EECE 655

Assignment 1 report

Samia Noaman

DNS spoofing attacking tool-DNS spoofing detection tool

**DNS spoofing attacking tool:**

Imports:

- Scapy library: Used for packets manipulations and network analyses in my case used to intercept DNS requests and send spoofed DNS responses.

* Requests library: Makes HTTP requests simpler to manipulate and fetch responses from webservers.

Definitions:

-Dns\_spoofing definition: In general, this function processes captured packets by the sniffing function. First, it checks if the packet contains DNS query record through the (has.Layer) function, if yes it jumps to the next line (packet [DNSQR].qname.decode()) which extracts the requested domain name from the DNS query and prints it. It may be useful to know what the user is trying to browse but in this case we will be spoofing every request.

<http://10.0.2.15:8080/spoofed_page.html> is the URL for the spoofed\_page that I created where the user will be redirected. Afterwards, it sends the HTTP get request to this URL to load the spoofed page through the (requests.get(url)) and then included in the DNS response through the( html\_content) variable.

The next stage will be the crafting of the DNS response packet, this stage is divided into layers. First, the Ip layer where the source IP will be set to the original destination IP and the destination IP to the original source IP. Second, the UDP layer which sets the source and destination Ips corresponding to the original packet. Finally, the DNS layer which crafts the DNS response using specific parameters which are the id: The transaction ID from the original request, so the response can be matched with the request, qr=1 indicating that it is a query(0) or response(1) packet, aa=1 indicating whether the responding server is authoritative(1) or not(0) for the domain name in question, qd=packet[DNS].qd containing what the user is asking for in the query, here it passes the original DNS query section from the intercepted packet, ensuring the response correctly reflects the question asked by the client, an=DNSRR(rrname=packet[DNSQR].qname containing the answer providing the spoofed IP and finally the ttl set to 10 and the rdata providing the spoofed IP.

Then, I appended the HTML page that I created tp the spoofed response through the txt\_record parameter.

Finally, I send the spoofed packet through the send () function.

Sniff function:

The most important function that will sniff the traffic ongoing in the network where my attacking machine is, focusing on port 53 where UDP packets go through especially the dns queries, with the parameter prn set to the spoofing definition and the interface parameter iface set to eth0 which is the network interface of my virtual machine.

**DNS spoofing detection tool:**

Imports:

Same as above (scapy)

Datetime: For the timestamp

I created a dictionary where I will store legitimate IP addresses resolved before with their corresponding domain names so I could compare later between new DNS responses and previously seen legitimate ones.

In the detection tool I chose to log with a time stamp to keep track with the spoofed responses.

log\_spoofing\_attempt definition:

It prints that a spoofing in attempted with the corresponding time and date, with the corresponding domain queired, original response(legitimate one) and the spoofed one found and the source IP.

detect\_dns\_spoof definition: takes a packet pkt as input and detects whether it has been spoofed or not. First, it ensures that it is a dns packet for extra check, then it checks that the DNS response contains at least one answer where (ancount) is the number of answer records. This ensures that we are only analyzing DNS response packets, not queries.( if pkt.haslayer(DNS) and pkt[DNS].ancount > 0:).If the first if statement is satisfied, it checks if the packet has a DNS query section, extracts the domain name (qname) from the DNS query layer and decodes it if necessary. After, I checks if the domain name has been queried before and if a legitimate IP address for it has been stored in the legitimate\_responses dictionary, and retrieves the legitimate IP address associated with the domain from the dictionary. Then, I extracts and decodes the IP address from the DNS Resource Record in the current DNS response packet, compares the original IP with the IP found in the current DNS response. If the two IP addresses do not match, this indicates a potential DNS spoofing attempt, and calls the log\_spoofing\_attempt function to log the details of the spoofing attempt. If the domain does not already have a legitimate IP address stored in the dictionary, it checks if the DNS Resource Record (DNSRR) is present in the packet, stores the IP address found in the current DNS response in the dictionary.

Sniff function same as the attacking tool.

