

# **LAB 1: Setting up Environment**

## **I. Purpose**

Set up the environment for malware testing by using Kali Linux as DNS server, and Windows Server 2008 as victim running on VMware to simulate and observe simulated network traffic using INetSim

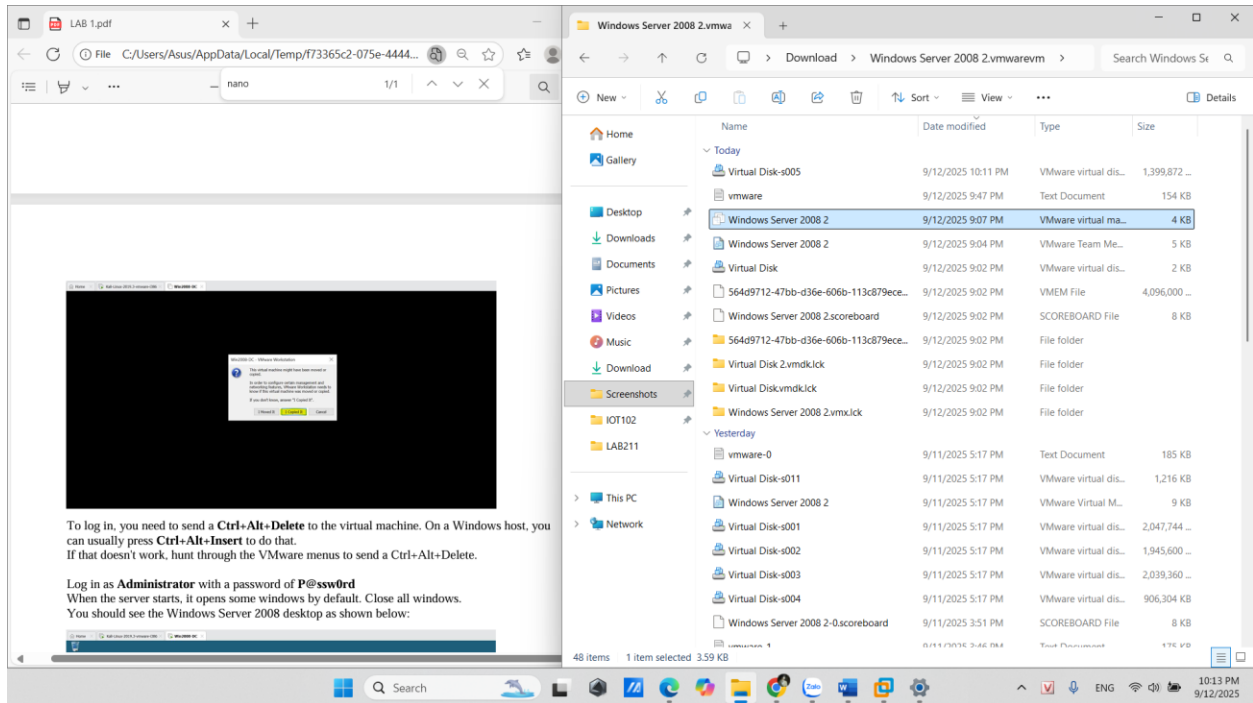
## **II. Target**

- Configure Kali (NAT) and INetSim to respond to DNS/HTTP queries from Windows machines
- Set up Window VM to point DNS to Kali and check InetSim page via browser
- Use nmap on Windows to scan fake domain (YOURNAME.com) and record services simulated by INetSim

