

Internet & Networking

Table of Contents

|  |  |
| --- | --- |
| Network Protocols - HTTP & HTTPS | 3 |
| DNS (Domain Name System) | 6 |
| Public and Private Networks | 10 |
| OWASP Vulnerabilities | 20 |

***1. A report on HTTP and HTTPS transactions.***

HTTP Get and Post requests

Get request example:  
[Get](https://web.postman.co/workspace/d9d8bbf8-63a3-4336-8e0f-88afdf924052/request/24008556-4ac16d98-d39f-47a9-8382-df0b7311db71) (created now)

Post request example:

[Post](https://web.postman.co/workspace/d9d8bbf8-63a3-4336-8e0f-88afdf924052/request/24008556-c4874bf3-67da-43ba-a5e7-a14f11942c35) (I used this in my project)

These requests were simpler, so I learnt more things from gpt

The main differences between GET and POST requests lie in how data is transmitted, the visibility of the data, and the intended purpose of the request. Here's a breakdown of the structural differences between GET and POST requests:

### GET Request:

* Data in URL:
  + Parameters and data are appended to the URL in the query string.
  + Visible in the browser's address bar.
* Data Transmission:
  + A limited amount of data can be sent.
  + Limited by the maximum length of a URL (varies across browsers but generally around 2,000 characters).
* Caching:
  + GET requests are cacheable by default since they are considered idempotent.
* Idempotence:
  + GET requests are considered idempotent, meaning making the same request multiple times has the same effect as making it once.
* Security Implications:
  + Not suitable for sensitive data as the data is visible in the URL.

### POST Request:

* Data in Body:
  + Parameters and data are included in the body of the request.
  + Not visible in the URL.
* Data Transmission:
  + Can handle larger amounts of data compared to GET.
  + No specific limit on data size is imposed by the HTTP protocol.
* Caching:
  + POST requests are not cacheable by default since they may have side effects.
* Idempotence:
  + POST requests are not considered idempotent, meaning making the same request multiple times may have different outcomes.
* Security Implications:
  + Suitable for sensitive data as the data is not visible in the URL.

2. Demonstration of Web Server implementation covering the above

I have built a *Django* server that runs on a publicly available hosting service (https) and on localhost as well (http).

**The server actually takes latitude and longitude of the ip of the device that request the service and return the live tentative temperature where ip is located.**

[feel the temperature](https://show-temp.onrender.com) (not ready yet)

[Github repo](https://github.com/nileshpratap/bootcamp/tree/main/showtemp)

[Postman request example](https://web.postman.co/workspace/My-Workspace~d9d8bbf8-63a3-4336-8e0f-88afdf924052/request/24008556-2fcb871a-526b-4fdc-99b2-83d61cc85e5f)

[Live Project (on HTTPS)](https://feeltheheat.vercel.app/)

**Most Important security differences between HTTP and HTTPS traffic:**

HTTP (Hypertext Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure) are both protocols used for communication over the internet, but they differ significantly in terms of security. Here are some of the key security differences between HTTP and HTTPS:

* Data Encryption:
  + HTTP: Sends data in plain text, which means that any data exchanged between the user's browser and the web server is visible and can be intercepted by attackers. This lack of encryption makes it vulnerable to eavesdropping and data theft.
  + HTTPS: Encrypts data using Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer (SSL). This encryption ensures that the data transmitted between the user and the server is secure and cannot be easily intercepted or tampered with.
* Data Integrity:
  + HTTP: Does not provide data integrity, meaning that data can be altered during transit without detection.
  + HTTPS: Ensures data integrity through encryption. Any tampering with the encrypted data during transit will result in the decryption failure, alerting both the client and server to potential malicious activity.
* Authentication:
  + HTTP: Does not provide any mechanism for server or client authentication. This lack of authentication makes it susceptible to man-in-the-middle attacks.
  + HTTPS: Uses digital certificates to authenticate the identity of the server. This helps users ensure they are connecting to the intended website and not a malicious impersonator.
* Secure Transactions:
  + HTTP: Transactions, such as form submissions and credit card details, are sent in plain text, making them susceptible to interception.
  + HTTPS: Ensures that sensitive transactions are secure by encrypting the data exchanged between the client and server, protecting it from eavesdropping and unauthorized access.
* URL Scheme:
  + HTTP: Uses the standard "http://" URL scheme.
  + HTTPS: Uses the "https://" URL scheme. Browsers typically display a padlock icon to indicate that the connection is secure.
* Trust Indicators:
  + HTTP: Lacks visual indicators in web browsers to denote a secure connection. This can lead to users being unaware of potential security risks.
  + HTTPS: Browsers display visual indicators such as a padlock icon, the word "Secure," or a green address bar to inform users that the connection is secure. This builds trust and encourages users to interact with the site confidently.
* SEO Impact:
  + HTTP: Search engines may rank HTTPS-enabled sites higher than HTTP-only sites as part of their ranking algorithms.
  + HTTPS: Provides a potential SEO boost due to the emphasis on secure connections.

***2. DNS***

### 1. DNS Query Tracing: Utilize tools to trace the path of a DNS query for a specified website.

Tool:

Dig is a network tool through which you can query DNS name servers. Our technical support personnel can use this information to diagnose issues resolving DNS records.

