Homework 3
COSI 141A-1: System and Network Security

Due: November 23, 2025 at 11:59 PM

In class we discussed DNS poisoning from different perspectives. One such vantage point is *on-path*, where an attacker can passively monitor traffic that includes DNS queries and inject properly forged responses. Such attacks can be carried out by an adversary observing traffic on open WiFi networks or by malicious ISPs. Depending on the origin of the packets (a DNS resolver or a client), an attacker can poison the resolver cache or send the client an *incorrect* DNS resolution.

Writing a DNS Injector (100 Points)

In this task, your goal is to write an on-path DNS injector that will capture traffic from a network interface and attempt to inject forged DNS responses (crafted by you) to selected DNS A requests. You should write your program in a single file and name it *dnsinject.py*. Your DNS injector should execute using the following command and take in the following inline arguments:

```
>> python dnsinject.py [-i interface] [-h hostnames]
```

- `-i interface`: Listen on the victim's network device interface (e.g., eth0, virtual0) and inject forged packets. This argument is mandatory and must be provided. If the interface is missing or invalid, your program should print an appropriate error message and exit
- `-h hostnames.csv`: Path to CSV file with hostname-to-IP mappings (e.g., `www.example.com` → `1.2.3.4`) that your spoofer targets. If not provided, forge all responses with `127.0.0.1`.

To simplify testing, we provide a `Dockerfile` that builds a containerized environment for the victim client machine. Use the provided Docker configuration and follow the instructions to build and run the container. The network is configured so that the container functions as the victim, while your host machine acts as the on-path observer/attacker that injects forged DNS responses.

Instructions:

- To complete the task, you will need to understand the format of a DNS query and response (DNS Format).
- Your injector should only handle IPv4 DNS A requests.
- To make your code efficient when observing DNS packets, apply packet filters so you only capture relevant traffic.
- Pay attention to the header values in the request and the response. Some will remain the same, while others will be flipped. You will also have to add the IP in the response from the hostname file (if provided).
- DNS uses the TXID and source-port randomization to make guessing harder. With the vantage point of an on-path attacker, you can see both of these.

- You may test your injector using your home network and the default resolver. However, **your final submission must include results obtained against the university's DNS resolver**(for example, by running your code on campus or via the university VPN).
- The Wireshark or tcpdump utilities must be used to capture packets and debug your code. This will be helpful when you are crafting your packet and comparing it with a legitimate DNS response.

Testing your DNS injector across different resolvers

In this subtask, you will evaluate the performance of your DNS injector against multiple public DNS resolvers. For a comprehensive list of public DNS resolvers from different regions, see Public DNS Server List and DNS Checker's Public DNS Servers. Use the `dig` utility to query a specific resolver, for example: `dig @8.8.8.8 example.com`. You should test your DNS injector against at least five different public DNS resolvers from the lists, and you should strategically select resolvers that are geographically and administratively diverse to observe how these factors influence your injector's performance. For each resolver, conduct at least 10 injection attempts and report the tool's success rate.
The success rate is defined as:

$$\text{success rate} = \frac{\text{number of spoofed responses accepted by the client}}{\text{number of forged responses sent}} \times 100\%.$$

For example, if you ran 10 trials against resolver "8.8.8.8 (Google)" and the client accepted 2 forged responses, the success rate is 20%.

What to submit

For this task, you are required to upload the following files in an archive named `hw3.zip`.

1. `dnsinject.py`: Your main program that performs DNS injection. Ensure your code is well-documented.
2. `injection.pcap`: A pcap file containing the packets that demonstrate a successful injection attack you conducted, for all domains provided in the hostnames file. Filter out unrelated traffic. For this trace, use the default resolver from the main setup. If a successful injection cannot be achieved, you may instead use one of the alternative resolvers from the resolver-testing subtask. In all cases, include a detailed description in your report, explaining any factors that prevented success with the default configuration.
3. `analysis.pdf`: Provide *in-detail* explanations of your approach to tool design, injection, and how you successfully tested it. List the links to all online resources you referred to and any challenges you faced during the implementation. Additionally, you should report your measurements and analysis of the testing of your DNS injector across different resolvers as described above. Include a table summarizing the success rates for each resolver, along with any relevant observations and insights you gathered during your experiments (e.g., resolver X has higher success rate due to....).

Hints:

- Do not begin the assignment without first understanding in detail the DNS protocol and packet structure. Use the course's lectures and the resources provided in the handout. As a first step, capture DNS packets using Wireshark or tcpdump and analyze them.
- Beating the race condition between the legitimate DNS response and your forged response is critical. However, it is also important to ensure that your forged response is well-formed. Pay attention to all fields in the DNS response packet.
- In `resources.zip` you will find (i)a hostname-to-IP mapping file for your target dnsinjector, and (ii)a `Dockerfile` to build a containerized environment for the victim client along with instructions.
- More about Scapy and its capabilities: Geeksforgeeks and here.

Good luck!