Assignment 4 – CSCI 397

Instructions and Overview

- **Due Date:** Wednesday, Dec 15, 11:59 pm EST
- This is an individual assignment**.** You are allowed to discuss your approach with a partner; however, you must submit your own work on Canvas. You may discuss general ideas and approaches but **NOT** exact solutions. If you work with someone, make sure to mention their name along with your submission.
- If you plan to use any late days for the assignment, make sure to submit the details on the Late Days Google Form provided on Canvas.

Task 1: Writing a DNS Injector **[100 Points]**

For this assignment, you are required to write your solution in Python programming language, version 3.5 or later. The goal of this assignment is to write an on-path DNS packet injector To complete this assignment, you are allowed to use standard Python libraries as well as packet generating modules such as Scapy, dpkt etc.

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

In this assignment your goal is to write an on-path DNS injector which will capture traffic from a network interface in promiscuous mode, and attempt to inject forged responses to selected DNS A requests. You should write your program in a single file and name it as **`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`**: Listen on network device interface (e.g., eth0). If not specified, your program should select a default interface to listen on. The same interface should be used for injecting forged packets.

- **`-h`**: Read a hostname file containing a list of IP address and hostname pairs specifying the hostnames to be hijacked.

Hints:

- To complete this assignment, you will need to understand the format of a DNS query and response. Here is a resource that you might find useful.
- As the homework requires you to observe DNS packets, to make your code efficient you can apply a filters for the packets you intercept. This ensures you are looking at only the relevant packets.
- The Wireshark utility can be used to capture packets and investigate the values. This will be helpful when you are crafting your packet and comparing it with a legitimate DNS response.
- Pay attention to the header values in the request and the response. Some will remain the same while the 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 hard. With the vantage point of an on-path attacker you are able to see both these.

There are multiple ways to approach this problem. Be creative in your approach! If you have any questions, make sure to.

Testing Your Code

One way to test your tools is to have a guest VM running on your computer that makes DNS queries. Your DNS injection program running on the host OS and is able to view those queries, and inject forged responses. To make requests within the VM you may use the dig utility.

Your program will be tested for both correctness as well as efficiency. By correctness, we mean that your forged responses are properly formed and accepted by domain lookup tools such as dig and nslookup. While testing your code, you can force requests to non-existent DNS resolvers. In such a case a legitimate response will never arrive and if your forged response is accepted by the DNS lookup tool, you know you have crafted a successful packet.

To ensure for efficiency, make sure your program is fast enough and inject packets to the correct destination before the legitimate server response arrives. Remember, to successfully forge a response, you have to win the race condition. We will be testing your code against the default UIC resolver, as well as common open DNS resolvers.

Alternative Testing Setup: If you are unable to setup a VM, you can use the dig utility within your host system to generate DNS requests. You can then run your program to observe packets on the same interface you making queries on, and inject forged responses.

Submission Instructions

For this assignment you are required to upload the following files in an archive named as **`hw4.zip`**.

1. **`dnsinject.py`**: Your main program that performs DNS injection

2. **`injection.pcap`**: A pcap file containing the relevant packets of a successful injection attack that you have conducted. Make sure to filter out any traffic which is unrelated to your attack.

3. **`explanation.txt`**: Describe in a paragraph about your approach on injection, how you successfully tested it, and list the links of online resources you referred to.