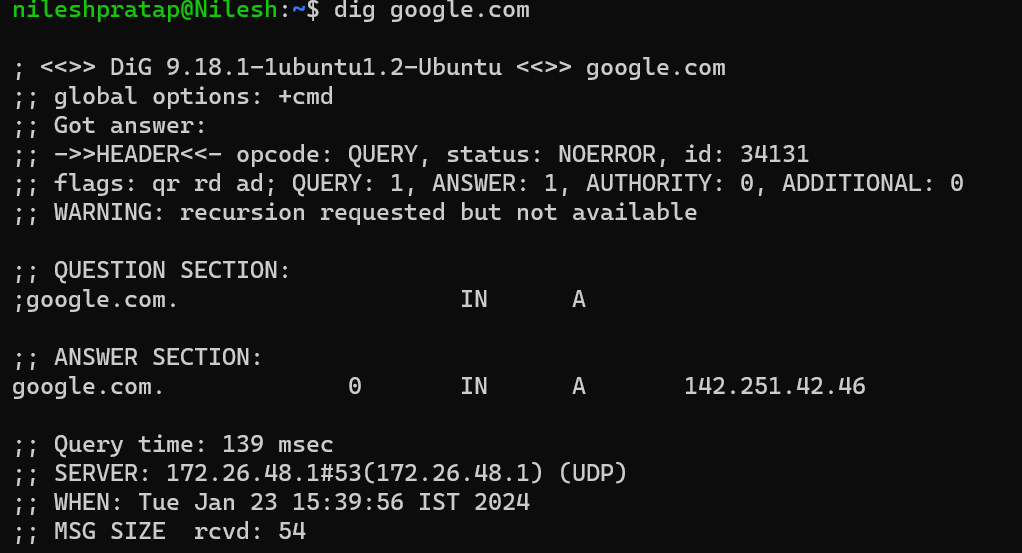
For ubuntu:

Command: dig example.com

Alternative command: nslookup example.com

For windows:

We need to download the tool ‘dig’ first.



Also, we can use  
dig +cmd +short google.com

+cmd is a global option that tells dig to use each next term as a command.

* qr: Indicates that this is a response.
* rd: Recursive query was requested.
* ad: Indicates that the answer is authentic (DNSSEC authenticated).

use case and differences between recursive and non-recursive DNS requests:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Recursive DNS Request** | **Non-Recursive DNS Request** |
| Use Case | - Common for end-users (clients) making DNS queries. | - Often used by authoritative DNS servers when responding to other DNS servers or resolvers. |

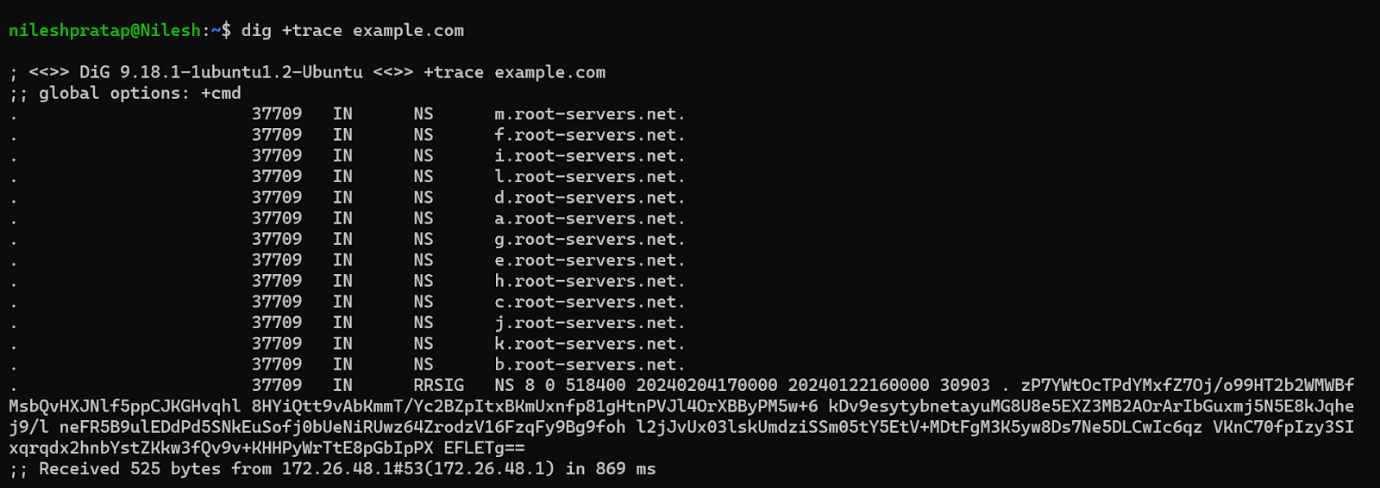
|  |  |  |
| --- | --- | --- |
| Query Behavior | - Submits the DNS query to one or more DNS servers recursively until a final answer is obtained. | - Responds with either the requested information or a referral to another DNS server (authoritative server). |

|  |  |  |
| --- | --- | --- |
| Caching | - Recursive servers often cache DNS records to improve response times for subsequent queries. | - Authoritative servers may cache records, but they are not primarily designed for caching. |

MSG SIZE  rcvd: 54 this shows the size of the response of dns request (represented in bytes)

To capture the route of dns query:

dig +trace example.com



;; initial servers are the root servers as we can see.

;; Received 525 bytes from 172.26.48.1#53(172.26.48.1) in 869 ms

Tells that the response was received from the DNS server with IP address 172.26.48.1, port 53, and it took 869 milliseconds to complete the query.

DNS lets users connect to websites using domain names instead of IP addresses. Learn how DNS works.

The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through [domain names](https://www.cloudflare.com/learning/dns/glossary/what-is-a-domain-name/), like nytimes.com or espn.com. Web browsers interact through [Internet Protocol (IP)](https://www.cloudflare.com/learning/network-layer/internet-protocol/) addresses. DNS translates domain names to [IP addresses](https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/) so browsers can load Internet resources.

## How does DNS work?

In order to understand the process behind the DNS resolution, it’s important to learn about the different hardware components a DNS query must pass between. For the web browser, the DNS lookup occurs "behind the scenes" and requires no interaction from the user’s computer apart from the initial request.