**Note**: In the attacking tool I tried to install a DNS resolver on virtual machine so the spoofed page could show on the browser not only in the webserver in the terminal, but it didn’t work even after trying on several Linux virtual machine it even corrupted one of them.

**Instructions on how to run the programs + results:**

After importing the necessary libraries, open 3 terminals in a virtual machine where the network adapter is set to NAT or bridged adapter, and if tried between 2 machines they should be set on bridged adapter so they could be in 1 network subnet. In the first terminal, input the following command python3 -m http.server 8000 it could be port 80, 8080,etc… as long as the port isn’t running another task, this command will display how the spoofed response will GET request the spoofed page as response. In the 2nd terminal command sudo python3 detect3.py which will run the detection tool, in the 3rd terminal command sudo python3 dns3.py running the attacking tool. Both tools need higher privileges(sudo) because of the sniffing function mainly, and other functions. Then, browse any website on the browser or through the ping/dig commands then observe how the DNS packets will be intercepted and spoofed on the dns3 terminal, how the html spoofed\_page will be requested in order to load with the spoofed response on wen server terminal and how spoofed packets are being detected on the detect3 terminal, and check hoe the browser will try to load the original requested website but won’t and finally will display an “unable to connect warning”.

The spoofed page can be seen separately also on the browser through <http://10.0.2.15:8080/spoofed_page.html>

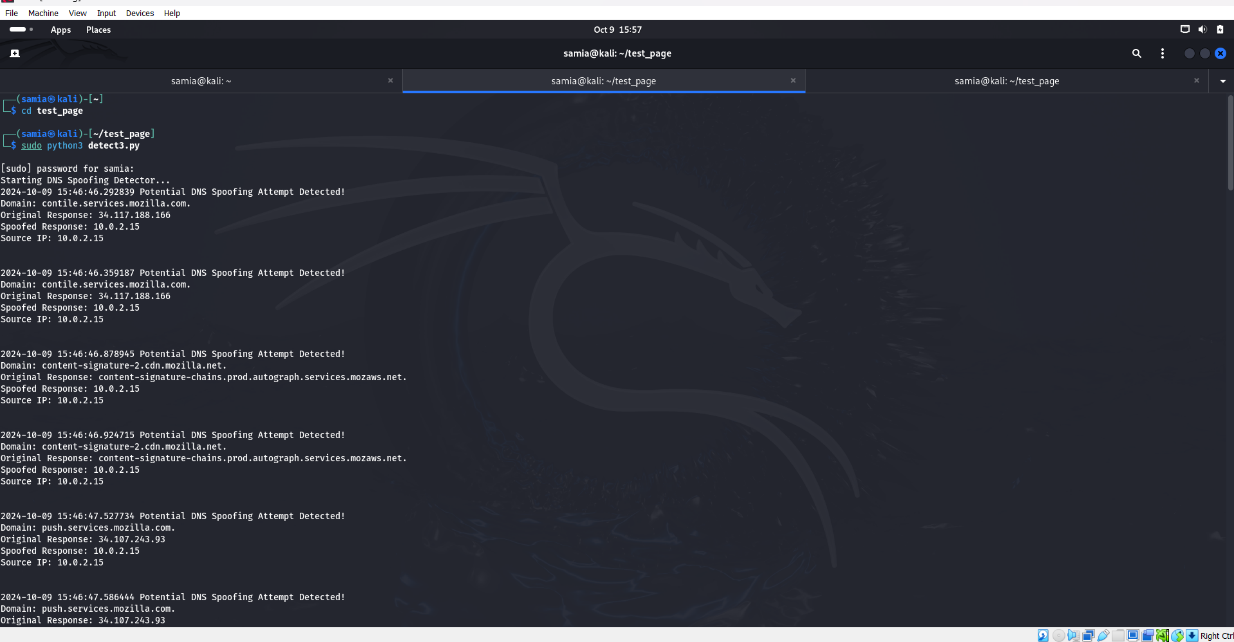
**Results screenshots showing outputs:**