## **III. Lab instruction:**

### **1. Set up virtual machine**

Firstly, download and unzip the file Window Server 2008



**Fig 1.1: Window Server 2008**

I also download the .iso file of Kali Linux

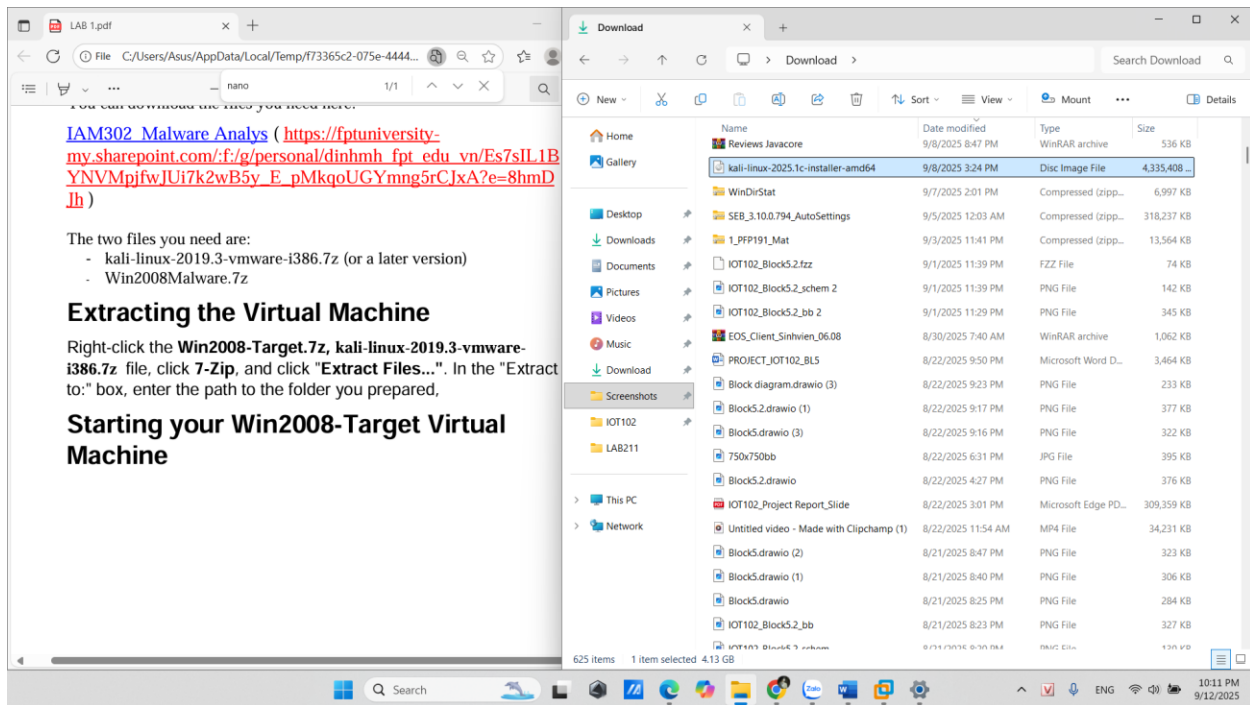
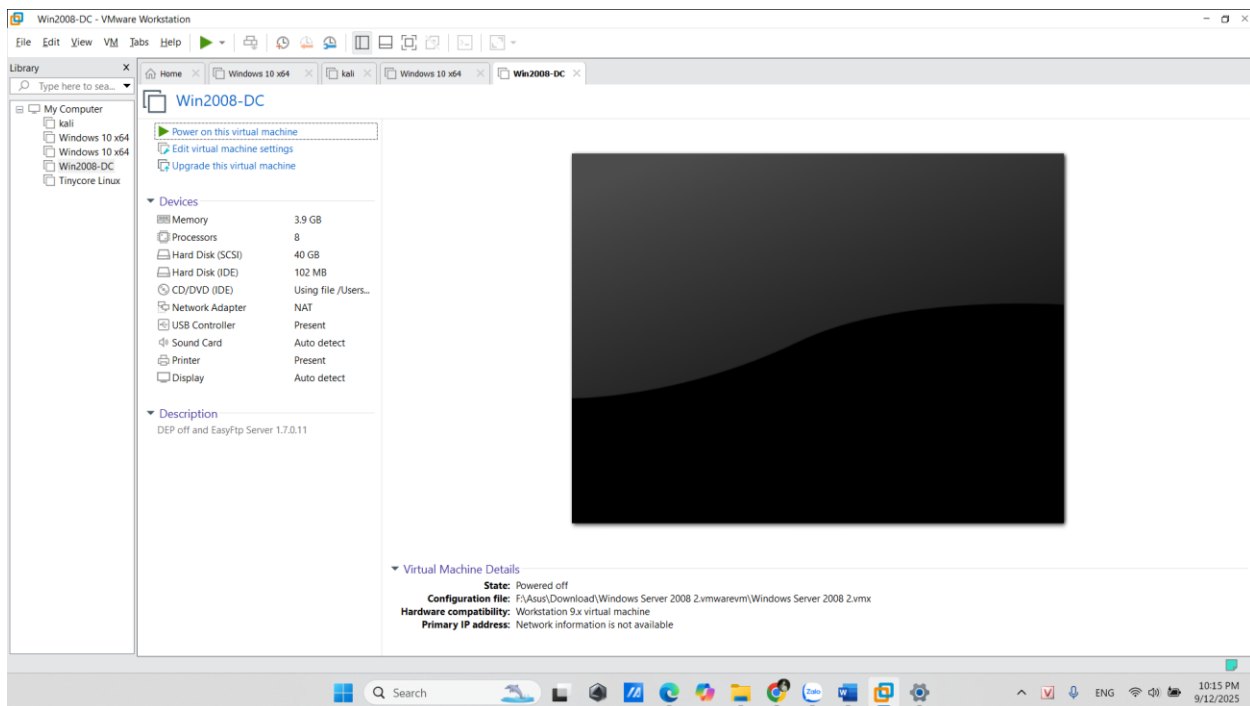
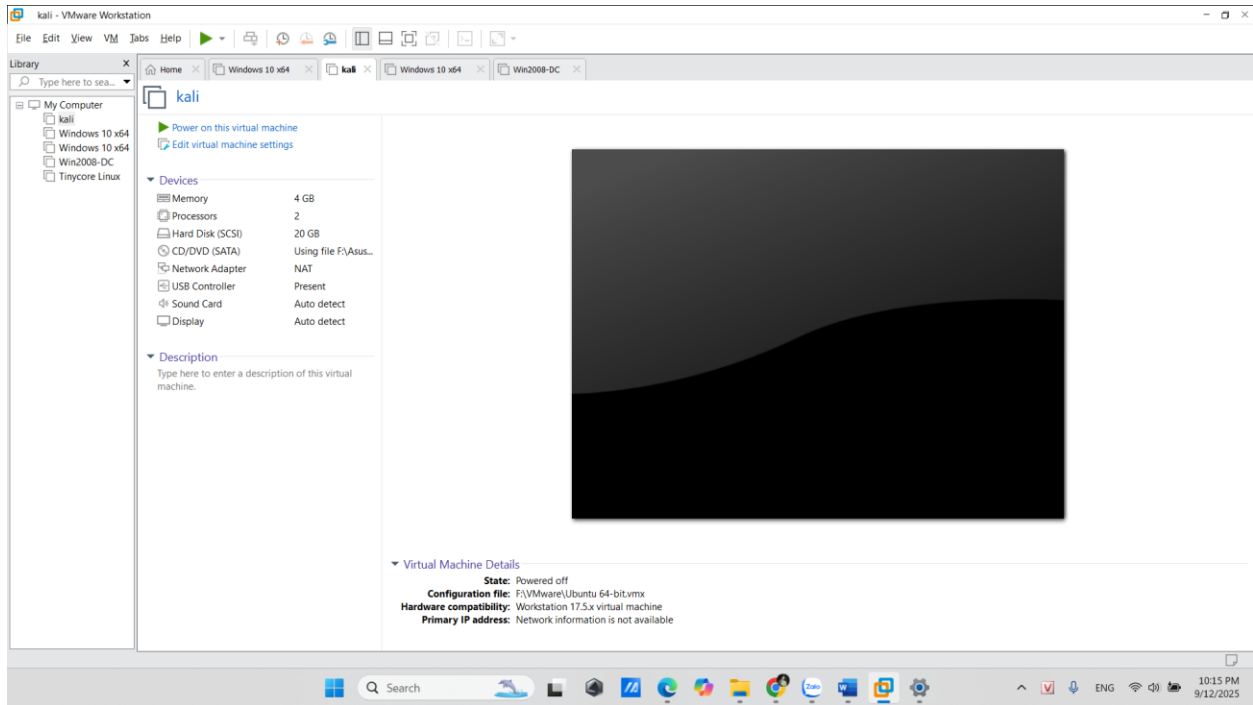