There are 4 DNS servers involved in loading a webpage:

1. [DNS recursor](https://www.cloudflare.com/learning/dns/dns-server-types/):

A recursive resolver (also known as a DNS recursor) is the first stop in a DNS query. The recursive resolver acts as a middleman between a client and a DNS nameserver. After receiving a DNS query from a web client, a recursive resolver will either respond with cached data, or send a request to a root nameserver, followed by another request to a TLD nameserver, and then one last request to an authoritative nameserver. After receiving a response from the authoritative nameserver containing the requested IP address, the recursive resolver then sends a response to the client.

During this process, the recursive resolver will cache information received from authoritative nameservers.

1. Root nameserver

A root server accepts a recursive resolver’s query which includes a domain name, and the root nameserver responds by directing the recursive resolver to a TLD nameserver, based on the extension of that domain (.com, .net, .org, etc.)

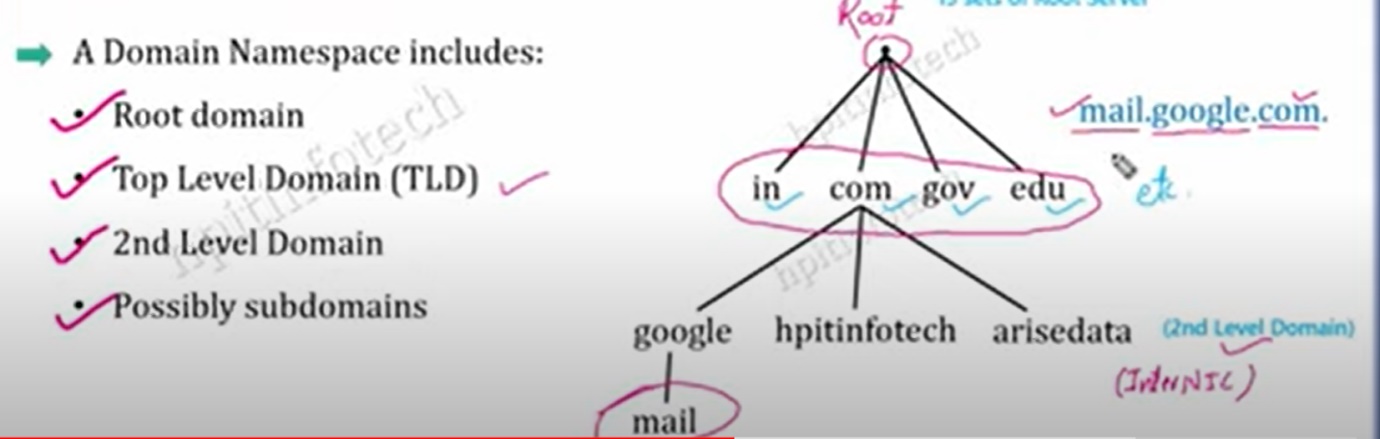
there are 13 root nameservers, that does not mean that there are only 13 machines in the root nameserver system. There are 13 types of root nameservers, but there are multiple copies of each one

1. [TLD nameserver](https://www.cloudflare.com/learning/dns/dns-server-types/) (Top Level Domain server)

A TLD nameserver maintains information for all the domain names that share a common domain extension, such as .com, .net, or whatever comes after the last dot in a URL. For example, a .com TLD nameserver contains information for every website that ends in ‘.com’.

1. [Authoritative nameserver](https://www.cloudflare.com/learning/dns/dns-server-types/)

The authoritative nameserver contains information specific to the domain name it serves (e.g. google.com) and it can provide a recursive resolver with the IP address of that server found in the [DNS A record](https://www.cloudflare.com/learning/dns/dns-records/dns-a-record/),



**Demonstration of DNS server setup.**

[resource to DNS setup using nginx](https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-server-blocks-virtual-hosts-on-ubuntu-16-04)

**Result:**

Nginx first server block

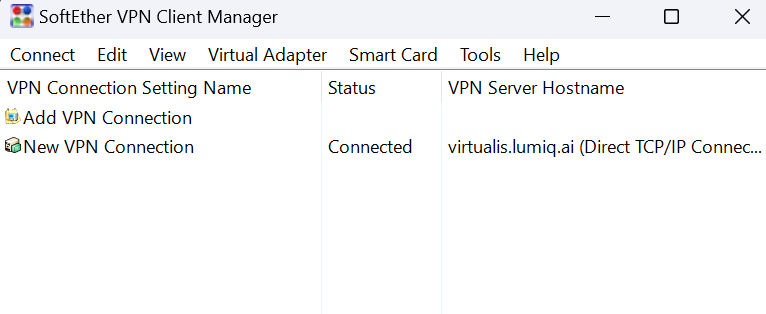
**3.Public and Private Networks**

**Objective: Understand and demonstrate the functional differences between public and private networks, including network address translation (NAT) and port forwarding.**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Public Network** | **Private Network** |
| Ownership | Owned and operated by a third-party organization or government. | Owned and operated by a specific individual, organization, or enterprise. |
| Accessibility | Accessible by the general public or authorized users. | Restricted access, typically limited to a specific organization or group of users. |
| Security | Higher security concerns due to exposure to the public. Requires robust security measures. | Lower security concerns since access is restricted, but still needs adequate security. |
| IP Addressing | Uses public IP addresses that are globally unique. | Uses private IP addresses within the network, and a single public IP for external communication (NAT). |
| Network Address Translation (NAT) | Routinely uses NAT to map multiple private IP addresses to a single public IP address for external communication. | NAT is used to enable multiple devices within the private network to share a single public IP address for external communication. |
| Port Forwarding | Often requires port forwarding for specific services to be accessible from the outside. | Port forwarding may be used, but it's generally less common due to higher security concerns. |
| Costs | May involve subscription fees or usage charges. | Controlled by the owner, with potential costs associated with infrastructure and maintenance. |
| Performance | Performance may vary depending on network congestion and usage. | Generally, better performance since resources are dedicated and not shared with the public. |
| Scalability | More scalable due to resources provided by service providers. | Scalability may be limited by the organization's infrastructure and budget. |
| Visibility | Publicly visible and may be discoverable through search engines. | Less visibility, and internal resources are not accessible to the public without proper authorization. |

1. Private Network Access: Gain access to a private network.

Gained access to Lumiq’s private network through the SoftEther VPN client manager



2. Local Web Server Connection: Connect to a web server hosted within the Lumiq environment.

1. First connected to lumiq’s private network
2. Then sent a http request to the redmine server hosted on lumiq’s private network
3. Entered credentials and established a connection with the redmine server.



