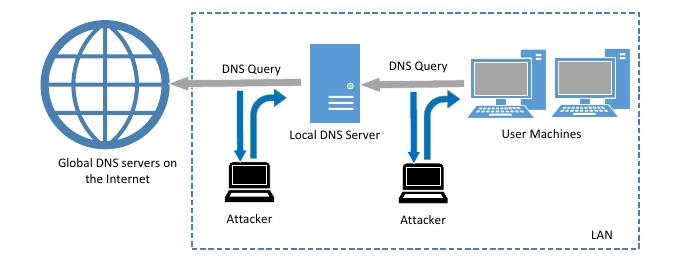
EECE 655 Assignment 1: Developing DNS spoofing attacking and detection tools

# Introduction

During the past two weeks, we have developed several tools to conduct Domain Name System (DNS) spoofing attack and detect them successfully. In order to do that, we have conducted an in-depth research on the attack to see how it works and how it can be detected, and we have referred to some sample codes from previous work to better understand the attack. After that, we have developed two tools for conducting DNS spoofing attack: one that sends a fake DNS query response to the victim, directing him to a malicious website, and one that sends forged DNS queries to the DNS server. To detect the two attacking methods, we have developed two detection tools: one to monitor DNS query responses to detect the first attacking method, and one to track DNS queries to detect the second attacking method. All tools were developed using the Scapy library and Python programming language.

# Setup

To conduct necessary tests on the attack and detection scripts a suitable environment was built on an Ubuntu virtual machine run over VirtualBox. Inside this VM a docker network was spawned having all the major members of the attack scenario connected to a docker router. A user which was the target of the attack of IP 10.9.0.5, a local DNS server with BIND9 DNS service running and a cache setup with DNSSEC off IP 10.9.0.53, an attacker of IP 10.9.0.153. Each  member is a separate docker container where commands can be run separately on, this way attack scripts will be run on the attacker side, and detection scripts run on the user side.

# Attacking Tool 1

In version 1 of the attack tool, the attacker exploits the multiple vulnerabilities that exist in the DNS exchanging techniques, and how the user has no sort of regard to how the reply arrives to him, as well as the DNS server in other issues like authentication. These vulnerabilities include but are not bound to: no authentication from the user, nor the DNS server about the DNS queries and their answers, so any attacker can use that and masquerade as the DNS server. The vulnerability of communicating un-encrypted packets and their headers and payloads which allows the attacker to sniff the sort of packet they need and change its content and configuration. This attack consists of both the victim and attacker being on the same local network. The attacker here sniffs the LAN for packets coming from their target specifically, and having a DNS segment/packet, which would be a request to the DNS server for URL checking. As soon as the attacker sees such a DNS packet as a query, they immediately send a reply as DNS answer to the victim telling them of a falsified IP of the questioned URL. This malicious IP will reroute the victim  to the attacker’s website to steal info from the victim. The tool checks if the IP of the DNS is from the victim, and checks that they are asking for this website (here www.example.com). Then the attacker constructs a falsified DNS packet with the wrong IP as an answered resource and sends it back to the victim, hoping that they get it before the DNS server sends their real answer. Then the victim is redirected to the malicious IP and ignores the packet sent by the DNS server after.

# Detection Tool 1

The aim for this tool is to detect the DNS spoofing attack developed in “Attacking Tool 1”. This detection tool sniffs the network traffic for a user’s interface, displays all DNS packets for this interface and detects a DNS spoofing attack if existed. Detection is done by tracking the number of DNS query responses for a single DNS query packet. Knowing that every DNS query should only have one response, the tool sends the user an alert if it detects two or more DNS responses for the same query. If two responses were sent for a single query, then we have multiple IP resolves for the same website, which is not possible. This guarantees that there is a DNS spoofing attack on this interface. Fortunately, this tool warns the user about that if existed.

# Attacking Tool 2

In version 2 of the attack tool, the goal is to avoid version 1 of the detection algorithm.  In order to achieve this avoidance, the main idea is to flood the victim with DNS queries that appear to be coming from the victim themselves, so that the detection algorithm doesn’t compare the spoofed DNS answer from the attacker with the real DNS answer from the server. We execute the two attacks in such a way that, when the spoofed DNS answer reaches the victim, we start flooding the victim with the aforementioned DNS queries. The detection algorithm is waiting for two different replies to the same DNS query, so that it compares them to make sure they’re the same and that spoofing isn’t taking place. However, once the fake answer is received, a new DNS query will now be detected due to attack version 2 taking place, and this query will have one answer, as another query will take its place. This carries on, until the actual DNS reply arrives in the form of an answer to a single DNS query, and so the detection algorithm is fooled since it’s only receiving one answer each time it checks a DNS query.

# Detection Tool 2

The attacker found a way to bypass our detection by sending a fake query in between both of the answers received (The original DNS answer, and the attacker’s spoof answer). So our code did not detect any suspicious attack because for each query there was one answer not two answers. We had to develop another way of detecting any suspicious traffic happening on the machine. We created an array that stores all of the queries and all of the answers. From the array, we checked if for one same query was multiple different answers, if it was the case, the detection tool would warn the user, and this is how we were able to detect the second version of the attack.

# Attacking Tool 3

The only thing that is different in this code, as compared to attack v.1 is that the IP that we are sniffing is not the victim, but the local DNS server. When the user requests a specific IP mapping of a URL, considering the DNS server does not already have the answer to this IP, this DNS server requests a DNS query from a root server on a different network, that's where the attacker sniffs this packet and send a wrong spoofed answer packet, and makes the DNS server save it in cache, and then it drops any answer that arrives after from the real root server. With that, the user would receive a normal DNS answer to the IP, and that would evade detection V.2 algorithm. The only change in this code is on line 112, where the sniffed IP is 10.9.0.53 (local DNS server) instead of 10.9.0.5 (the victim).

# Testing and Results

The testing of scripts and results will be shown in a separate proof of concept video.

# Contributions

## 1. Attackers:

Ali Hijjawi showed effort in working on the attacks and upgrading them to avoid detection. His part of the code was done and indicated in each of the attackv1 and attackv2, as well as the small contribution and motivation to go for attackv3. Not only did he write his own part of the code, but he also fixed all the issues that arose within the attacking code. Ali was responsible in firstly creating the DNS falsified segment of the attacking packet and in setting up the DNS flood on the victim afterwards, as well as pushing for the final bit of attack 3.

Issam Misto's contribution was essential to have everything working properly. Issam’s code contribution is clearly stated in the code in attackv1 and attackv2. He was a great support, and Ali and him worked well together to execute the attack. He was splitting the code and making the attackers work as fairly as possible. Issam was responsible for creating the IP and DNS segments as well as constructing the packet used for the flood.

## 2. Defenders:

Ryan Al Alam was highly motivated to detect the attacks sent by the other teams. He setup the user to filter the required packet and constructed detectv2 to enforce detection of multiple resolves using packet history as indicated in the code.

Abdel Rahman Al Ladiki had the role of constructing the initial detection detectv1 algorithm which checked responses per query and alerted the user in case of multiple resolves per query as indicated in code zones.

# References

* <https://null-byte.wonderhowto.com/how-to/build-dns-packet-sniffer-with-scapy-and-python-0163601/>
* <https://seedsecuritylabs.org/>
* <https://infosecwriteups.com/how-i-pranked-my-friend-using-dns-spoofing-6a65ff01da1>
* <https://github.com/pponakala/dnspoisoning>
* <https://github.com/faddy/dnsuck>
* <https://github.com/vinsdragonis/DNS-Spoofer/blob/main/dns_spoof.py>