Fig 1.2: Kali Linux

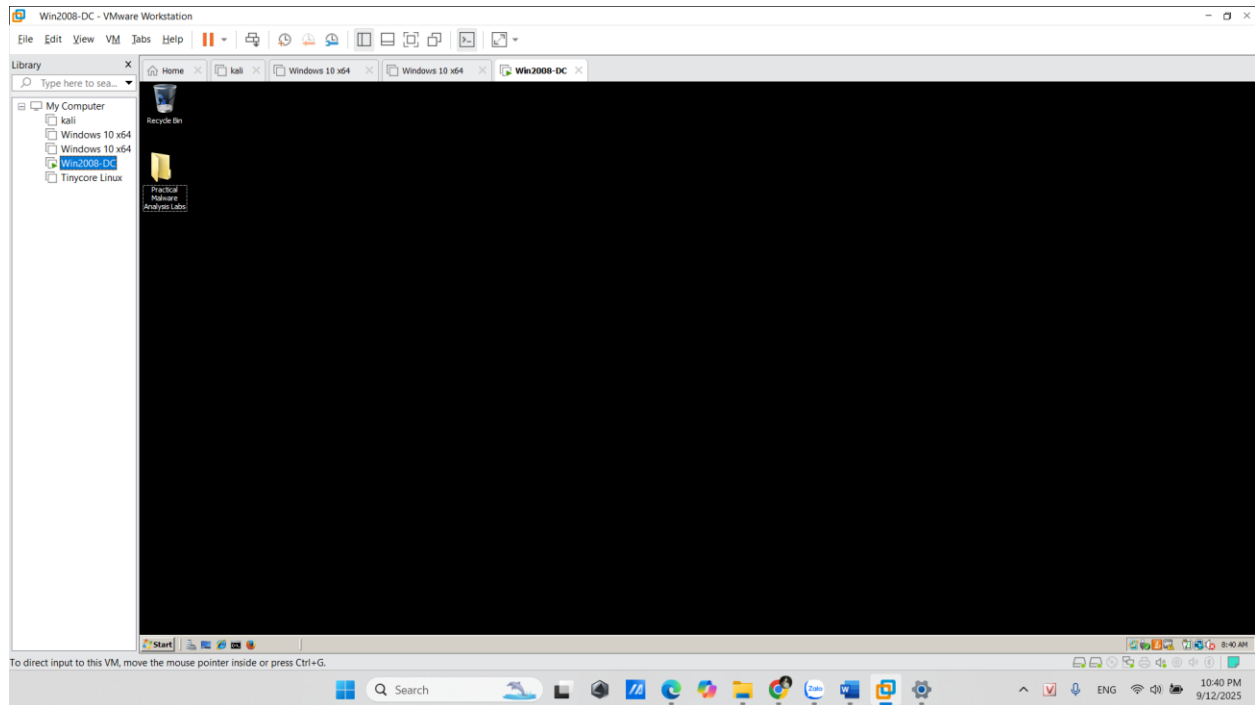
Next, I import two virtual machine into VMware Workstation





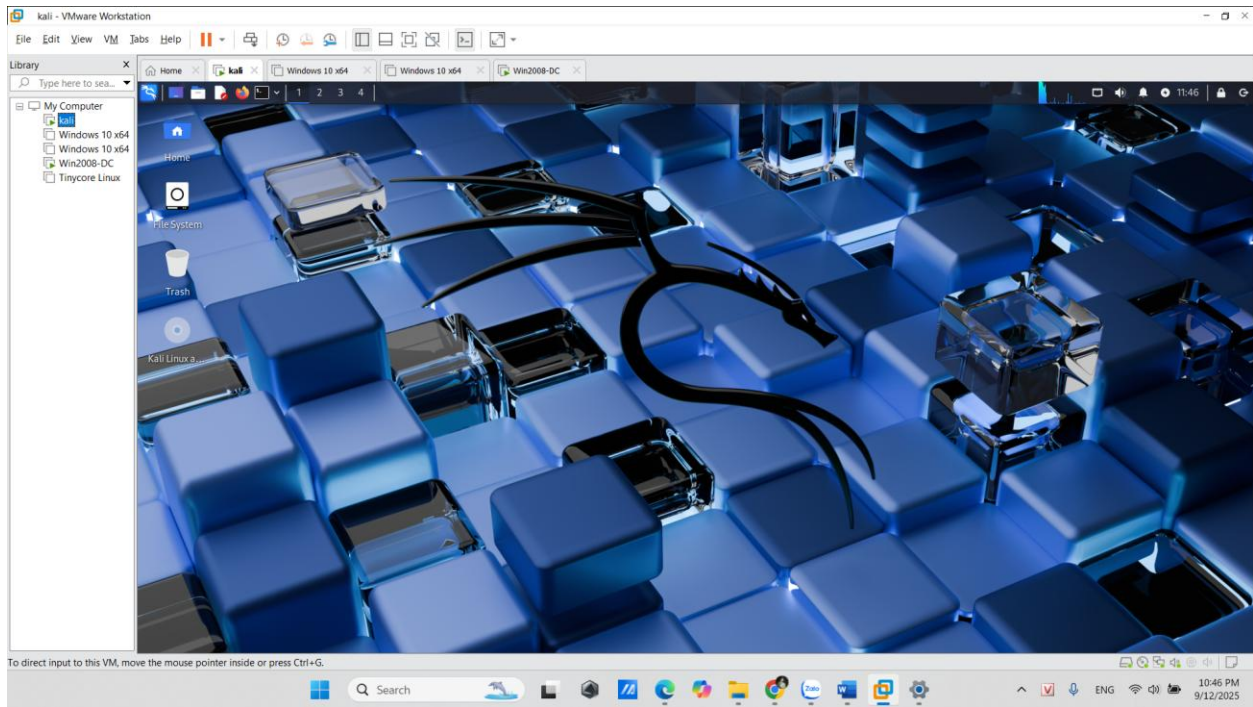
**Fig 1.3, 1.4: Setting virtual machine in VMware**