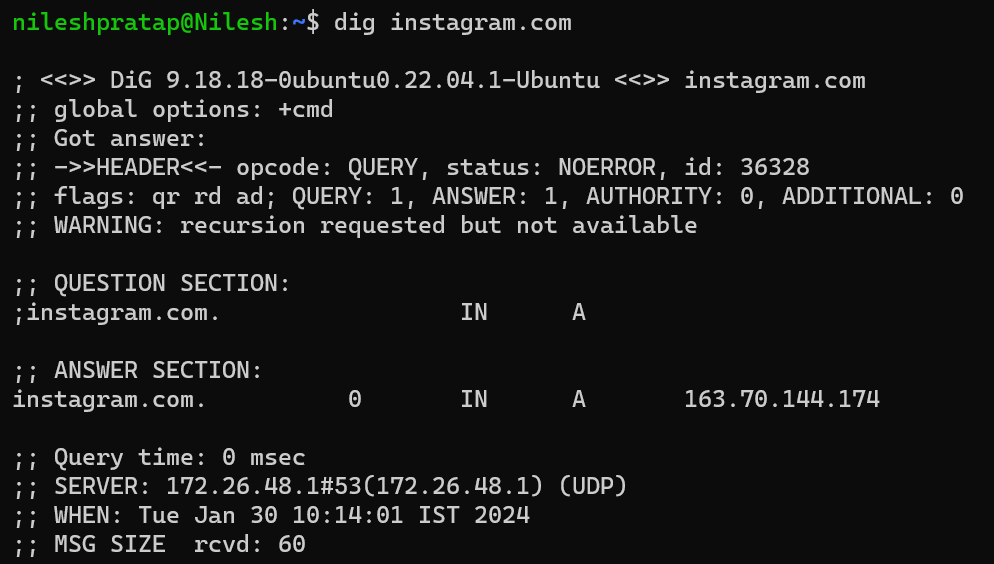
3. Firewall Configuration: Implement firewall rules to manage traffic between the Networks.

[firewall configuration](https://www.fortinet.com/resources/cyberglossary/firewall-configuration)

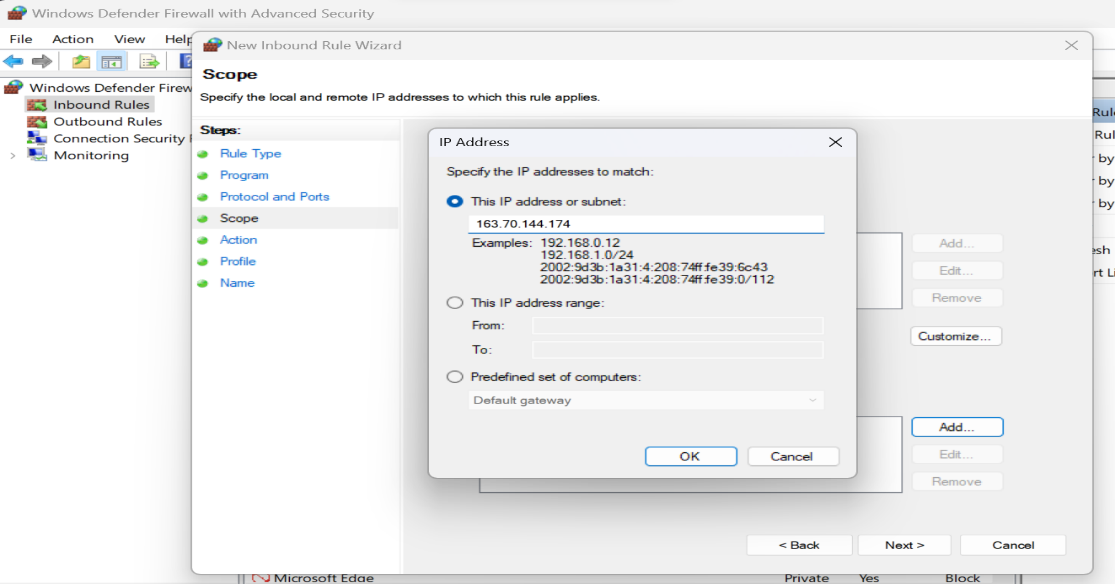
[demo in windows defender firewall](https://www.youtube.com/watch?v=we2pn7w-3IU)

→ Process to block the Instagram

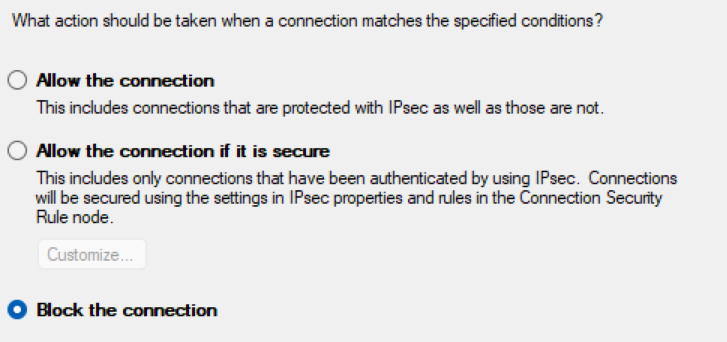
1. Determine IP or IP range that we want to block (here instagram.com)



