Lab4: Bash Scripting for Security Automation, Monitoring, and Analysis (with and without LLMs)

Instructor: Dr. Maryam R. Aliabadi
Lab Duration: 2 hours
Date: Sep 26th, 2025

Objectives:

- Learn Bash scripting basics.

- Automate security policy enforcement on Linux.

- Compare manual scripting and LLM-assisted scripting.

Prerequisites:

- Basic Linux command-line knowledge

- Understanding of common security policies

- A text editor for writing Bash scripts (e.g., Nano, Vim, VS Code)

Directory Structure:

```
mkdir -p ~/bash_lab/{part1, part2, part3, part4, part5}
```

Part 1: Basic Bash Scripting & Security Monitoring

Goal: Practice fundamental Bash scripting and start basic monitoring tasks.

Task 1: Hello World & System Info

```
#!/bin/bash
# Print greeting and system info
echo "Hello, Security World!"
date
uptime
```

a) Save as hello.sh in ~/bash_lab/beginner/

b) Run: bash hello.sh

Task 2: List files and save to log

```
#!/bin/bash
# List /etc and /home directories and save output
ls /etc /home > /tmp/file_list.log
echo "File list saved to /tmp/file_list.log"
```

Task 3: Check/create directory

```
#!/bin/bash
# Check if directory exists; create if not
DIR="/tmp/testdir"
if [ ! -d "$DIR" ]; then
  mkdir $DIR
  echo "Directory $DIR created"
else
  echo "Directory $DIR already exists"
fi
```

Task 4: Monitor basic login activity

```
#!/bin/bash
# Display recent login attempts
echo "Recent login attempts:"
last -n 5
```

Deliverables: Scripts + /tmp/file_list.log output + login activity printout

---

Part 2: Automating Security Policies & Log Analysis

Task 1: Check and correct file permissions

```
#!/bin/bash
chmod 644 /etc/passwd
chmod 600 /etc/shadow
echo "File permissions corrected"
```

Task 2: Monitor failed SSH logins

```
#!/bin/bash
FAILURES=$(grep "Failed password" /var/log/auth.log | tail -n 10 | wc -l)
if [ $FAILURES -gt 3 ]; then
  echo "Alert: More than 3 failed login attempts"
fi
```

Task 3: Disable inactive users

```
#!/bin/bash
USER="testuser"
LAST_LOGIN=$(lastlog -u $USER | awk 'NR==2 {print $4}')
```

```
if [ "$LAST_LOGIN" == "**Never" ]; then
  sudo usermod -L $USER
  echo "User $USER disabled"
fi
```

Task 4: Analyze recent sudo activity

```
#!/bin/bash
# Display last 10 sudo commands
tail -n 10 /var/log/auth.log | grep sudo
```

Deliverables: Scripts + demo outputs + sudo activity monitoring output

---

Part 3: Complex Automation, Backup & Threat Analysis

Task 1: Security report & backup script

```
#!/bin/bash
BACKUP_DIR="/tmp/etc_backup"
mkdir -p $BACKUP_DIR
cp -r /etc/* $BACKUP_DIR
find /home -type f -perm -o+w > /tmp/world_writable.txt
echo "Backup and security scan complete" > /tmp/security_report.txt
```

Task 2: Scan logs for suspicious activity

```
#!/bin/bash
grep -i "error\|fail\|unauthorized" /var/log/auth.log > /tmp/suspicious_activ
ity.log
echo "Suspicious activities saved to /tmp/suspicious_activity.log"
```

Task 3: Schedule with cron

```
# Add cron job to run script daily at 2 AM
crontab -e
# 0 2 * * * /home/student/bash_lab/advanced/security_backup.sh
```

Task 4: Generate a summary report

```
#!/bin/bash
echo "Security Summary:" > /tmp/security_summary.txt
wc -l /tmp/world_writable.txt >> /tmp/security_summary.txt
wc -l /tmp/suspicious_activity.log >> /tmp/security_summary.txt
echo "Summary saved to /tmp/security_summary.txt"
```

Deliverables: Script + /tmp/security_report.txt + /tmp/security_summary.txt + cron job verification

---

Part 4: Automating Security Policy Enforcement

You are system admin and you are given the following security policy:

Security Policy

i) User Account

- Each user must have a **unique account**; no shared accounts.
- Accounts created **only by administrators** after approval.
- Home directories must have permissions set to **700**.
- Password requirements:
    - Minimum **8 characters**.
    - Must include **uppercase, lowercase, numbers, and symbols**.
    - Expire every **60 days**.
    - Cannot reuse the **last 3 passwords**.

ii) Group Account

- Users are members of **only required groups**.
- Only admins belong to **sudo/wheel groups**.
- **Direct root access** is not allowed; use sudo.
- Group memberships are reviewed **quarterly**.

iii) Password & Authentication Controls

- **PAM** must enforce authentication policies.
- **MFA is required** for admin users.
- Accounts locked after **5 failed login attempts**.
- Password aging enforced via chage.

iv) File & Directory Permissions

- /etc/passwd → 644
- /etc/shadow → 640 (root-only)
- Sensitive files must not be **world readable**.

- Default **umask = 024**.

v) Account Monitoring & Auditing

- Enable logging of login attempts (/var/log/secure).
- Use last, lastlog, and faillog to review activity.
- Use **auditd** to track account changes.
- Disable **inactive accounts immediately**.

vi) SELinux

- SELinux must be set to **Enforcing mode**.

Tasks:

1. Write a bash script that automates enforcing the security policy. You may want to revisit Lab2.

Deliverables: enforce_policy.sh script (well-commented) + Screenshot of script execution.

2. **Bonus**: Bash script can not only *check* for violations, but also **fix them, alert admins, and keep logs with timestamps**. For example, *Checks SELinux status with getenforce and prints a warning if not **enforcing***. Enhance your script to automatically remediate security violations, send alerts, and log all actions with timestamps.

Deliverables: Enhanced enforce_policy.sh script (well-commented) + remediation logs + alert reports, Screenshot of script execution.

Part 5: Bash with LLM Assistance

Objective: Compare your manually written Bash enforcement script with a script generated by an LLM to explore efficiency, readability, and coverage.

Tasks

1. **Generate a security enforcement script using an LLM**
    - Ask the LLM to create a Bash script that enforces the same security policy as in **Part 4**:
        - User account rules (home directories, password requirements)
        - Group account restrictions (sudo/wheel membership)

        - Password & authentication controls
        - File & directory permissions
        - Account monitoring & auditing
        - SELinux enforcement
2. **Compare the scripts of Part 4 and 5**
    - Evaluate **efficiency** (e.g., fewer lines, better loops, reusability).
    - Evaluate **readability** (comments, structure, variable names).
    - Evaluate **coverage** (does the LLM script enforce all policy items from Part 4?).
    - Suggest improvements for both scripts.

Deliverables

1. **LLM-generated enforcement script** (well-commented)
2. **Comparison file**: llm_comparison.txt containing:
    - Summary of differences
    - Observations on efficiency, readability, and coverage
    - Suggested improvements

Lab Submission

1. Scripts in subfolders: part1, part2, part3, part4 and part5
2. Log files demonstrating execution
3. Short reflection comparing manual vs LLM-assisted scripting and analysis
4. Create a Google Drive folder named **Lab4-Solution** and upload your **bash_lab** directory to this folder. Collect all log files, written answers, and reflections into a single PDF document in the following format, and add it to the same folder. Finally, submit the zipped folder through the Learning Hub.

Filename: Lab3-FirstName-Lastname-StdNo.PDF

Good luck!