

# PROJECT EXHIBITION

## REVIEW 2

### DNS Spoofing

### Detection Protection Tool



GUIDED BY-

Dr. Hariharasitaraman.S, SCSE,  
VIT Bhopal  
University.



# Team members

Sarthak Yadav 21MEI10036

Raghunandan Tomar 21MEI10024

Vakul Raina 21MEI10040

Aryan khatik 21MEI10004

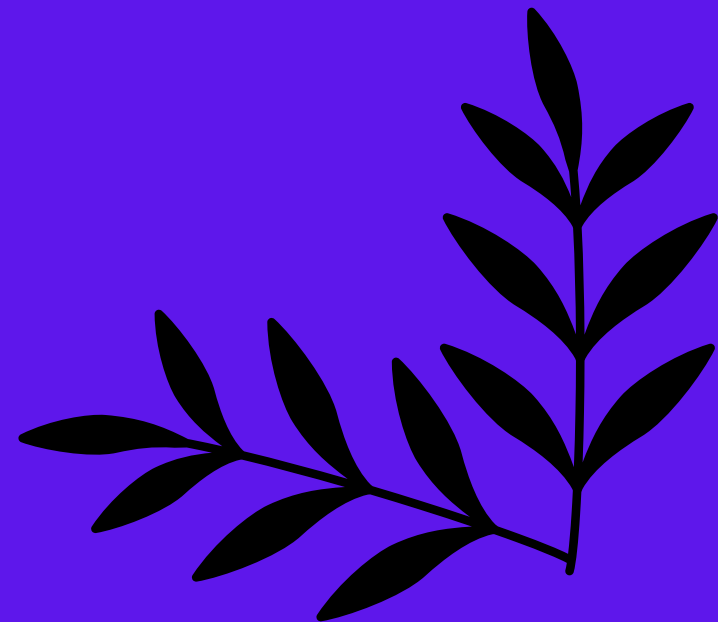




# **CONTENTS**

---

1. Introduction
2. Objective
3. Problem Statement
4. Literature Review
5. Proposed Work
6. Modules
7. Implementation and demo
8. Timeline
9. Reference





# Introduction

- DNS stands for Domain Name Server. So, DNS spoofing is an attack wherein the attacker can redirect the victim to another website which is not intended to be visited by the victim by injecting a fake DNS record in the DNS cache. As a result of these fake DNS entries, the victim can be redirected to any IP of the attacker's choosing.
- This attack can be easily performed in the Kali Linux environment using various inbuilt tools.
- One such tool that we used is Ettercap. Using ettercap, we first poison the victim's ARP cache with our MAC address.
- Then we search for available hosts in the network and add the gateway as TARGET2 and the victim as TARGET1 and start unified sniffing.
- Finally we add the dns\_spoof plugin that is available in ettercap and wait for some time for it to start.
- Once we get the message that the target website's DNS has been spoofed, we can check the victim machine and try opening the website in it. We observe that the victim is redirected to some other website even though the address bar says that it is the original website.

# Objective

- Protection of the confidential information of the users is crucial to prevent certain cyber attacks. In the DNS poisoning attack, end-users are particularly vulnerable.
- We have made a simple python program that browses different IP addresses in the ARP cache and checks if any two IP addresses have the same MAC addresses or not. If they have the same MAC address, that means the ARP cache is poisoned.
- Once the poisoning is confirmed, the user is prompted with a message saying that the ARP cache has been poisoned. Along with this message, the attacker's MAC address is also displayed.

# Problem Statement

Even though these attacks have a limitation of only working on LAN, these are very common and easily performed and virtually undetectable by a novice. As a result, this is a threat that has to be addressed. So, we have come up with a python program which is capable of detecting, preventing and recovering from an ARP poisoning attack on the victim's linux system by which man in the middle attack can be stopped and dns won't be spoofed

# Literature Review

1. Detection :- a simple python program that browses different IP addresses in the ARP cache and checks if any two IP addresses have the same MAC addresses or not. If they have the same MAC address, that means the ARP cache is poisoned. Once the poisoning is confirmed, the user is prompted with a message saying that the ARP cache has been poisoned. Along with this message, the attacker's MAC address is also displayed.

2. Prevention :- For prevention, we designed an algorithm and coded it in python. This algorithm does the task of making all the entries in the ARP cache as static because of the fact that they cannot be altered by the attacker once made static. As a result, no matter how many fake ARP replies are sent to the victim, its ARP cache won't be changed.