1. Add this IP in the Windows Defender Firewall Block connection feature.



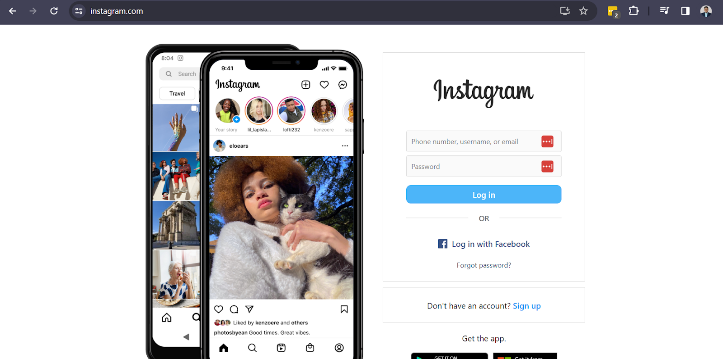
1. Choose to block



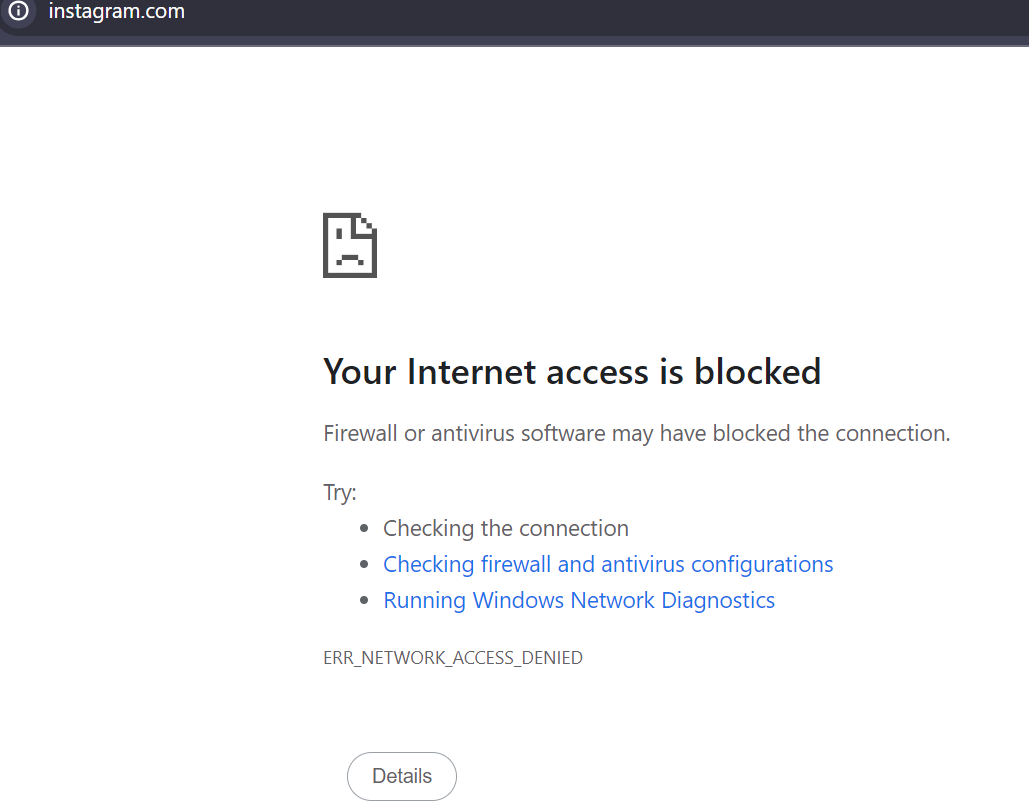
1. Save the setting and give it a name.

Now

Before



After



4. SSH Connection: Make an SSH connection from your laptop to your peer laptop. Do it with password and passwordless mechanisms.

Establishing a connection with a VM by following a **password** mechanism

1. Create a network adapter in VM that allows all traffic and its type to be bridged adapter as the host(destination) is our main local system. Save and enable the adapter
2. Launch the VM
3. Change firewall settings to allow traffic from port 22 and assign it to ssh if not present already.

First check to make sure sshd is installed (using debian examples)

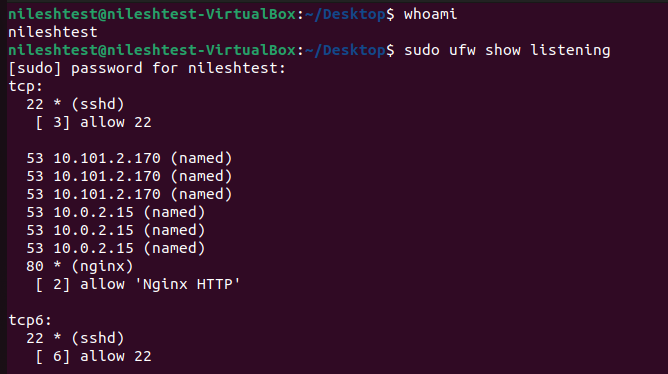
            sudo apt-get install openssh-server And if so, is it running:

ps -ef | grep sshd

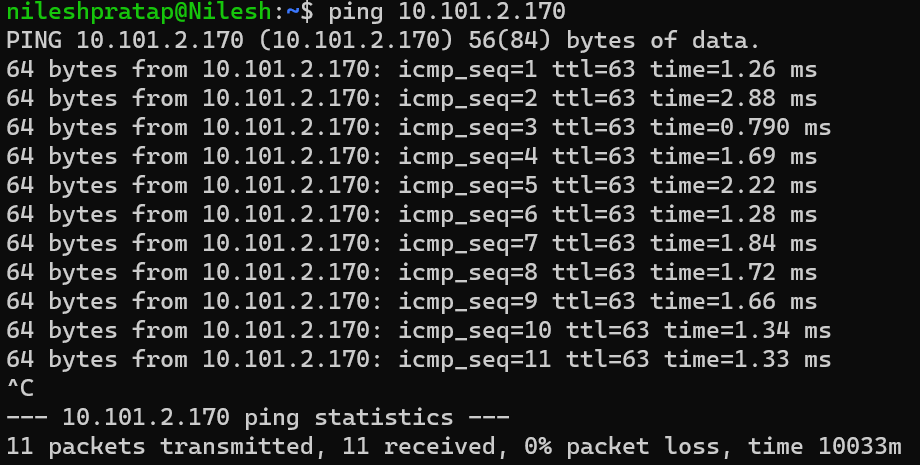
then check to see if it is listening to port 22