Log in Windows Server 2008 (target), log in as **Administrator**



**Fig 1.5: Window Server login screen**

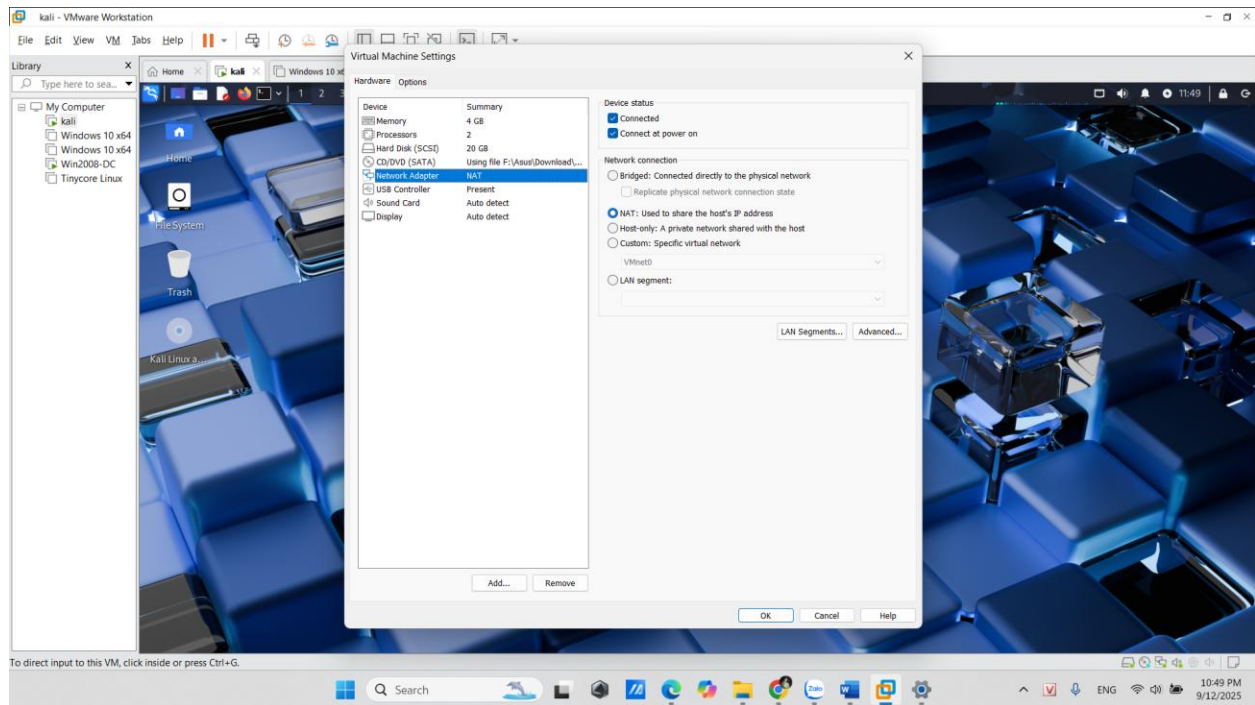
Log in Kali with the username **root** and a password of **toor**



**Fig 1.6: Kali Linux login screen**

## **2. Setting the Kali Linux VM to NAT Networking**

In the setting, change the **Network Adapter** of Kali into **NAT**, that let the VM can share their own specific IP Address with host machine.



**Fig 2.1: Virtual Machine Settings – Network Adapter Tab**

### **3. Finding the Kali Machine's IP Address**

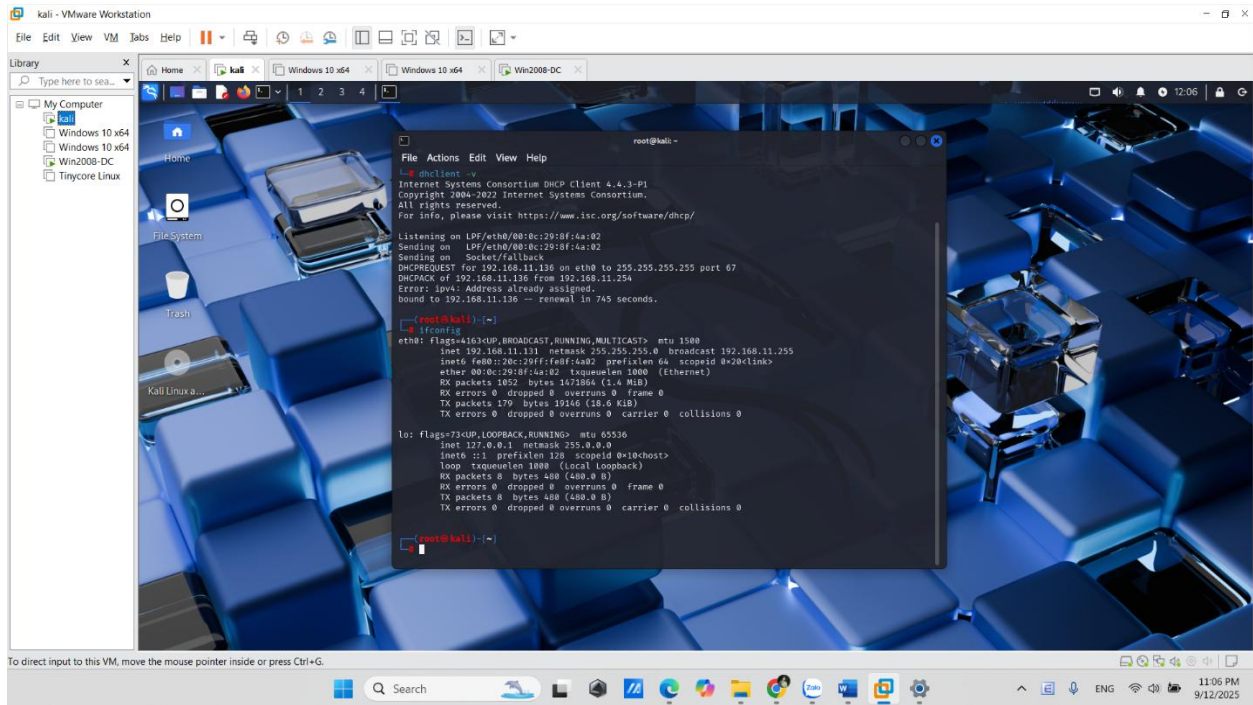
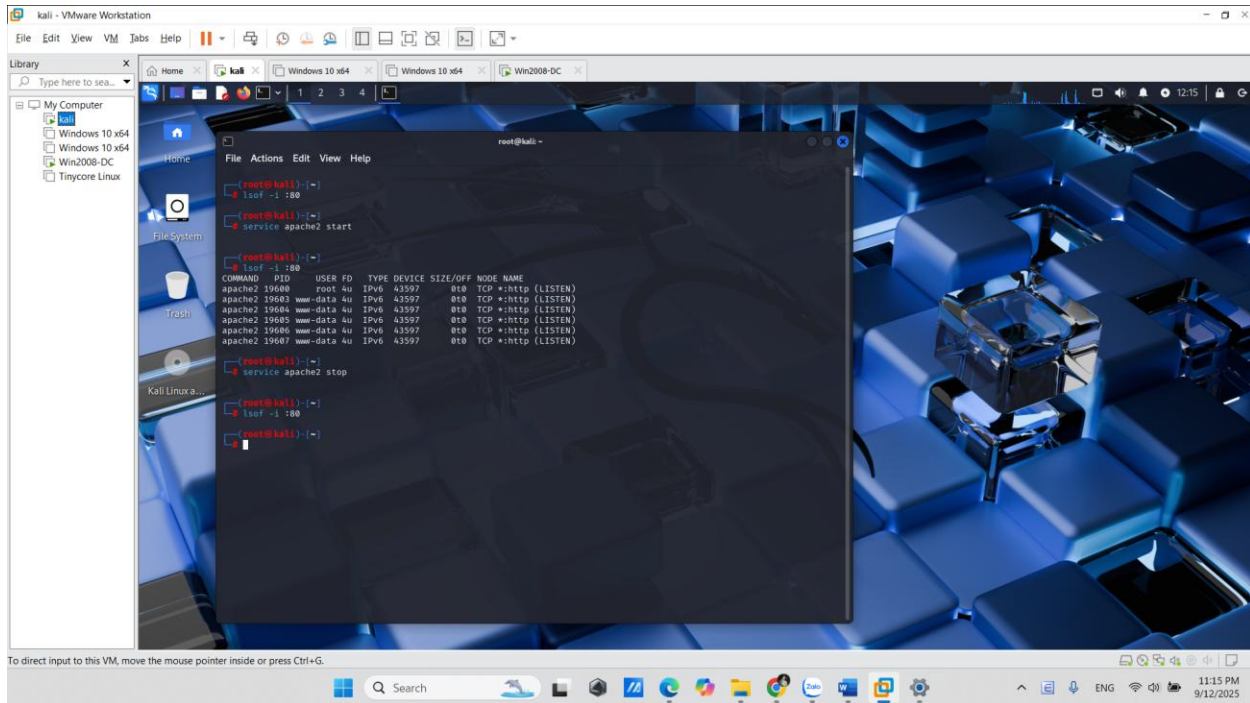