3. Recovery :- Once we have detected that ARP cache is poisoned, we can execute the recovery algorithm which is also coded in python. The main idea of the recovery process is copying a clean set of entries into the cache or altering a specific entry in the cache. The clean ARP cache entries are stored when the application is started, provided that the cache is cleaned while starting it. If the user knows the MAC address of the gateway, or any other node in the network, they can enter the ARP entry using one of the features of the recovery system.

# Proposed work

- The objective of this project is to develop a prevention against DNS spoofing by using a python program
- The program will help user from getting their Arp poisoned



# Modules

There are 3 modules :

1.Detection

2.Prevention

3.Recovery

## Detection of attack :-

## Prevention :-

- First we'll observe the ARP cache.
- Here the flag masks are denoted with 'C', that means it's dynamic and hence our system is vulnerable to the attack.

### ANTI-ARP POISONING TOOL

-----

- 1.Scan for possible ARP attacks
- 2.View your ARP cache
- 3.Create a static ARP entry
- 4.Make all entries of ARP as static
- 5.Clear ARP cache
- 6.Run continuous test
- 7.Exit

Enter your option:

2

Address	HWtype	HWaddress	Flags Mask	Iface
_gateway	ether	52:54:00:12:35:00	C	enp0s3

- Now, in order to prevent the attack from being performed, we have to change the ARP cache from dynamic to static.

```
ANTI-ARP POISONING TOOL
```

```
-----
```

```
1.Scan for possible ARP attacks
2.View your ARP cache
3.Create a static ARP entry
4.Make all entries of ARP as static
5.Clear ARP cache
6.Run continuous test
7.Exit
```

```
Enter your option:
```

```
4
```

```
Enter password:sky
```

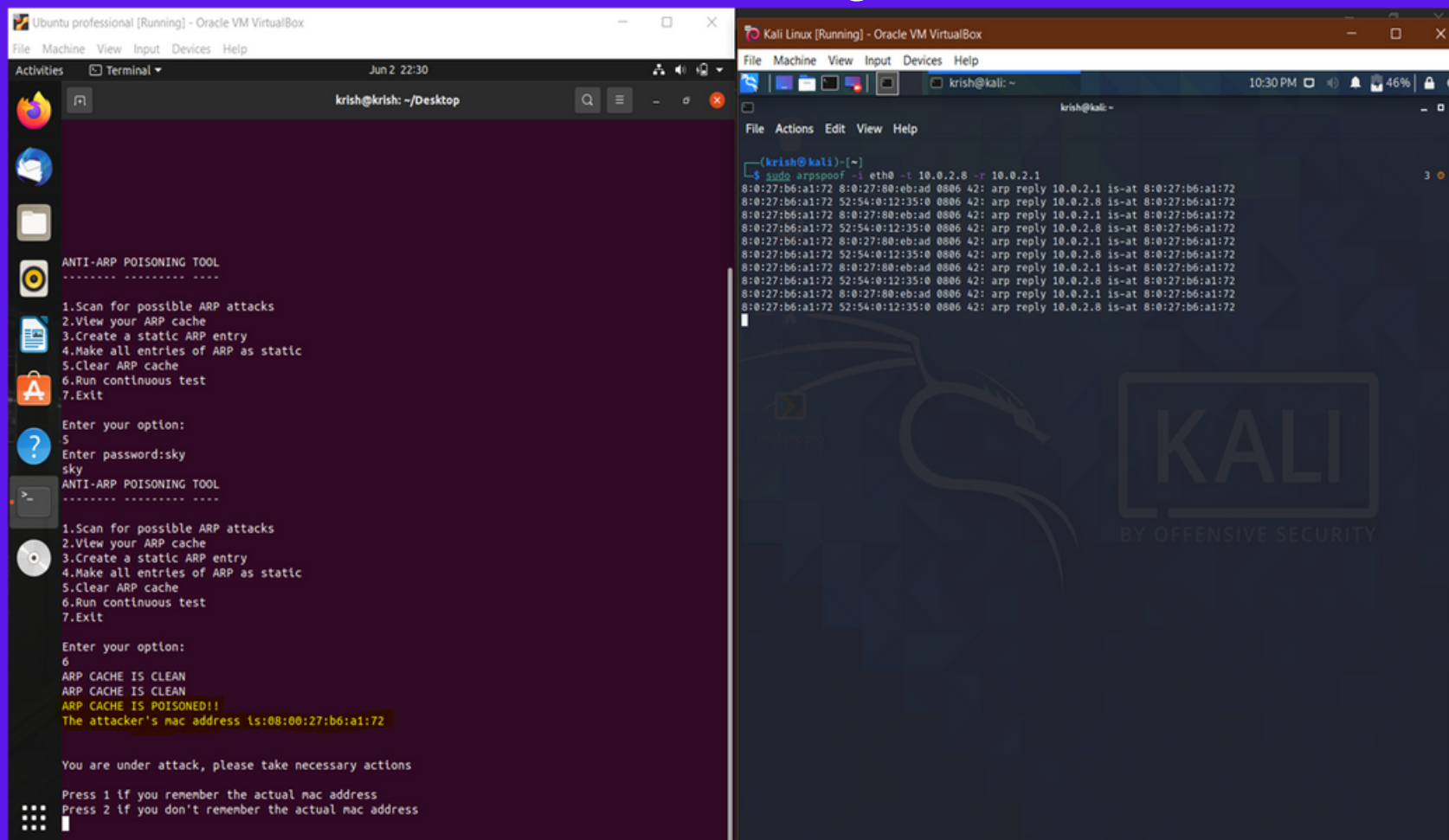
- Here as we can see, even if we perform the attack, the ARP cache is clean.