If ufw shows it as closed then run (again a debian/ubuntu example)

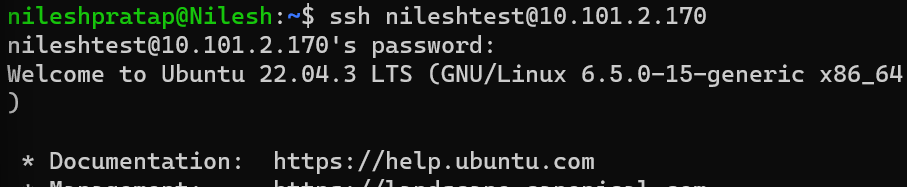
sudo ufw allow 22



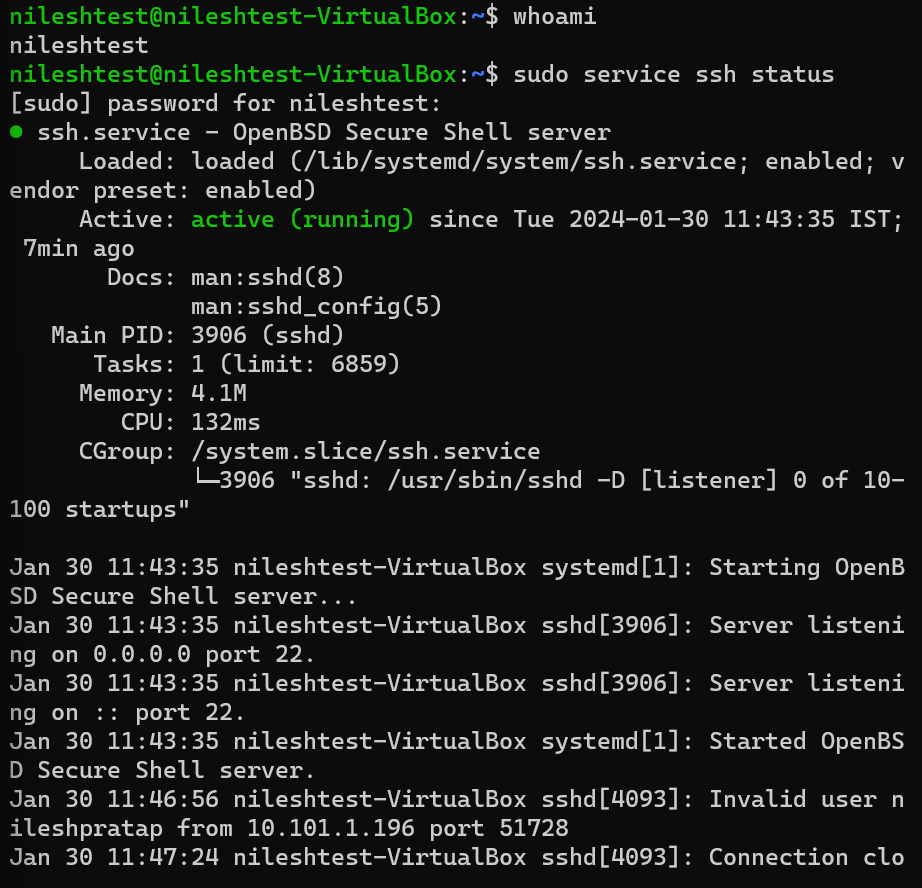
1. First ping the ip of this vm in the host system:



1. Apply SSH now with the username@ip of the VM to access and we will have to enter the password.



1. Checking the logged-in user and verifying the SSH connection:



Establishing a connection with a **passwordless** mechanism

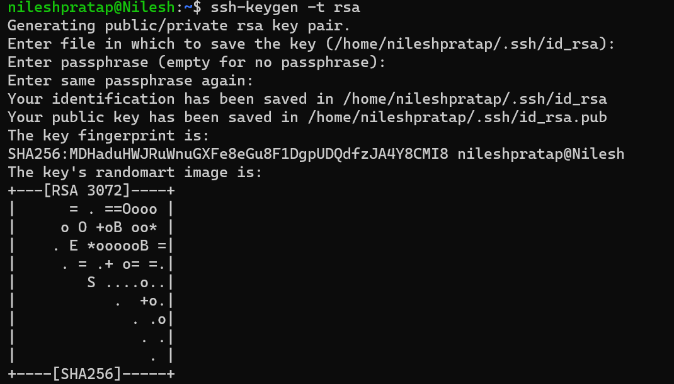
[Followed process](https://www.strongdm.com/blog/ssh-passwordless-login#:~:text=At%20your%20command%20line%20prompt,enter%20it%20on%20each%20use.)

Key idea:

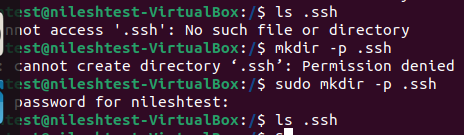
SSH passwordless login is an SSH authentication method that employs a pair of public and private keys for asymmetric encryption. The public key resides on the server, and only a client that presents the private key can connect.

