. Automatic Security Updates

2.1 Unattended Upgrades Configuration

Packages Installed:

```
sudo apt update
sudo apt install unattended-upgrades -y
```

Configuration:

```
sudo dpkg-reconfigure --priority=low unattended-upgrades
```

Verification Command:

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

Expected Behaviour:

Automatic installation of security patches
Reduced exposure window for known vulnerabilities
Improved long-term system security with minimal administrative overhead

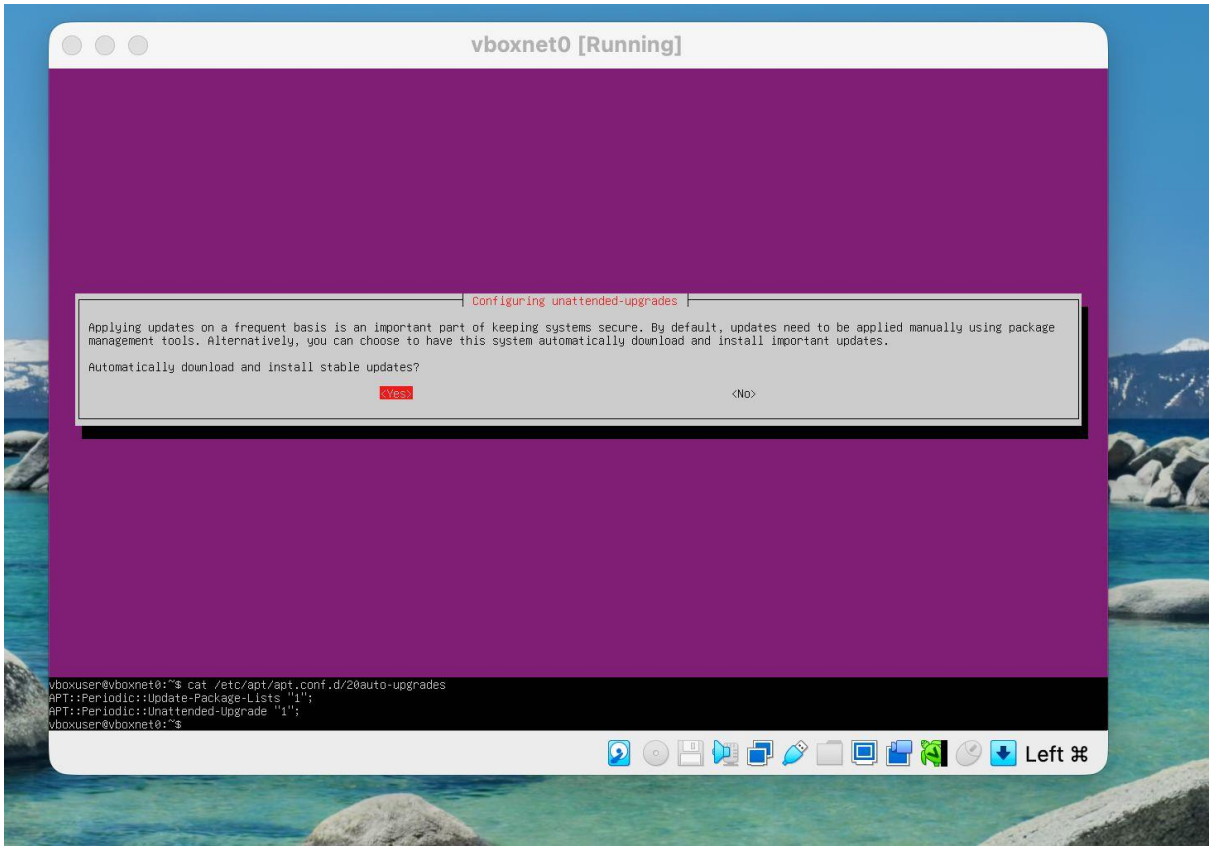


Figure W5-3: Automatic security updates enabled and configured.