Figure : Detecting terminal

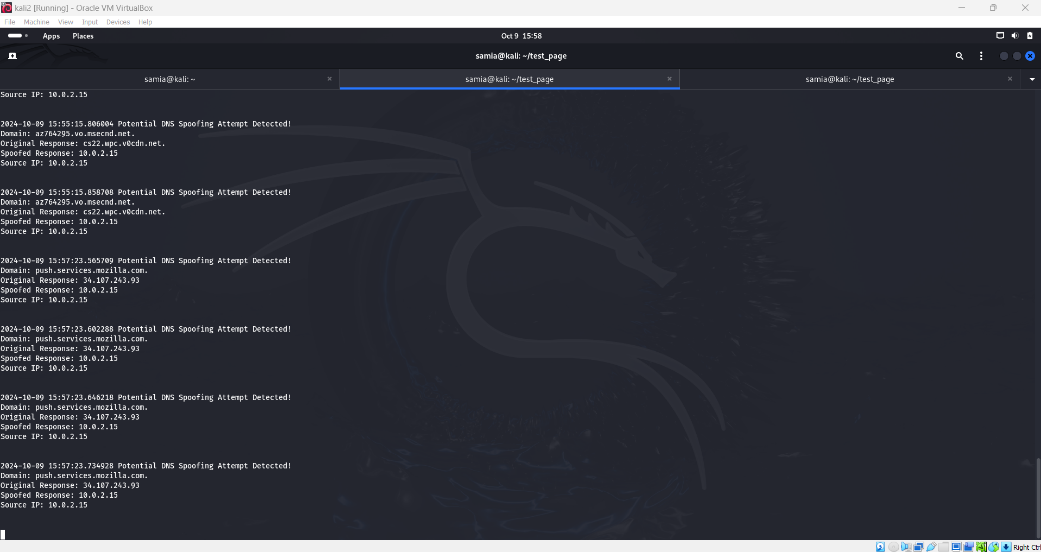


Figure : Detectin termianl cont.

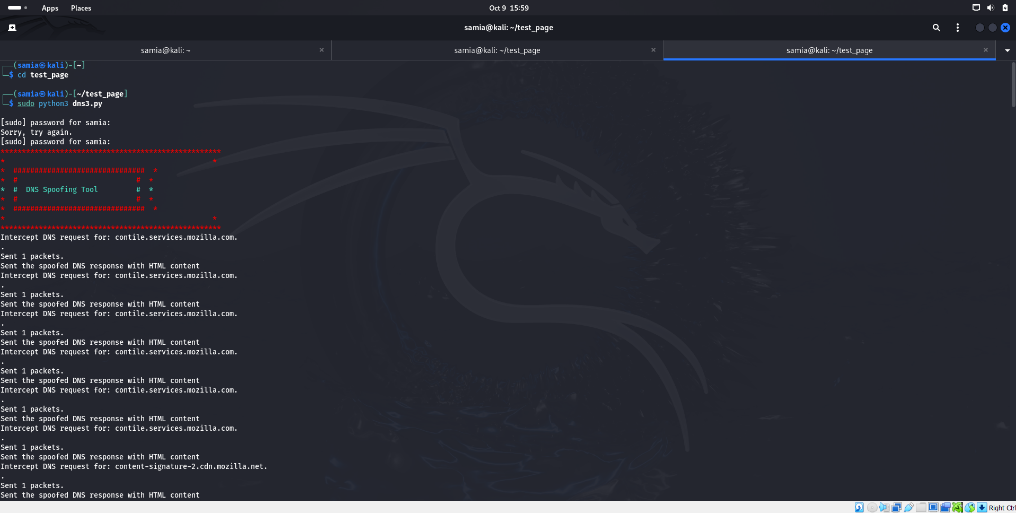


Figure : spoofing terminal

A screenshot of a computer

Description automatically generated

Figure : spoofing terminal cont.

A screenshot of a computer

Description automatically generated

Figure : http server terminal

A screenshot of a computer

Description automatically generated

Figure : Browser result

**Contribution:**

Samia Noaman: Did 100% of the project

Alaa Khanafer: Did not contribute to the project due to the ongoing circumstances