The image shows two terminal windows side-by-side. The left window is titled 'Ubuntu professional [Running] - Oracle VM VirtualBox' and shows a terminal session with the user 'krish' at the prompt 'krish@krish: ~/Desktop'. The terminal output includes a table with headers 'Address', 'HWtype', 'HWaddress', 'Flags Mask', and 'Iface'. The table has one row with values: '\_gateway', 'ether', '52:54:00:12:35:00', 'C', and 'enp0s3'. Below the table is a menu for 'ANTI-ARP POISONING TOOL' with options 1 through 7. The user enters option 4, and the terminal displays 'Enter password:sky' and 'ANTI-ARP POISONING TOOL'. The user then enters option 1, and the terminal displays '1.Scan for possible ARP attacks'. The user enters option 2, and the terminal displays '2.View your ARP cache'. The user enters option 3, and the terminal displays '3.Create a static ARP entry'. The user enters option 4, and the terminal displays '4.Make all entries of ARP as static'. The user enters option 5, and the terminal displays '5.Clear ARP cache'. The user enters option 6, and the terminal displays '6.Run continuous test'. The user enters option 7, and the terminal displays '7.Exit'. The terminal then displays 'Enter your option:' and the user enters '1'. The terminal then displays 'ARP CACHE IS CLEAN'. A red arrow points from this message to the right terminal window.

The right window is titled 'Kali Linux [Running] - Oracle VM VirtualBox' and shows a terminal session with the user 'krish' at the prompt 'krish@kali: ~'. The terminal output shows the command 'sudo arpspoof -i eth0 -t 10.0.2.8 -r 10.0.2.1' being executed. The output of the command is a series of lines showing 'arp reply' messages from '10.0.2.1' to '10.0.2.8' at various times. The terminal also shows a large 'KALI' logo with the text 'BY OFFENSIVE SECURITY' below it.

# Recovery :-

- Here we have an option to run the test continuously.



