Lab8- DNS, Postfix & Simple Spoofing Detection

Instructor: Dr. Maryam R. Aliabadi
Lab Duration: 1.5 hours
Date: Oct 31st, 2025

Lab Overview

In this lab you will configure a simple local DNS (BIND9) and Postfix email server on Ubuntu, then practice identifying a spoofed email. The steps use easy commands and are suitable for introductory students.

Learning Outcomes

By completing this lab, students will be able to:

1. Understand the role of email protocols and headers in cybersecurity.
2. Configure basic email services (Postfix) on Ubuntu and test mail delivery.
3. Examine and interpret key email headers: From:, Received:, and Authentication-Results:.
4. Identify signs of email spoofing using SPF, DKIM, DMARC, and sender/received IP analysis.
5. Apply simple automation to detect suspicious or potentially spoofed emails using rules derived from manual analysis.
6. Develop basic text-processing or scripting skills in Bash or Python to parse and analyze email headers.

What You Need

- Ubuntu VM with a student account (use sudo for admin tasks)

- Internet access for package install

- Packages used: bind9, bind9utils, postfix, mailutils, telnet

---

Part 1 — Configure a Local DNS Server (BIND9)

A local DNS cache helps your system resolve names faster. Follow these steps:

Open Terminal and **Install DNS packages:**

```
sudo apt update
sudo apt install -y bind9 bind9utils
```

Edit config file:

```
sudo nano /etc/bind/named.conf.options

Inside the options { } block set:

options {
    directory "/var/cache/bind";
    listen-on { any; };
    allow-query { any; };
    recursion yes;
};
```

Check syntax: `sudo named-checkconf`

Restart service: `sudo systemctl restart bind9`
`sudo systemctl enable bind9`

Allow DNS in firewall: `sudo ufw allow 53/udp`
`sudo ufw reload`

Test DNS: `dig @localhost example.com +trace`

Part 2 — Configure a Local Email Server (Postfix)

Create a simple local mail system to send and receive messages on your Ubuntu VM.

Install Postfix and mail tools: `sudo apt install -y postfix mailutils`
`(When asked: choose 'Local only'; set system mail name to 'testserv')`

Edit Postfix settings: `sudo postconf -e "inet_interfaces = all"`
`sudo postconf -e "myhostname = testserv"`

Create mail alias: `sudo nano /etc/aliases`
`Add: root: student`
`Then: sudo newaliases`

Start and enable Postfix: `sudo systemctl start postfix`
`sudo systemctl enable postfix`

Send a test email: `echo "This is a test message" | mail -s "Postfix Test" student@localhost`

Check mailbox: `mail  # press Enter to view messages, 'q' to quit`

Part 3 — Detecting a Spoofed Email

You will send one normal email and one spoofed email, then check headers and logs to spot differences. Follow each step exactly.

Step 1 — Send a normal message:

```
echo "This is a legitimate message" | mail -s "Normal Email" student@localhost
```

Check mail logs:

```
sudo tail -n 20 /var/log/mail.log
```

Look for lines showing postfix/smtpd and from=<student@localhost> — this is the normal message.

Step 2 — Send a spoofed message (using telnet): `telnet localhost 25`

```
Then type the following (press Enter after each line):

EHLO spoof
MAIL FROM:<fake@unknown.com>
RCPT TO:<student@localhost>
DATA
From: "Admin" <admin@trusted.com>
To: student@localhost
Subject: Spoof Test

This is a fake (spoofed) message.
.
QUIT
```

Step 3 — View message headers: `sudo head -n 40 /var/mail/student`

Check the 'From:' header (claims admin@trusted.com) and compare to 'Received:' lines (shows localhost). Mismatch indicates spoofing.

Step 4 — Check for authentication in logs: `sudo grep "connect from" /var/log/mail.log | tail -n 10`
`sudo grep sasl /var/log/mail.log`

If 'sasl' entries exist, the message was authenticated. Spoofed messages typically lack 'sasl' entries.

Part 4 — Manual and Automated Email Analysis

You are given **10 real-looking email header snippets** provided in headers.txt. You need to identify how many of the emails are legit and how many of them are spoofed.

Task 1 – Manual Analysis

For each header:

1. Identify **sender IP** from Received fields
2. Check SPF, DKIM, DMARC results
3. Compare "From" domain with the actual sending server's domain
4. Decide if spoofing occurred and document why

Task 2 – Automate Analysis

build a small program (Bash or Python) that reads headers.txt and flags messages as SUSPICIOUS or OK using the same simple rules you used in manual analysis.

Simple rules (use these — they match the manual exercise)

- If **SPF** ≠ pass → suspicious.
- If **DKIM** ≠ pass → suspicious.
- If **DMARC** ≠ pass → suspicious.
- If the From: domain does not match the envelope sender (smtp.mailfrom) → suspicious (when smtp.mailfrom is present).
- If the earliest (bottom-most) Received: line shows a **local/private IP** (127.0.0.1, 10.x.x.x, 192.168.x.x, 172.16-31.x.x) while the From: domain appears external → suspicious.
- Otherwise mark OK.

Note: these are simple, classroom rules for spotting spoofed emails. Real-world detection uses more advanced checks and deeper analysis.

Deliverable

1- **Lab Report (PDF or Word)** containing:

- Screenshots or terminal outputs showing the Postfix server configuration and test emails.
- A table demonstrating the result of Manual analysis of each email snippet (10 messages), including:
    - Sender IP from Received: headers
    - SPF/DKIM/DMARC results
    - Comparison of From: domain and actual sending server domain
    - Determination of whether spoofing occurred and rationale
    - Legit or Forged

2-Automation Script

- A student-written Bash or Python script that reads the headers.txt file, applies the simple spoof detection rules, and outputs a summary for each message.
- The result of running the script clearly flagging suspicious messages.

Submit your report in a zipped folder with the following name format through the Learning Hub.

Filename: Lab8-FirstName-Lastname-StdNo.PDF

Good luck!

Quick Commands Reference

```
sudo tail -n 200 /var/log/mail.log

sudo grep 'connect from' /var/log/mail.log -n

sudo head -n 200 /var/mail/student

sudo postconf -n | egrep 'inet_interfaces|myhostname'

dig @localhost example.com +trace
```