Homework 5: DNS Spoofing Attacks

Overview:

DNS Spoofing [1] is the process of making a DNS entry to point to another IP than it is supposed to point to. In this homework, you are going to implement the DNS ID spoofing technique as described in the DNS Spoofing [1] paper.

Here are some useful tools that you would familiarize yourself with in this homework:

- **Dnsmasq** which is a lightweight caching DNS server. You will configure and run DNS servers on hosts in Mininet.
- **Scapy** which is a packet manipulation tool written in python. You will use this tool to sniff and manipulate traffic.

We assume you know how to create your own Mininet topology in python from your previous assignments, we won't be providing any skeleton Mininet code this time. Further, you might want to go through the links provided in the *Useful Links* section of this handout to get well versed with the tools you will be using.

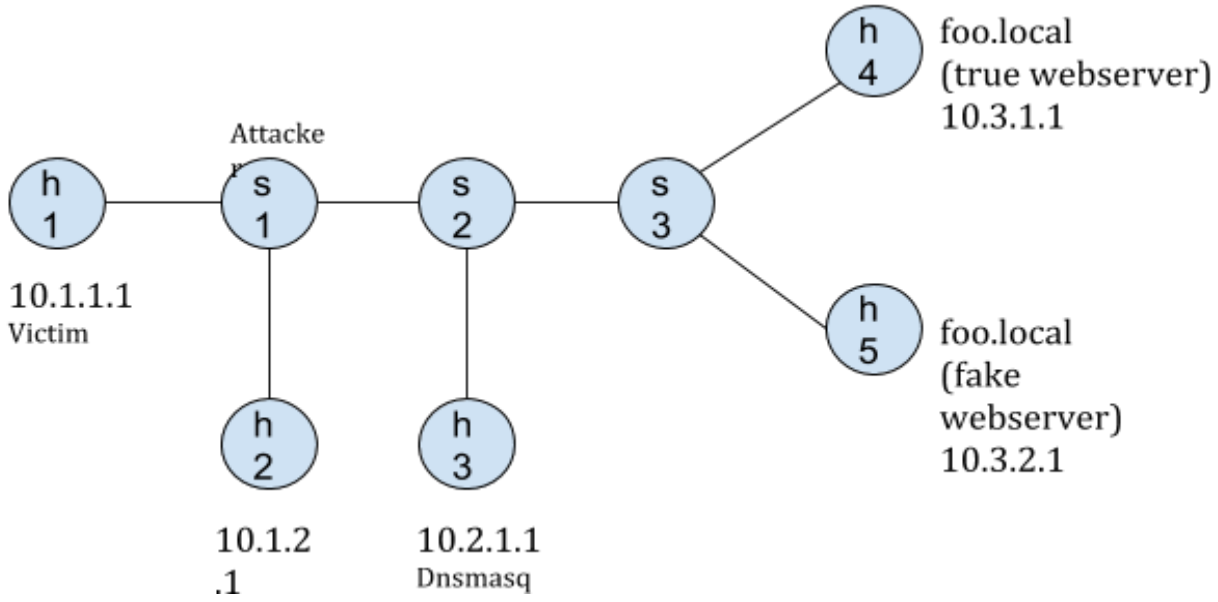
Tasks:

Task 0: Setup VM and Install dependencies

1. Use floodlight VM
2. Install Dnsmasq: sudo apt-get install dnsmasq
3. Install Scapy: sudo pip install scapy

Task: DNS ID Spoofing

Step 1: Create Topology



- Add a **proactive controller (c0)** similar to how you did in assignment 2.
- s1, s2, s3 are switches
- h1 - h5 are hosts

Step 2: Launch attack

Understanding the attack: Hosts h4 and h5 are hosting simple HTTP servers. Host h3 runs dnsmasq and points 'foo.local' to IP 10.3.1.1. In this case, we assume that the attacker has control over switch s1. The attack code runs on s1. Use scapy to write your attack code.

Attack.py: The code should basically sniff the Victim's traffic. For every DNS request, spoof the DNS ID and craft a DNS response packet with this ID in a way that the resolved IP points to the fake web server (10.3.2.1).

You might find the sniff and send functions in Scapy particularly useful.

NOTE: You should add the host '10.2.1.1' to /etc/resolv.conf.

Here is how you will cause the DNS ID spoofing attack that you read in the paper:

1. Start a simple HTTP server on h5
2. Start a simple HTTP server on h4
3. Run dnsmasq on h3 with the required options. Basically, point foo.local to

10.3.1.1.

4. Before the attack, request to foo.local should resolve to the correct IP.
5. Run the attack script on s1.
   In default switch configuration of mininet, kernel-mode switch does not know where to route packets, and we need this functionality for s1 to inject the DNS response packet to h1. Therefore, you need to explicitly add routing entry and an arp entry for the switch s1 to be able to inject traffic to h1. To do this, use the following commands:

   - <u>To add routing entry:</u>

     s1 route add -host <h1 ip-address> <interface of s1 connecting to h1>

   - <u>To add ARP entry:</u>

     s1 arp -s <IP of h1> <HW Address of h1>

Note that this is not specific to the attack but because of the mininet environment specifics, we need to do this. In a real attack, this step is would not be necessary.

6. Now, the request to foo.local (Use wget command) should point to the fake web server.

NOTE: Add a delay to the link between s2 and h3 to ensure that the response from the attack reaches before the response from the actual dns server. Ideally, the attack should be faster than the actual dns response, but in this case, the python code will take longer to execute so you will add the delay to make the link slower in order to delay the response from the actual dns server.

Submission:

1. Submit the python files that contain the code that you used to build the topology for the task.
2. Submit the attack code in a file named 'attack.py'.
3. Submit a writeup (andrewID_hw4.pdf) that contains the following details:
   a. A step by step illustration of how each attack works with screenshots of the attacks. Also, show the dns logs on the dns servers. Explain how each command works and why you used specific options. We don't expect you to write an essay. Screenshots with 1-2 lines of relevant explanation is enough. We just want enough information to validate your understanding.
   b. A **<u>short</u>** explanation on how your scapy attack code works

4. Name your bitbucket repo in the format '**andrewID-hw5**'. Your repo should contain the

following files:

-- andrewID-hw5
    |--- hw5-topo.py
    |--- attack.py
    |--- andrewID_hw5.pdf

5. share on Canvas your bitbucket repository info.
6. Provide access to the instructor and the course TA's (so that we can clone your repository using git clone. ***Warning: If you do not follow instructions and we cannot access your repository, you may not get a credit***.

Useful Links:

1. http://users.ece.cmu.edu/~vsekar/Teaching/Spring19/18731/reading/DNS_Spacefox.pdf
2. Mininet Basics: http://mininet.org/walkthrough/
3. Mininet Basics: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet
4. Dnsmasq Reference: http://linux.die.net/man/8/dnsmasq
5. Scapy Reference: https://scapy.readthedocs.io/en/latest/usage.html
6. Scapy Reference: https://scapy.readthedocs.io/en/latest/
7. For Note 1: http://mininet.org/api/classmininet_1_1node_1_1Node.html