The image displays two terminal windows from Oracle VM VirtualBox. The left window, titled 'Ubuntu professional [Running] - Oracle VM VirtualBox', shows the 'ANTI-ARP POISONING TOOL' interface. It lists seven options: 1. Scan for possible ARP attacks, 2. View your ARP cache, 3. Create a static ARP entry, 4. Make all entries of ARP as static, 5. Clear ARP cache, 6. Run continuous test, and 7. Exit. The user enters option 5, then option 6, and the tool displays 'ARP CACHE IS CLEAN' and 'The attacker's mac address is:08:00:27:b6:a1:72'. The right window, titled 'Kali Linux [Running] - Oracle VM VirtualBox', shows a terminal session where the user runs the command 'sudo arpspoof -i eth0 -t 10.0.2.8 -r 10.0.2.1'. The output shows a continuous stream of ARP replies from the attacker's MAC address to the victim's MAC address.

```
krish@krish: ~/Desktop
ANTI-ARP POISONING TOOL
-----
1.Scan for possible ARP attacks
2.View your ARP cache
3.Create a static ARP entry
4.Make all entries of ARP as static
5.Clear ARP cache
6.Run continuous test
7.Exit

Enter your option:
5
Enter password:sky
sky
ANTI-ARP POISONING TOOL
-----
1.Scan for possible ARP attacks
2.View your ARP cache
3.Create a static ARP entry
4.Make all entries of ARP as static
5.Clear ARP cache
6.Run continuous test
7.Exit

Enter your option:
6
ARP CACHE IS CLEAN
ARP CACHE IS CLEAN
ARP CACHE IS POISONED!!
The attacker's mac address is:08:00:27:b6:a1:72

You are under attack, please take necessary actions

Press 1 if you remember the actual mac address
Press 2 if you don't remember the actual mac address
```

```
(krish@kali)~$ sudo arpspoof -i eth0 -t 10.0.2.8 -r 10.0.2.1
8:0:27:b6:a1:72 8:0:27:80:eb:ad 0806 42: arp reply 10.0.2.1 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 52:54:0:12:35:0 0806 42: arp reply 10.0.2.8 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 8:0:27:80:eb:ad 0806 42: arp reply 10.0.2.1 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 52:54:0:12:35:0 0806 42: arp reply 10.0.2.8 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 8:0:27:80:eb:ad 0806 42: arp reply 10.0.2.1 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 52:54:0:12:35:0 0806 42: arp reply 10.0.2.8 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 8:0:27:80:eb:ad 0806 42: arp reply 10.0.2.1 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 52:54:0:12:35:0 0806 42: arp reply 10.0.2.8 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 8:0:27:80:eb:ad 0806 42: arp reply 10.0.2.1 is-at 8:0:27:b6:a1:72
8:0:27:b6:a1:72 52:54:0:12:35:0 0806 42: arp reply 10.0.2.8 is-at 8:0:27:b6:a1:72
```

- Now once the attack is detected, it will ask us if we remember the actual mac address. In this case we don't.
- So the system will find the actual MAC address from the previously stored clean set of values and restructure the ARP cache.
- Once this is done, our ARP cache is clean again, as it was before the attack happened.







Code :-

click the link to view the code

<https://drive.google.com/file/d/1y2q3XeDSsrZAV7HL9KzJQQBWnQ7mUlbc/view?usp=sharing>

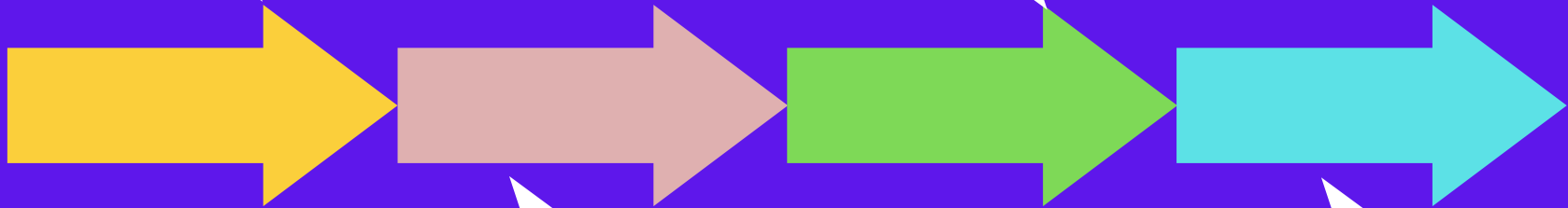
# TIMELINE

Discuss and gathered info about project

Completed modules

Planned and started working on our python program

completed and tested the project



# References

- [1] Y. Kim, S. Ahn, N. C. Thang, D. Choi and M. Park, "ARP Poisoning Attack Detection Based on ARP Update State in Software-Defined Networks," 2019 International Conference on Information Networking (ICOIN), Kuala Lumpur, Malaysia, 2019, pp. 366-371, doi: 10.1109/ICOIN.2019.8718158.
- [2] B. Zdrnja, "Malicious JavaScript Insertion through ARP Poisoning Attacks," in IEEE Security & Privacy, vol. 7, no. 3, pp. 72-74, May-June 2009, doi: 10.1109/MSP.2009.72.
- [3] S. Kumar and S. Tapaswi, "A centralized detection and prevention technique against ARP poisoning," Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), Kuala Lumpur, 2012, pp. 259-264, doi: 10.1109/CyberSec.2012.6246087.

*Thank  
you!*