→ doing these on the host machine.:

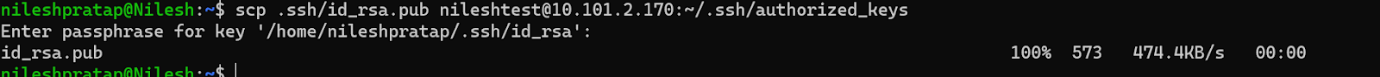
1. Key generation on host.



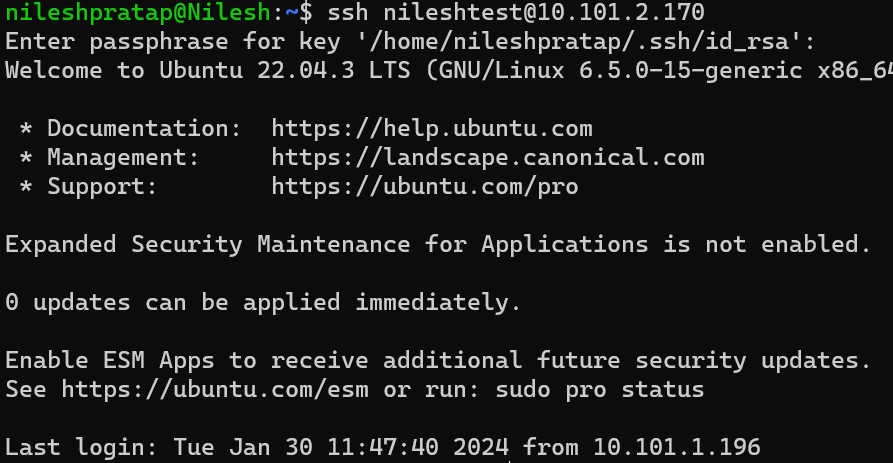
1. Creating a ssh directory on a server.



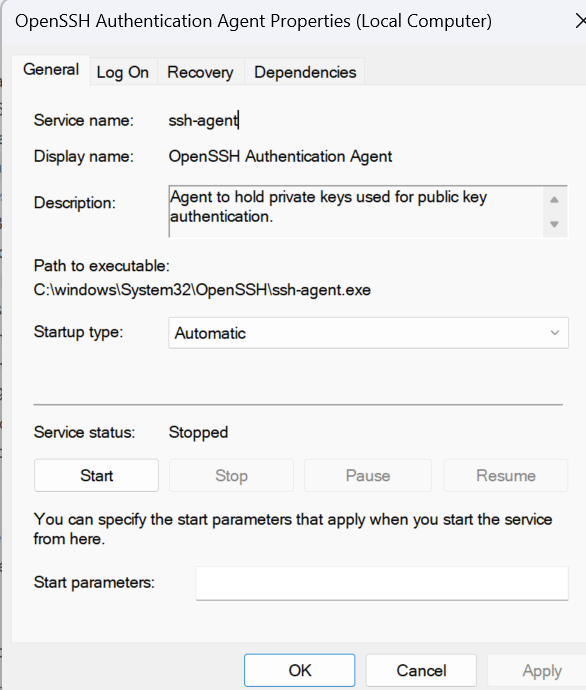
1. Uploading the key to the server/vm in this case



1. Login and remember the paraphrase

Login:  


1. To remember this connection in order to avoid entering the paraphrase each time:



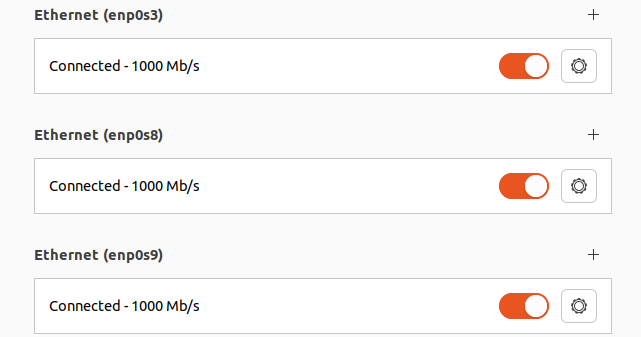
5. NAT and Port Forwarding: Exhibit how private network devices communicate with the internet using NAT and port forwarding techniques.

port forwarding is used to allow the remote computers to connect to a specific service or computer on a Local Area Network.

Why do we need it?

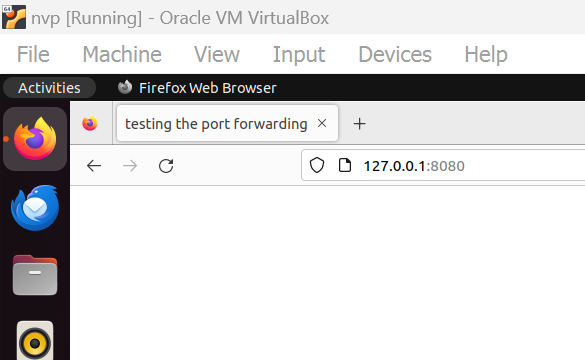
Because the firewall or other security mechanisms might block the external traffic even though it is legitimate and server our purpose.

