A black background with red text

Description automatically generated

# Cover Page

**EECE 655 Internet Security**

**Title:**

**ASSIGNMENT 1**

*Prepared by:*

**Ali El Nehmani**

**Fatima Hammoud**

*Submitted to:*

**Prof. IMAD H. ELHAJJ**

**Fall 2024**

Table of Contents

[Cover Page 1](#_Toc179561130)

[1) Introduction: 3](#_Toc179561131)

[2) Detailed Description of Tools: 3](#_Toc179561132)

[a) Attack Tool: 3](#_Toc179561133)

[b) Detection Tool: 4](#_Toc179561134)

[3) Team members’ contributions: 5](#_Toc179561135)

[a) Fatima Hammoud: 5](#_Toc179561136)

[b) Ali El Nehmani: 5](#_Toc179561137)

[4) Instructions on How to Run the Programs: 5](#_Toc179561138)

[Description of the testing environment (controlled environment details): 5](#_Toc179561139)

[5) Testing Results: 6](#_Toc179561140)

[a) Attack Tool: 6](#_Toc179561141)

[b) Detection Tool: 7](#_Toc179561142)

[6) References: 9](#_Toc179561143)

## Introduction:

* Our aim is to develop a tool that disrupts a victim's ability to access specific websites. The methodology starts with packet sniffing to get the victim's MAC address, then ARP spoofing to initiate a Man-in-the-Middle attack, allowing the attacker to intercept traffic between the victim and the gateway. The tool then performs DNS spoofing, redirecting the victim’s web traffic to malicious websites instead of the intended destinations. Additionally, an effective detection tool is developed to identify/detect these attacks.

## Detailed Description of Tools:

## Attack Tool:

* **Purpose:** The attack tool performs ARP spoofing and DNS spoofing attacks to intercept and manipulate network traffic.
* **Libraries used:**
* **Scapy:** Used to create and send ARP requests and responses, and to intercept and modify DNS responses.
* **Subprocess:** used to execute system-level commands, such as configuring IP tables to redirect network traffic for the DNS spoofing attack.
* **Time:** provides time-related functions to manage packet sending intervals and ARP spoofing loops.
* **IPaddress:** verifies whether the IP addresses entered for DNS redirection are valid IPv4 addresses.
* **Colorama:** used for enhancing the output in the terminal with colors, making it easier to distinguish between different types of messages (alerts, errors, prompts)
* **Netfilterqueue:** allows the tool to interact with the Netfilter queue in Linux, capturing and processing packets before they are forwarded.
* **Attack Mechanism:**
* The attacker sends malicious ARP replies to both the victim and the gateway.
* The victim is tricked into associating the attacker’s MAC address with the gateway’s IP address, while the gateway is tricked into associating the attacker’s MAC address with the victim’s IP address.
* This places the attacker in the middle of the communication, allowing them to intercept and relay or modify any traffic between the victim and the gateway.
* Once the ARP spoofing is in place, the attacker is positioned as a man-in-the-middle, intercepting all traffic including DNS queries, HTTP requests and other formats of traffic.
* When the victim makes a DNS request to a specific website, the attacker intercepts the request and sends a fake DNS response, redirecting the victim to a malicious webpage instead of the legitimate site they were trying to visit.