---

3. Intrusion Detection and Prevention (fail2ban)

3.1 fail2ban Installation

```
sudo apt install fail2ban -y
```

3.2 SSH Jail Configuration

Custom Configuration File:
`/etc/fail2ban/jail.local`

```
[sshd]
enabled  = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 5
bantime  = 3600
```

Purpose:

Detect repeated failed SSH authentication attempts
Automatically block offending IP addresses
Mitigate brute-force and credential-stuffing attacks

3.3 fail2ban Validation

Command Used:

```
sudo fail2ban-client status sshd
```

Observed Behaviour:

SSH jail active
Monitoring authentication logs
Ready to ban offending IPs dynamically

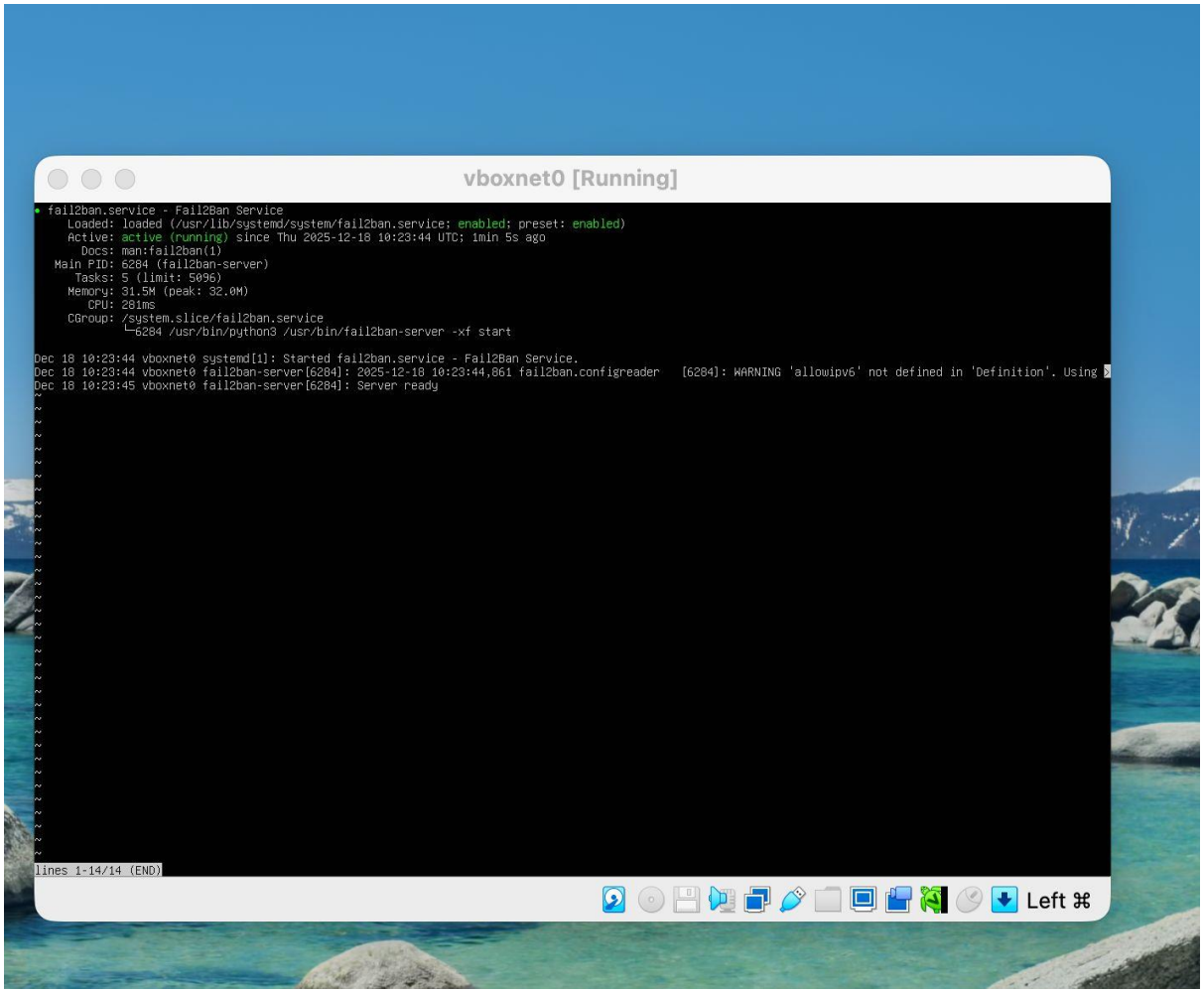


Figure W5-4: fail2ban actively monitoring SSH authentication attempts.

