Lab 4: DNS Attacks: Local Poisoning, Spoofed Responses, and NS Manipulation

EE P 595 — Computer Systems Security (Autumn 2025)
Instructor: Dinuka Sahabandu

Due: see Canvas

Contents

---

1 Introduction

This lab can be attempted individually or as a team (**stick to your project teams**). Using the experience from **Lab 2** (SEED VM & Docker LAN) and **Lab 3** (traffic sniffing/spoofing practice), attempt and complete **Tasks 1–4**. You will briefly mention the tasks in your Overleaf project and complete them on your SEED VM using Docker.

No pre-loaded code is provided. Your task is to (i) study the sample code snippets in the **Lab 4 PDF (SEED DNS Local lab PDF)**, and (ii) *adapt* those snippets to our lab setup and your coding style to complete each task. **Use of GenAI tools is allowed** *with acknowledgement in your report* (include a short note on what you asked and how you used).

Learning outcomes

After completing this lab, you will be able to:

- Explain the DNS resolution path and where spoofing/poisoning can occur.
- **Directly spoof** DNS responses to a user and compare with authentic replies.
- **Poison a local DNS cache** to persist malicious records across queries.
- **Manipulate NS records** to redirect an entire domain and reason about cross-domain limits.

2 Part 0 — VM Setup (Carryover from Lab 2)

What this part does. Confirms your SEED VM has Python, Scapy, Docker/Compose, and basic net tools. This is one-time per machine (reuse from Lab 2). If you need a refresher, see Lab 2 instructions.

3 Part 1 — Get Lab Files and Start the DNS Docker LAN

Download the DNS lab files from:
https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Local/

Download, unzip, and compose up (Compose v1)

Note. Many SEED VMs ship with `docker-compose` (v1).

```
# Create a working folder INSIDE the VM
mkdir -p ~/seed-labs/dns-local && cd ~/seed-labs/dns-local

# Download the official bundle for DNS Local
wget -O Labsetup.zip \
  https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Local/Labsetup.zip

# Unzip and enter the setup folder
unzip -q Labsetup.zip
cd Labsetup

# Clean up orphans if you've run other labs
docker-compose down --remove-orphans

# Build and start (Compose v1)
docker-compose build
docker-compose up -d
```

Sanity checks

```
# Containers should be running
docker ps --format 'table {{.Names}}\t{{.Status}}'

# List the DNS-related hosts (names vary by template, e.g., user/localdns/attacker)
docker ps --format '{{.Names}}'

# From the user container, try a simple dig through the local DNS (10.9.0.53)
docker exec -it hostA-10.9.0.5 bash -lc 'dig ns.attacker32.com'
```

Ready when: (i) containers are up, and (ii) you can query `ns.attacker32.com` from the user container and receive a response via the local DNS server.

If you hit a "ContainerConfig" error

```
docker-compose down --remove-orphans
docker-compose build
docker-compose up -d
```

Optionally prune stale networks/volumes and retry:

```
docker network prune -f && docker volume prune -f
```

4 Part 2 — Where to Run Code (Attacker Container)

Why this matters. You will sniff and spoof DNS packets. The `attacker` container is configured to see traffic; write your code under the shared `/volumes` directory so it appears inside containers.

Open a working shell and code folder

```
# List containers and pick the attacker container name
docker ps --format '{{.Names}}'

# Open a shell inside attacker (replace with your attacker name if different)
docker exec -it attacker bash

# Create/use a shared code folder
mkdir -p /volumes/code && cd /volumes/code
```

Interface note. In Scapy sniffing code, set the `iface='...'` to the correct bridge or host interface visible to your attacker (print interfaces with `ip -brief addr` and confirm by sniffing a few packets).

5 Tasks: DNS Attacks (Local)

5.1 Task 1 — Directly Spoof DNS Response to the User

Concept in 30s. Sniff a DNS query for a target name (e.g., `www.example.com`) from the user, then rapidly send a forged DNS reply to the *user* so it arrives before the legitimate answer.

Hints & guardrails

- Start from the sample sniff/spoof code in the PDF; adapt the `rrname`, `rdata`, interface, and transaction ID.
- From the user container, trigger queries with `dig www.example.com`.
- If the authentic Internet reply is too fast, consider the router delay tip in the PDF to slow outbound traffic.