Fig 3.1: Kali Linux run dhcp on each adapter

## 4. Checking for a Web server and Configuring INetSim





**Fig 4.1: List of process that using port 80**

After starting apache2 for web server, we stop them and move to next step

First, backup config file: **cp /etc/inetsim/inetsim.conf /etc/inetsim/inetsim.conf.orig**

And then open config: **nano /etc/inetsim/inetsim.conf**

**Note:** Edit your IP address to your local IP you've got from previous step

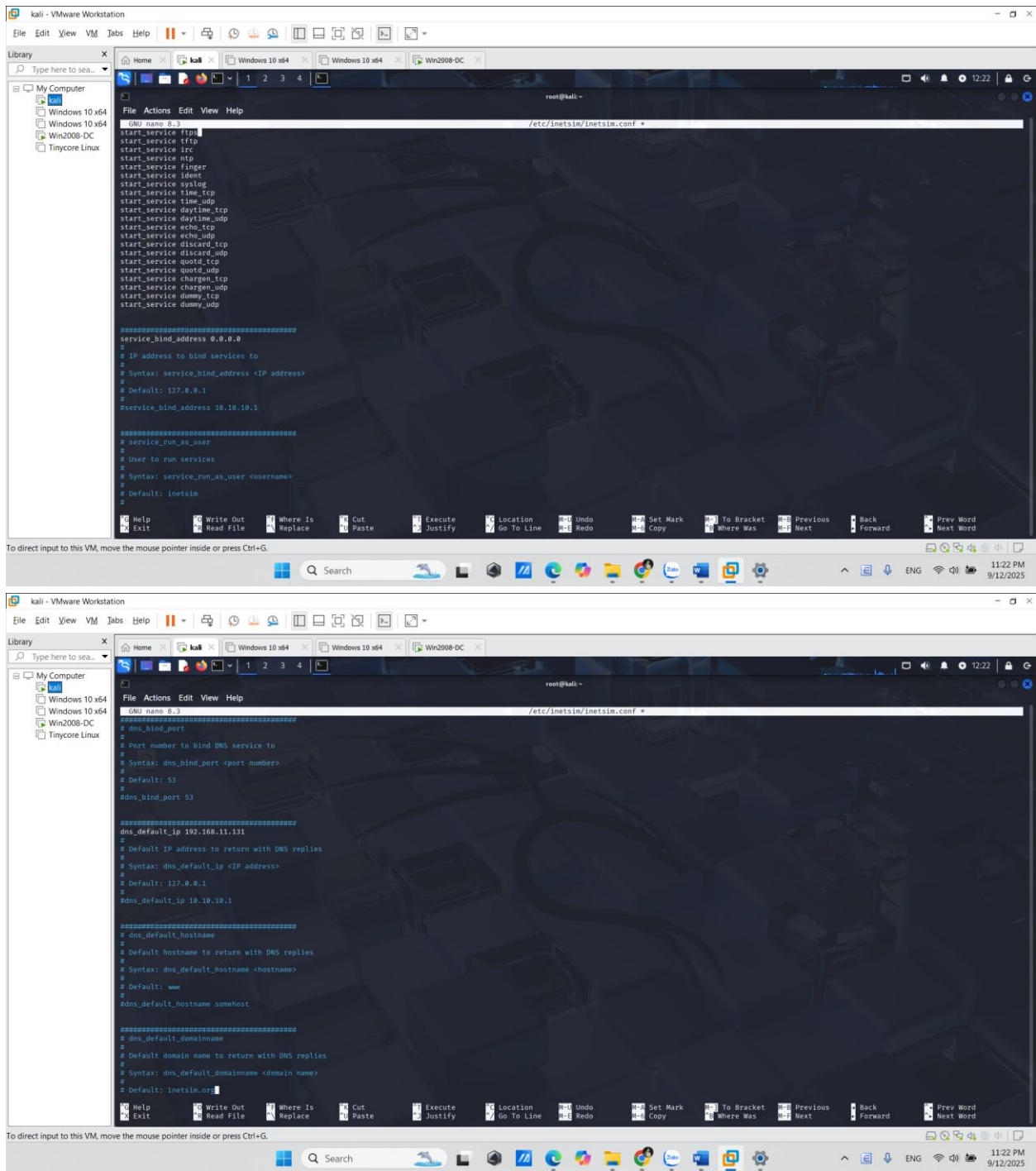
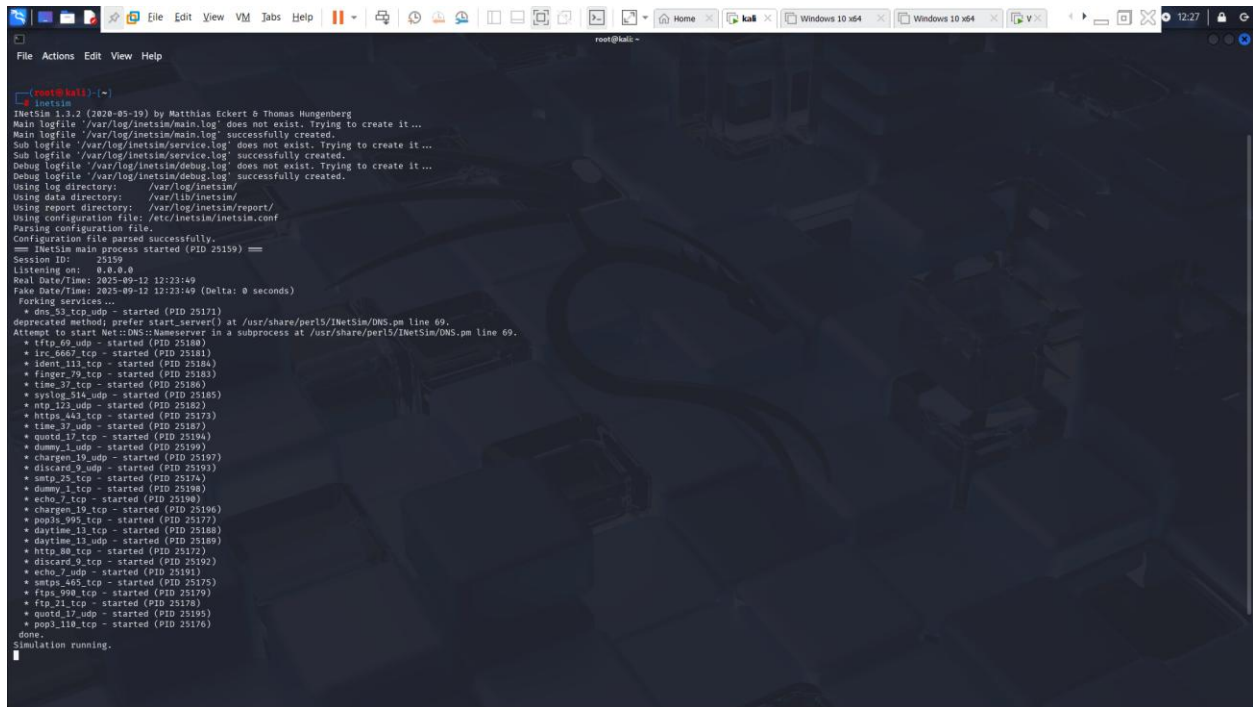