## Detection Tool:

* **Purpose:** The detection tool monitors network traffic to identify various attack patterns, including ARP spoofing and DNS spoofing.
* The script implements a logging system to record detection alerts in a file named attack\_detection.log for later review and analysis.
* **Libraries used:**
* **Scapy:** Used for packet sniffing and analyzing network traffic.
* **dnspython:** Utilized for DNS resolution, DNSSEC validation, and PTR record lookups**.**
* **logging:** Records detection events and alerts to a log file to track potential attacks.
* **Detection Mechanisms:**
* ARP spoofing detection occurs when the packet sniffer captures ARP traffic. The sniff() function monitors network packets, and the packet\_monitor() function processes them. If an ARP packet is detected, it triggers the detect\_arp\_spoofing() function.
* The code logs ARP spoofing detections. When detect\_arp\_spoofing() identifies a mismatch between the real MAC address (from getmacbyip()) and the MAC address from the ARP reply (hwsrc), it logs a warning using the logging.warning() function.
* **DNS Spoofing Detection:** Uses DNSSEC to validate DNS responses and checks PTR records to verify the IP address linked to a domain. Warnings are logged for any DNS spoofing.
* DNSSEC enhances DNS security by validating responses using digital signatures through the **validate\_dnssec()** function, which queries DNSKEY records to ensure data integrity. PTR records facilitate reverse lookups, mapping IP addresses to domain names via the **resolve\_ptr()** function.
* **detect\_dns\_spoofing(pkt):** This function detects DNS spoofing by first validating DNS responses with DNSSEC. If validation fails, it checks the PTR record for the response IP to confirm it matches the queried domain name.
* **dns.query():** This function, part of the **dnspython** library, is used to send DNS queries over UDP, such as querying DNSKEY records in **validate\_dnssec().**

## Team members’ contributions:

## Fatima Hammoud:

* Fatima was responsible for developing the attack tool and writing her portion of the report. Additionally, she took charge of setting up the environment and testing both tools, verifying their functionality and ensuring they work as intended and recorded the video.

## Ali El Nehmani:

* Ali was responsible for developing the detection tool, organizing and contributing to the writing of the report & ensuring all required materials were submitted, also handled debugging any errors that arose in both tool scripts.

## Instructions on How to Run the Programs:

## Description of the testing environment (controlled environment details):

* The environment consists of two virtual machines:
* Kali Linux: the attacker machine
* Windows 10: the defender machine
* Configuring the network:
* Configure both machines to be on the same network type, explicitly, NAT mode. This allows the attacker to intercept traffic from the victim.
* Test the connection by pinging each machine from the other.
* Note that the windows firewall might be set to not reply to ICMP echo requests. Add a new rule to allow ICMP echo replies.
* To isolate dependencies and avoid conflicts, a Python virtual environment is created on Kali. This ensures that the libraries required for the DNS spoofing tool are installed without affecting the system-wide Python environment.

**Commands:**

Sudo apt install python3-venv

Python3 –m venv myenv

Source myenv/bin/activate

* Install scapy and netfilterqueue libraries in the virtual environment
* Enable IP forwarding in Kali: echo 1 | sudo tee /proc/sys/net/ipv4/ip\_forward
* Configure IP tables: sudo iptables –I FORWARD –j NFQUEUE –queue-num 0
* Open the windows 10 virtual machine, paste the defend\_tool.py code on a text editor on Desktop. Make sure you have python and npcap installed on your machine.
* Open the command prompt (run as administrator), and type:

cd <path/to/your/code>

python defend\_tool.py

The defend tool is now running on your machine.

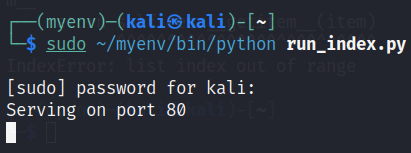
* Open three terminals on Kali, and activate the python environment in each using the command: Source myenv/bin/activate
* On the first terminal, run run\_index.py using: sudo ~/myenv/bin/python run\_index.py
* On the second terminal, run arp\_spoof.py using: sudo ~/myenv/bin/python arp\_spoof.py
* On the third terminal, run dns\_spoof.py using: sudo ~/myenv/bin/python dns\_spoof.py
* Enter in the first prompt that appears the IP address of your Kali machine, and in the second prompt [www.example.com](http://www.example.com). Your attack is now running.