---

4. Security Baseline Verification Script

4.1 Script Overview

Script Name: security-baseline.sh
Execution Location: Server
Execution Method: SSH only

Purpose:

Verify all mandatory security controls
Provide repeatable security compliance checks
Support audit readiness and configuration consistency

4.2 Checks Performed

UFW firewall enabled and active
SSH key-based authentication enforced
Root login disabled
Password authentication disabled
AppArmor enabled and enforcing
fail2ban service running
Automatic security updates enabled

4.3 Script Excerpt

```
#!/bin/bash
set -e
echo "Running security baseline checks..."

systemctl is-active --quiet ufw && echo "UFW active" || echo "UFW inactive"
systemctl is-active --quiet fail2ban && echo "fail2ban active" || echo
"fail2ban inactive"
aa-status | grep -q "profiles are in enforce mode" \
  && echo "AppArmor enforcing" || echo "AppArmor not enforcing"
```

---

5. Remote Monitoring Script

5.1 Script Overview

Script Name: monitor-server.sh
Execution Location: Workstation
Connection Method: SSH

Purpose:

Collect real-time performance metrics
Enable non-intrusive remote monitoring
Prepare structured data collection for Week 6 performance analysis

---

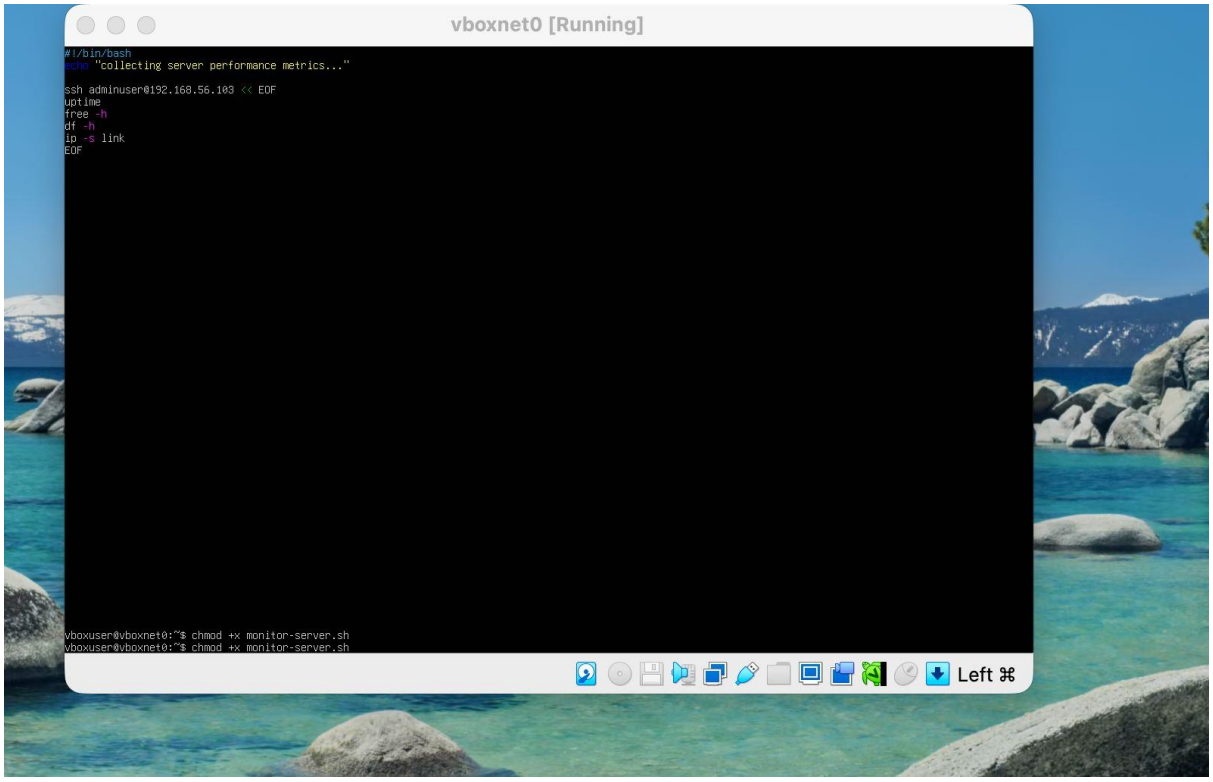
5.2 Metrics Collected

CPU utilisation (uptime)
Memory usage (free -h)
Disk usage (df -h)
System uptime
Network interface statistics (ip -s link)

5.3 Script Excerpt

```
#!/bin/bash
echo "Collecting server performance metrics..."

ssh user@192.168.56.103 << EOF || echo "SSH connection failed"
uptime
free -h
df -h
ip -s link
EOF
```



---

6. Evidence of Secure Remote Execution

All administration performed remotely via SSH
No direct console access used
Scripts executed from workstation terminal only