Fig 4.2: inetsim.conf

Save it and then execute **inetsim**



```
root@kali: ~  
[root@kali]# inetsim  
Inetsim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg  
Main logfile '/var/log/inetsim/main.log' does not exist. Trying to create it ...  
Main logfile '/var/log/inetsim/main.log' successfully created.  
Sub logfile '/var/log/inetsim/service.log' does not exist. Trying to create it ...  
Sub logfile '/var/log/inetsim/service.log' successfully created.  
Debug logfile '/var/log/inetsim/debug.log' does not exist. Trying to create it ...  
Debug logfile '/var/log/inetsim/debug.log' successfully created.  
Using log directory: /var/log/inetsim/  
Using data directory: /var/lib/inetsim/  
Using report directory: /var/log/inetsim/report/  
Using configuration file: /etc/inetsim/inetsim.conf  
Parsing configuration file.  
Configuration file parsed successfully.  
== Inetsim main process started (PID 25159) ==  
Session ID: 25159  
Listening on: 0.0.0.0  
Real Date/Time: 2025-09-12 12:23:49  
Fake Date/Time: 2025-09-12 12:23:49 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 25171)  
deprecated method: prefer start_server() at /usr/share/perl5/Inetsim/DNS.pm line 69.  
Attempt to start Net::DNS::Nameserver in a subprocess at /usr/share/perl5/Inetsim/DNS.pm line 69.  
* tftp_69_udp - started (PID 25180)  
* irc_6667_tcp - started (PID 25181)  
* ident_113_tcp - started (PID 25184)  
* finger_79_tcp - started (PID 25183)  
* time_37_tcp - started (PID 25186)  
* syslog_514_udp - started (PID 25185)  
* ntp_123_udp - started (PID 25182)  
* http_443_tcp - started (PID 25172)  
* time_37_udp - started (PID 25187)  
* quotd_17_tcp - started (PID 25196)  
* dummy_1_udp - started (PID 25199)  
* chargen_19_udp - started (PID 25197)  
* discard_9_udp - started (PID 25193)  
* smtp_25_tcp - started (PID 25174)  
* dummy_1_tcp - started (PID 25198)  
* echo_7_tcp - started (PID 25190)  
* chargen_19_tcp - started (PID 25196)  
* pop3_995_tcp - started (PID 25177)  
* daytime_13_tcp - started (PID 25188)  
* daytime_13_udp - started (PID 25189)  
* http_88_tcp - started (PID 25172)  
* discard_9_tcp - started (PID 25192)  
* echo_7_udp - started (PID 25191)  
* smtp_465_tcp - started (PID 25175)  
* rftp_990_tcp - started (PID 25170)  
* rftp_21_tcp - started (PID 25176)  
* quotd_17_udp - started (PID 25195)  
* pop3_110_tcp - started (PID 25176)  
done.  
Simulation running.
```

Fig 4.3: inetsim running

When running INetSim, an error occurred due to a missing or outdated **Net::DNS** Perl module. To fix it, we manually downloaded and installed the correct version using the following commands:

```
curl -LO https://www.net-dns.org/download/Net-DNS-1.22.tar.gz
```

```
tar -xvzf Net-DNS-1.22.tar.gz
```