## Testing Results:

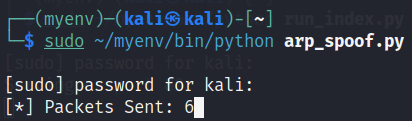
**Screenshots showing outputs from both tools:**

## Attack Tool:

* Hosting HTTP server on Kali Linux



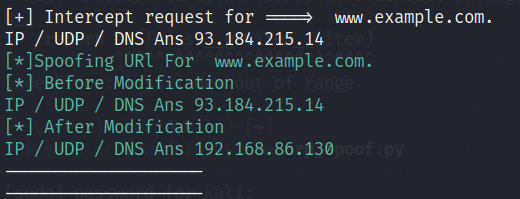
* Starting ARP spoofing on Kali Linux



* Starting DNS spoofing on Kali Linux

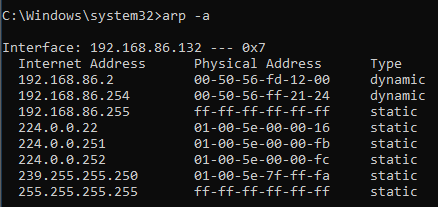


* Output of dns\_spoof when [www.example.com](http://www.example.com) is accessed on Windows 10 VM

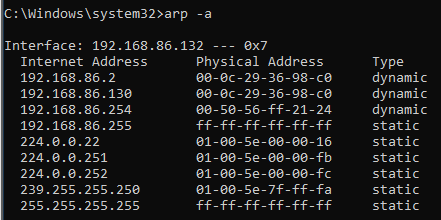


## Detection Tool:

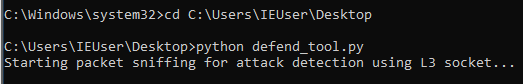
* ARP cache before conducting attack



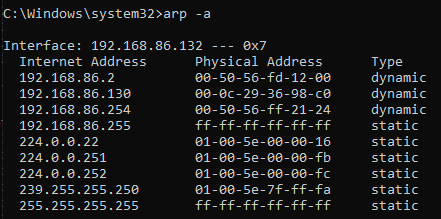
* ARP cache after conducting attack (defend\_tool not running)



* Starting defend\_tool on Windows 10 VM



* ARP cache after conducting attack (defend\_tool running)

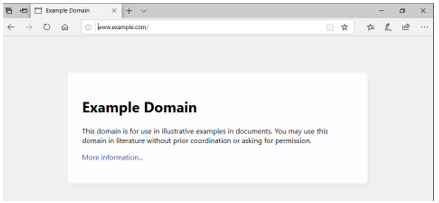


* Output when accessing [www.example.com](http://www.example.com) on Windows VM before running the defend\_tool

A screenshot of a computer

Description automatically generated

* Output when accessing [www.example.com](http://www.example.com) on Windows VM after running the defend\_tool



## References:

|  |  |
| --- | --- |
| [1] | Geeks, "Packet sniffing using Scapy," 5 July 2021. [Online]. Available: https://www.geeksforgeeks.org/packet-sniffing-using-scapy/. |
| [2] | Geeks, "How to Make a DNS Spoof attack using Scapy in Python?," 14 Oct 2022. [Online]. Available: https://www.geeksforgeeks.org/how-to-make-a-dns-spoof-attack-using-scapy-in-python/?ref=oin\_asr6. |
| [3] | Geeks, "How to Detect ARP Spoof Attack using Scapy in Python?," 28 June 2022. [Online]. Available: https://www.geeksforgeeks.org/how-to-detect-arp-spoof-attack-using-scapy-in-python/?ref=oin\_asr9. |
| [4] | Cloudflare, "How DNSSEC Works," [Online]. Available: https://www.cloudflare.com/en-gb/dns/dnssec/how-dnssec-works/. |
| [5] | R. Kumar, "Creating a DNS Spoofing Tool with Python: A Step-by-Step Guide," 6 Apr 2023. [Online]. Available: https://systemweakness.com/creating-a-dns-spoofing-tool-with-python-a-step-by-step-guide-a3217b3c4905. |
| [6] | Cloudflare, "What is a DNS PTR record?," [Online]. Available: https://www.cloudflare.com/en-gb/learning/dns/dns-records/dns-ptr-record/. |