Created a dedicated adapter on a VM to do port forwarding



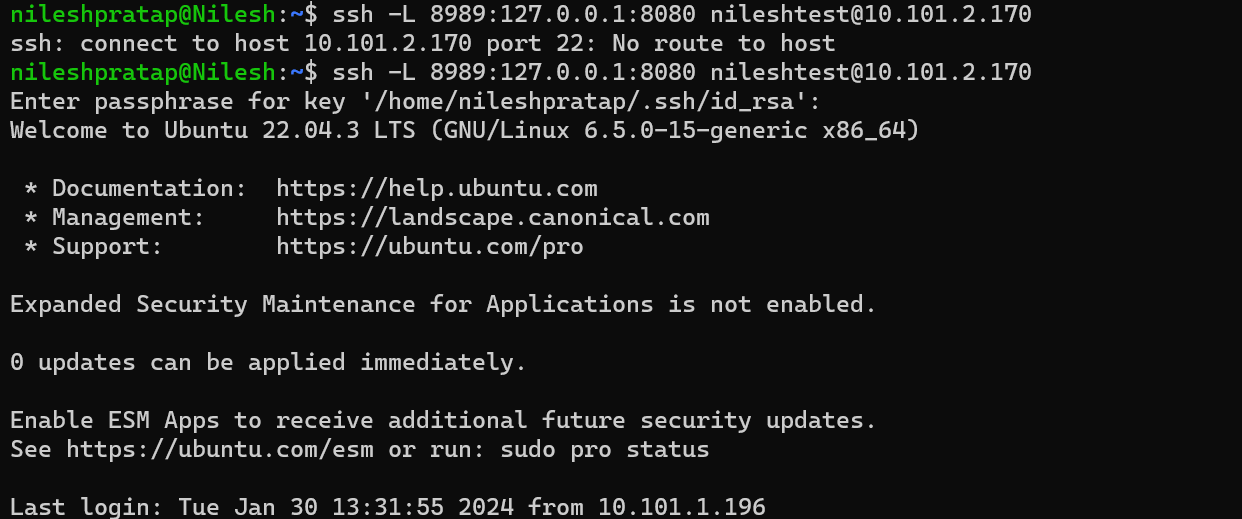
6. Port Forwarding Utilization: Access a website from a different laptop via port forwarding.

1. Hosted a html file on the VM using npm on port 8080

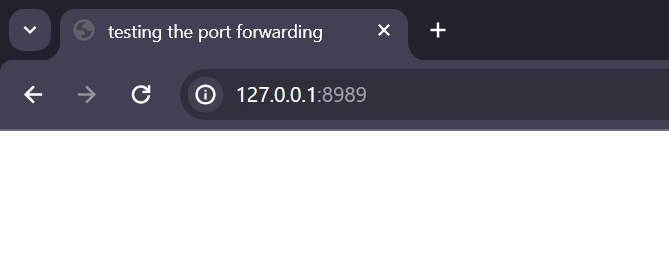


1. Then established a SSH connection with this VM from a local machine with port forwarding details (the webpage on the VM can be accessed on local machine’s port number 8989 with this command)

Command: ***ssh -L 8989:127.0.0.1:8080 nileshtest@10.101.2.170***



1. Tested on the browser of local machine.



1. ***OWASP Vulnerabilities***

This section consists of explanations of all the OWASP (Open Web Application Security Project) top 10 vulnerabilities. Solutions to avoid these are also mentioned.



[**Broken Access Control**](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)

- 94% of applications were tested for some form of broken access control.

- The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.

Broken access controls allow attackers to bypass authorization and perform tasks like they were privileged users such as administrators.

Impact: data breach, threat to app’s DB

[**Cryptographic Failures**](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

- cryptography failures often lead to sensitive data exposure or system compromise.

- If web applications don’t protect sensitive data such as financial information and passwords, attackers can gain access to that data

- Data exposure risk can be minimized by encrypting all sensitive data and disabling the caching\* of sensitive information.

[**Injection**](https://owasp.org/Top10/A03_2021-Injection/)

- second most occurrences in applications

- Injection attacks happen when untrusted data is sent to a code interpreter through a data submission to a web application.

- An attacker could enter SQL database code into a form that expects a plaintext username. If that form input is not properly secured, this would result in that SQL code being executed. This is known as an SQL injection attack.

- Injection attacks can be prevented by validating and/or sanitizing user-submitted data.

[**Insecure Design**](https://owasp.org/Top10/A04_2021-Insecure_Design/)

- risks related to design flaws

- Example: A movie theater chain that allows group booking discounts requires a deposit for groups of more than 15 people. Attackers threat model this flow to see if they can book hundreds of seats across various theaters in the chain, thereby causing thousands of dollars in lost income.

solution: improve the design aspects of the web application.

[**Security Misconfiguration**](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)

- With more shifts into highly configurable software, 90% of applications were tested for some form of misconfiguration

- is often the result of using default configurations or displaying excessively verbose errors.

solution:

- correctly configure the web app

- show more general errors rather than showing the app's vulnerabilities to the end user through the errors.

[**Vulnerable and Outdated Components**](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/)

- Example: Due to the volume of components used in the development, a development team might not know or understand all the components used in their application, and some of those components might be out-of-date and therefore vulnerable to attack.

- solution:

properly maintain the external components and use third-party tools to detect outdated and insecure components in an application

[**Identification and Authentication Failures**](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)

- functions related to authentication and session management, when implemented incorrectly, allow attackers to compromise passwords

- Solution: Multifactor authentication can help reduce the risk of compromised accounts

[**Software and Data Integrity Failures**](https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/)

- focuses on software updates, critical data, and CI/CD pipelines used without verifying integrity

- insecure deserialization is a deserialization flaw that allows an attacker to execute code in the system remotely.

- Example: An application deserializes attacker-supplied hostile objects, opening itself to vulnerability.

- solution: ensure data integrity during the testing phase

[**Security Logging and Monitoring Failures**](https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/)

- logging crucial events of the web applications is important.

- if these events are not logged, it makes our app more vulnerable in the sense that bad events are not even logged to be handled and it becomes difficult to understand the situation.

- solution: generate logs for all important events of the web application.

[**Server-Side Request Forgery**](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/)

- can happen when a web application fetches a remote resource without validating the user-supplied URL

- This allows an attacker to make the application send a crafted request to an unexpected destination, even when a firewall, VPN, or additional network access control list protects the system.

- these attacks increase the complexity of architectures.

- solution: continuously track, monitor, detect, and control such requests made by the server.

Resources:

[official site](https://owasp.org/www-project-top-ten/) [Blog1](https://www.cloudflare.com/learning/security/threats/owasp-top-10/) [Blog2](https://www.synopsys.com/glossary/what-is-owasp-top-10.html#A)