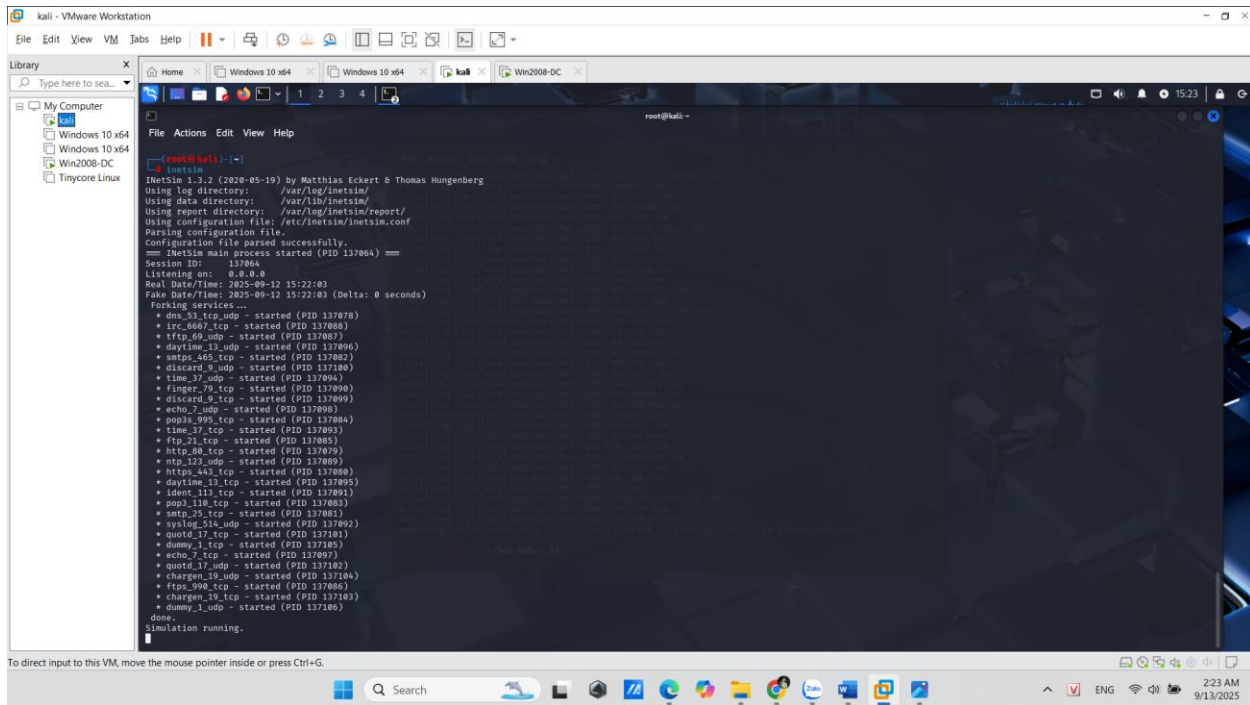
```
cd Net-DNS-1.22
```

```
perl Makefile.PL
```

```
make
```

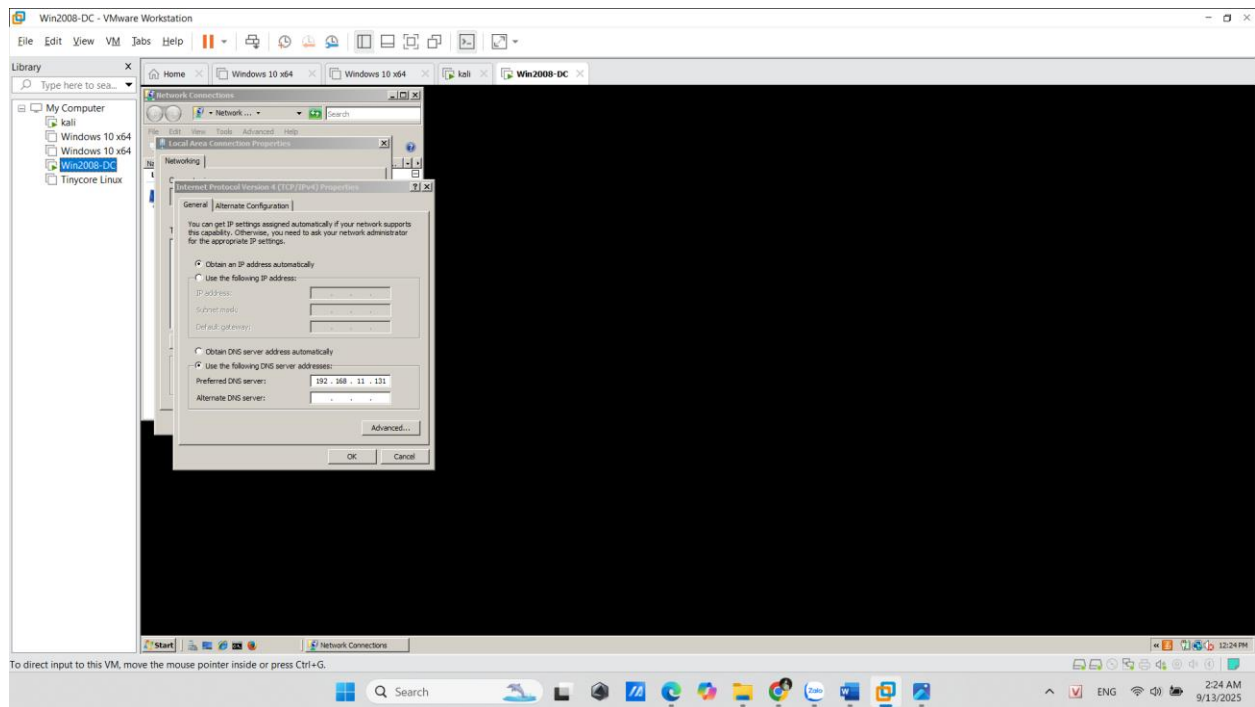
```
sudo make install
```

```
perl -MNet::DNS -e 'print "$Net::DNS::VERSION\n"
```