Concept Check — Q1 (User spoof vs. real). *Provide:* (i) a screenshot of your spoof code console logging the intercepted query, (ii) the `dig` output showing the spoofed answer, and (iii) 2–4 sentences explaining how your forged reply beat the legitimate one and how you verified it (e.g., TTL, source, or authority section).

5.2 Task 2 — DNS Cache Poisoning: Spoofing Answers to the Local DNS

Concept in 30s. Target the *local DNS server* so your forged A record is stored in its cache; subsequent user queries are then answered from the poisoned cache.

Flow

1. **Flush** cache on the local DNS: `rndc flush`.
2. Trigger a user query (`dig www.example.com`) and **spoof** the reply to the *local DNS* (correct 4-tuple, transaction ID, and port).
3. **Dump** cache to verify: `rndc dumpdb -cache` then `cat /var/cache/bind/dump.db`.

Concept Check — Q2 (Cache evidence). *Provide:* (i) a snippet of the cache dump showing the forged record, and (ii) 2–4 sentences explaining TTL behavior and why subsequent queries are answered instantly from cache.

5.3 Task 3 — Spoofing NS Records (Redirect an Entire Domain)

Concept in 30s. In addition to an Answer section, include an **Authority (NS) record** that delegates the target domain (e.g., `example.com`) to `ns.attacker32.com`. Once cached, queries for any name under the domain are steered to the attacker-controlled nameserver.

Flow

- Build a reply with: **Answer** (A for the requested name) *and* **Authority** (NS for the domain to `ns.attacker32.com` at `10.9.0.153`).
- Verify via `dig` that multiple hostnames in the domain (`www`, `mail`, etc.) now resolve through the attacker NS.
- Inspect cache to confirm the NS entry is present.

Concept Check — Q3 (Domain-wide effect). *Provide:* (i) two `dig` outputs for different hostnames under the domain showing redirection via the attacker NS, and (ii) 2–4 sentences explaining why an NS record impacts the *entire* domain vs. a single hostname.

5.4 Task 4 — Spoofing NS for Another Domain (Limits & Behavior)

Concept in 30s. While attacking a query for `example.com`, also include an Authority NS record attempting to delegate `google.com` to `ns.attacker32.com`. Observe what the local DNS *actually* caches and why.

Flow

1. Flush cache, craft a reply to the `example.com` query with **two** NS entries: one for `example.com`, one for `google.com`.
2. After the attack, dump cache and check which NS records were accepted.
3. Explain acceptance/rejection logic (cache policies, bailiwick rules, relevance of Authority to the original question).

Concept Check — Q4 (Acceptance rules). *Provide:* (i) cache dump snippet(s) highlighting what *was* cached vs. *not* cached, and (ii) 3–5 sentences discussing bailiwick/relevance and why cross-domain authority is (typically) ignored.

6 Deliverables

- PDF (max **3–4 pages**): Q1–Q4 concept checks (each: one key screenshot + a concise explanation).
- Appendix: brief acknowledgement of any GenAI assistance (tool + prompt summary + how you used the output).
- Zip archive `<UWNetID>_lab4_dns.zip`: include all code you wrote (`.py`, `.c`, or other), plus any small helper scripts/configs. If applicable, add a short `README` with run instructions.

7 Troubleshooting (quick)

- **Nothing cached:** ensure you spoofed the reply to the *local DNS* with the correct transaction ID and source port (fixed per lab config), and *before* the real reply.
- **Slow spoof arrives late:** apply the router `tc` delay tip from the PDF to slow outbound traffic; re-run.
- **Wrong interface:** verify the sniffing `iface` is the one carrying DNS queries (print packet summaries to confirm flow).
- **NS not honored:** revisit how Authority ties to the queried name and bailiwick rules; check you did not only spoof Additional without Authority.

Acknowledgments

The content in these labs is based on the labs provided with our course textbook, *Computer & Internet Security: A Hands-on Approach* (3rd ed.) by Wenliang Du, and on SEED Labs (https://seedsecuritylabs.org). We thank the authors for making these materials publicly available.