

DNS Security Lab

Kaminsky Cache Poisoning Attack



Client: 10.19.0.2

Attacker2: 10.19.0.10

Resolver: 10.19.0.53

Authoritative: 10.19.0.153

Network Security – Introduction to Cybersecurity

Luca Campa

 **Academic use only.** This setup is intentionally weak and must run only inside the internal lab Docker network.

Contents

Submission Rules & Setup	2
1 Overview	3
2 Environment	3
3 Start and Health Check	3
4 Task 1: Run Kaminsky Attack	3
4.1 Verify Poisoning	4
5 Troubleshooting	4

Rules & Setup

Rules

- Do not attack external DNS infrastructure or public resolvers.
- After running `docker compose up`, turn off any external network connections to ensure the lab environment is isolated. The provided docker network is already isolated, but better safe than sorry.
- You can use any online resources, but you must write the code yourself. I'm aware of the usage of AI tools, but I expect you to understand, describe and justify your solution.
- You can ask for help from the instructor or teaching assistants during office hours.
- Please submit your code and a brief report (no page limit) describing your implementation and results. Upload a zip file containing the report (a PDF file) and your scripts. Include a short guide on how to execute your scripts if instructions are different from the one presented in this document. The report should include:
 - A description of your implementation approach for each exercise.
 - An analysis of the results, including any challenges you faced and how you overcame them.

The report should NOT include:

- Your full scripts.
- If you want to include code snippets, make sure they are relevant to the discussion.

You can refer to the code by using the line numbers in the scripts.

The scripts must be well-commented. Each function should have a clear description of type of inputs, outputs and what it does. The code should be organized and readable.

- You will be asked to present one of your solutions (not all the exercises) during the Friday Lab Sessions. Make sure to be prepared to explain your code and the results you obtained.
- Keep the final state reproducible so the lab can be re-run from a clean container start.

Setup

What to do first

1. Create or start a VM for the lab.
2. Install the required packages inside the VM.
3. Start the Docker lab environment and verify the DNS containers are reachable.

```
sudo apt-get update
sudo apt-get install -y python3 python3-pip docker.io docker-
  compose-plugin dnsutils tcpdump tshark net-tools iputils-ping
  wireshark

pip3 install scapy dnspython cryptography

sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER
```

After logging back in, open the lab folder and start the exercises from the commands below.

1. Overview

This variant isolates Kaminsky-style cache poisoning in a dedicated environment.

Instead of BIND as resolver, this lab uses a custom Python resolver that is intentionally weak:

1. Fixed upstream source port: `33333`
2. Reduced transaction-ID range: `10000-10005`
3. Internal-only Docker network with no external connectivity

Learning Objectives

- Understand Kaminsky attack mechanics in a controlled setup
- Observe why TXID (Transaction ID) and source-port randomness matter for DNS security
- Verify poisoning impact from the client perspective

2. Environment

Container	Role	IP	Notes
wk_client	DNS client	10.19.0.2	Issues lookups
wk_attacker2	Attack node	10.19.0.10	Runs poisoning script
wk_resolver	Custom Python resolver	10.19.0.53	Weak randomness
authoritative	example.com server	10.19.0.153	Authoritative answers

3. Start and Health Check

```
cd dns-lab/with_kaminsky
docker compose up -d --build
docker compose ps
```

Baseline query:

```
docker exec wk_client dig @10.19.0.53 www.example.com +short
```

Expected baseline answer: `1.2.3.4`

4. Task 1: Run Kaminsky Attack

>_ Task 1

Goal: poison resolver cache so `www.example.com` resolves to `6.6.6.6`.

In `solutions/kaminsky_poisoning.py` you will find a skeleton of the attack script. You need to fill in the missing parts to perform the poisoning attack successfully. Then:

```
docker cp ./solutions/kaminsky_poisoning.py wk_attacker2:/solutions/  
docker exec -it wk_attacker2 /bin/bash  
cd /solutions  
python3 kaminsky_poisoning.py
```

The script repeatedly:

1. Triggers random subdomain lookups under `example.com`
2. Floods forged responses across the reduced TXID range
3. Tries to insert malicious data for `www.example.com`

4.1 Verify Poisoning

> Task 2

From the client, verify whether resolver cache now returns the attacker-controlled IP.

```
docker exec wk_client dig @10.19.0.53 www.example.com +short
```

Expected poisoned answer: `6.6.6.6`

5. Troubleshooting

```
# Restart resolver between runs  
docker compose restart resolver  
  
# Rebuild resolver after code changes  
docker compose up -d --build resolver  
  
# Check resolver logs  
docker logs wk_resolver --tail 100  
  
# Re-run attack  
docker exec wk_attacker2 python3 -u /solutions/kaminsky_poisoning.py
```

This environment is intentionally insecure and only for controlled lab usage. The Docker network is internal-only and isolated from the outside.