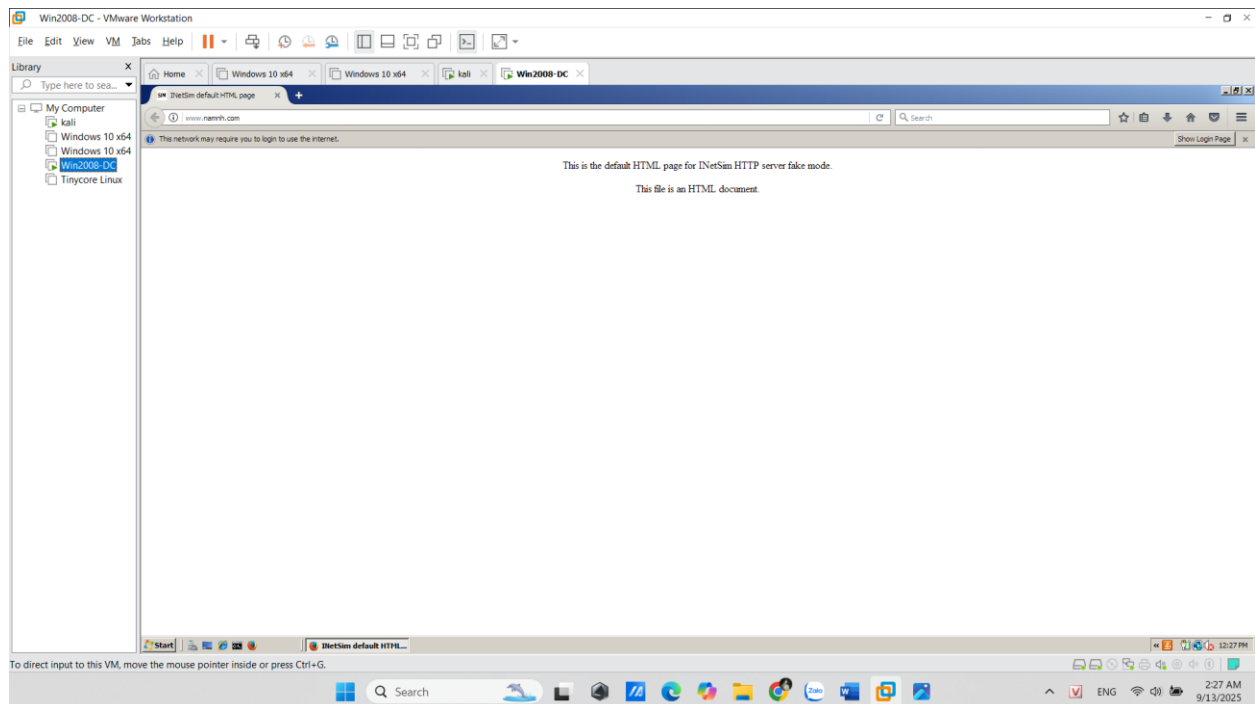


After this, it's working perfectly, no more error

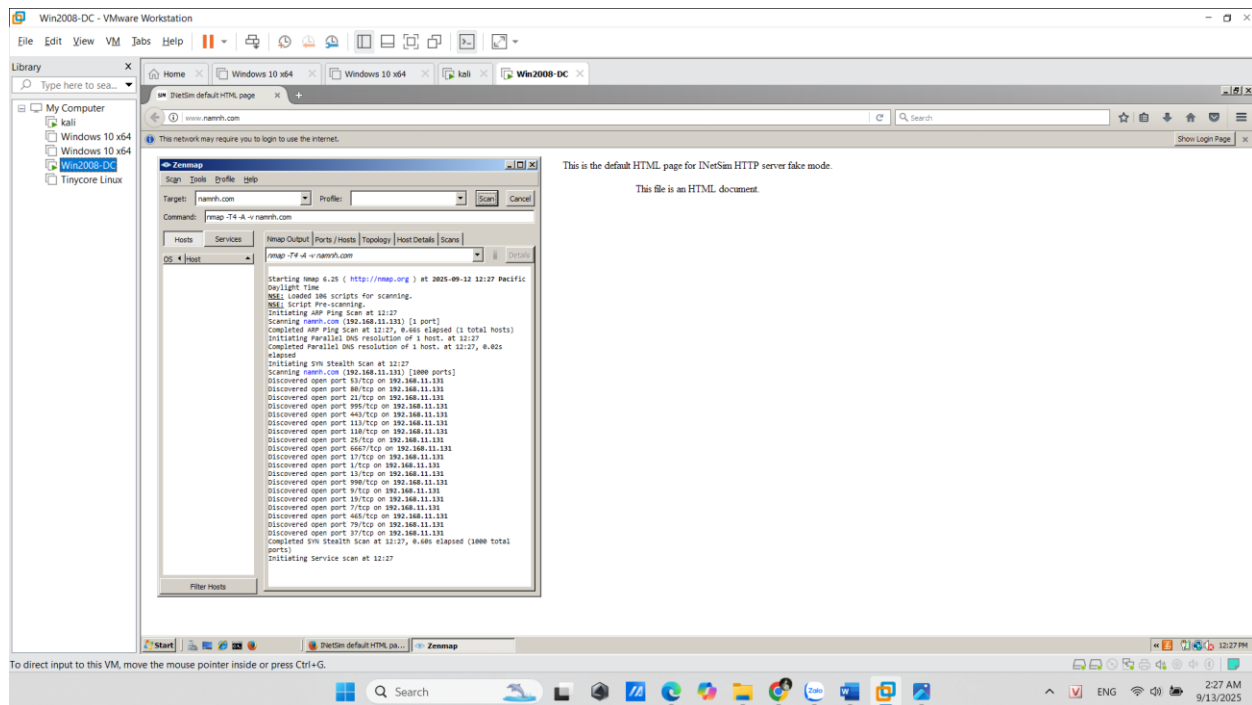
## 5. Network Configuration for Windows VM (DNS)



## Test HTTP on Windows (open YOURNAME.com)



## Scanning YOURNAME.com



## IV. Summary

- This lab focuses solely on configuring a self-hosted DNS responder on Kali Linux.
- All steps proceeded as expected except **Step 3**: on newer Kali releases you must downgrade/install Net::DNS (use Net-DNS-1.22) to resolve compatibility errors so INetSim runs correctly.

## Additional Question

### 1. PE (Portable Executable) Structure of Windows

- PE is a file format for executable files used in the Windows operating system, based on the COFF (Common Object File Format) file format

- A PE file is a data structure that contains the information needed for the operation system loader to load that executable file into memory and execute it
- **Structure:**
  - **DOS Header:** Every PE file begins with a 64-byte structure, which is what makes the PE file an MS-DOS executable
  - **DOS Stub:** It's a small MS-DOS 2.0, compatible executable that simply print the error message when the program is run in DOS mode
  - **NT Header:** consists of three main part:
    - **PE Signature:** A 4-byte signature that identifies the file as a PE file
    - **File Header:** A Standard COFF. It contains some information about the PE files
    - **Optional Header:** The most important header of the NT Header, because some file like object files do not have this header, however, this header is required for image files. Provide important information to the operating system loader
  - **Section Table:** located right after the Optional Header, is an array of Image Section Header, each section in the PE file has a section header; Each header contains information about the section it references
  - **Sections:** where the actual contents of the file are stored, including things like data and resources that the program uses, as well as the actual code of the program. There are several sections, each with its own purpose.

## 2. What is the differences between NAT, Host-only and Bridge in VMware Network Adapter?

- **NAT (Network Address Translation):**
  - Connect VM to Internet via host, more secure, not directly accessed
  - Used for personal lab
- **Bridged:**
  - VM as a device on LAN
  - Used when needing to access/be accessed from real network
- **Host-only**
  - Isolate VM from host or between VMs
  - Used for malware analysis without allowing access to the Internet