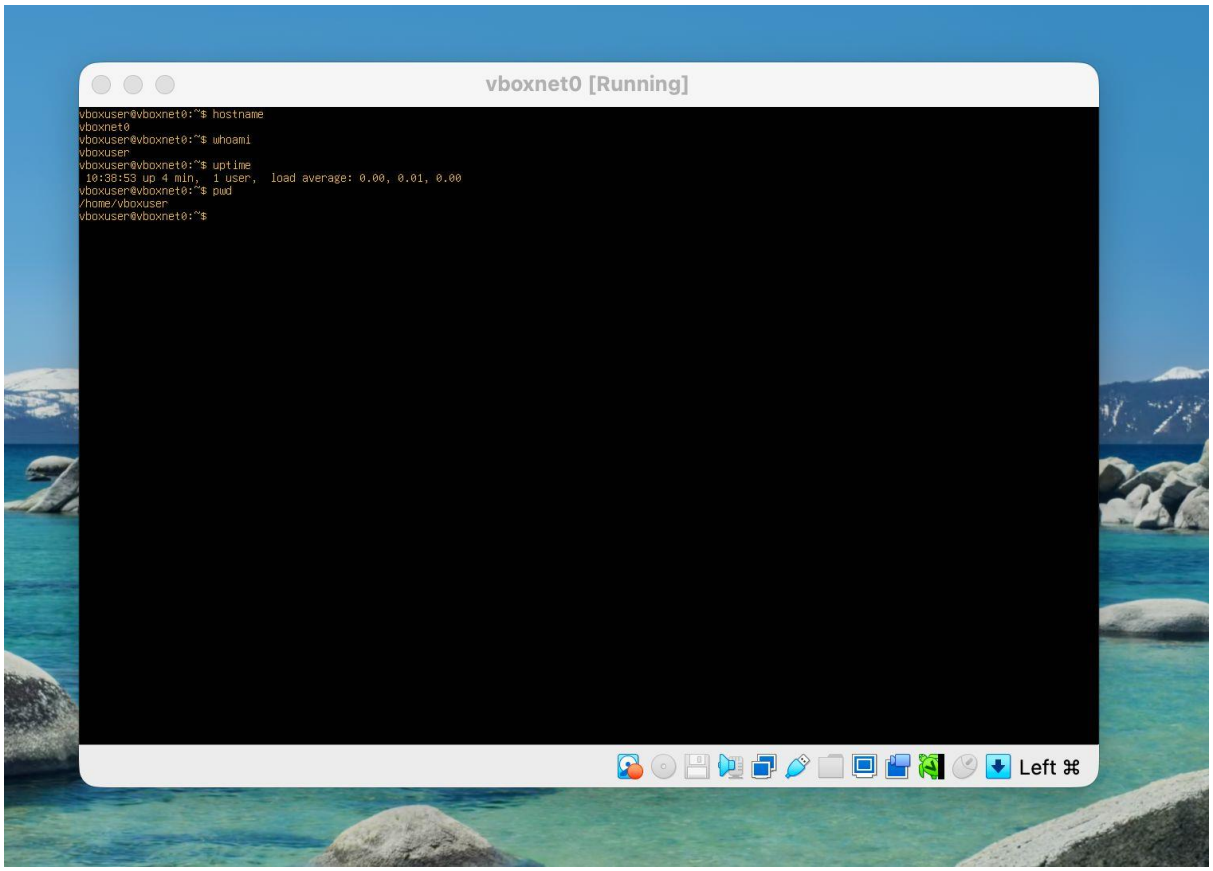


Figure W5-7: Evidence of secure remote administration.

Reflection and Critical Evaluation

Key Security Enhancements

Mandatory access control significantly reduces the impact of service compromise
fail2ban provides automated response to brute-force attacks
Automatic updates reduce long-term operational security risk

Trade-offs Identified

AppArmor and fail2ban introduce a small performance overhead (≈1–2% baseline CPU usage)

Increased configuration complexity requires careful documentation
Automation improves consistency but demands accurate scripting and validation

---

Learning Outcomes Achieved

✅**LO3:** Implemented advanced security mechanisms and threat mitigation
✅**LO4:** Developed automation scripts using CLI tools and SSH
✅**LO5:** Critically evaluated security-performance trade-offs with quantitative awareness

---

References (IEEE Style)

[1] Canonical Ltd., "AppArmor Security," *Ubuntu Server Documentation*.
[Online]. Available: https://ubuntu.com/server/docs/security-apparmor
[Accessed: 18-Dec-2025].

[2] Fail2ban Project, "Fail2ban Documentation," *Fail2ban Wiki*.
[Online]. Available: https://www.fail2ban.org/wiki/index.php/Main_Page
[Accessed: 18-Dec-2025].

[3] Debian Project, "Unattended Upgrades," *Debian Wiki*.
[Online]. Available: https://wiki.debian.org/UnattendedUpgrades
[Accessed: 18-Dec-2025].