# **Data Scraping Techniques: Detailed Explanation**

Data scraping, also known as web scraping, is the process of extracting data from websites and other online sources. This data can be used for various purposes such as market research, competitive analysis, academic research, and more. The goal of data scraping is to automate the extraction process, making it efficient and scalable.

# Common Data Scraping Techniques

1. HTML Parsing
2. DOM Parsing
3. XPath
4. CSS Selectors
5. API Interfacing
6. Headless Browsers
7. Data Extraction Tools

**1. HTML Parsing**

HTML parsing involves reading the raw HTML of a webpage and extracting the required data using string manipulation and regular expressions. This method is straightforward but can be error-prone and fragile, especially if the HTML structure changes.

**Example:**

//Using python

import re

import requests

url = 'http://example.com'

response = requests.get(url)

html = response.text

# Extract all headings (e.g., <h1>, <h2>)

headings = re.findall(r'<h[1-6]>.\*?</h[1-6]>', html)

print(headings)

**2. DOM Parsing**

DOM parsing involves loading the webpage into a Document Object Model (DOM) and using DOM traversal methods to extract data. This technique leverages the structured nature of the DOM, making it more reliable than raw HTML parsing.

**Example:**

//Using python

from bs4 import BeautifulSoup

import requests

url = 'http://example.com'

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

# Extract all headings (e.g., <h1>, <h2>)

headings = soup.find\_all(['h1', 'h2'])

for heading in headings:

print(heading.text)

**3. XPath**

XPath is a powerful language for navigating and querying XML documents, including HTML. It allows for precise extraction of data based on the document's hierarchical structure.

**Example:**

//Using python

from lxml import html

import requests

url = 'http://example.com'

response = requests.get(url)

tree = html.fromstring(response.content)

# Extract all headings (e.g., <h1>, <h2>)

headings = tree.xpath('//h1 | //h2')

for heading in headings:

print(heading.text\_content())

**4. CSS Selectors**

CSS selectors provide a way to query and extract elements from a webpage based on CSS rules. This method is intuitive for those familiar with CSS and is supported by many scraping libraries.

**Example:**

//Using python

from bs4 import BeautifulSoup

import requests

url = 'http://example.com'

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

# Extract all headings (e.g., <h1>, <h2>)

headings = soup.select('h1, h2')

for heading in headings:

print(heading.text)

**5. API Interfacing**

Many websites provide APIs that allow for structured access to their data. Using APIs is often the most reliable and ethical method for data scraping, as it adheres to the provider's intended usage.

**Example:**

//Using python

import requests

url = 'http://api.example.com/data'

response = requests.get(url)

data = response.json()

# Print retrieved data

print(data)

**6. Headless Browsers**

Headless browsers are web browsers without a graphical user interface. They can be automated to interact with web pages, fill out forms, and extract data. Tools like Selenium and Puppeteer are commonly used for this purpose.

**Example with Selenium:**

//Using python

Copy code

from selenium import webdriver

url = 'http://example.com'

driver = webdriver.Chrome()

driver.get(url)

# Extract all headings (e.g., <h1>, <h2>)

headings = driver.find\_elements\_by\_xpath('//h1 | //h2')

for heading in headings:

print(heading.text)

driver.quit()

**7. Data Extraction Tools**

There are various tools designed specifically for web scraping, such as Scrapy, Octoparse, and ParseHub. These tools offer high-level functionalities to simplify the scraping process.

**Example with Scrapy:**

//Using python

import scrapy

class ExampleSpider(scrapy.Spider):

name = 'example'

start\_urls = ['http://example.com']

def parse(self, response):

for heading in response.css('h1, h2'):

yield {'text': heading.get()}

**URL:**

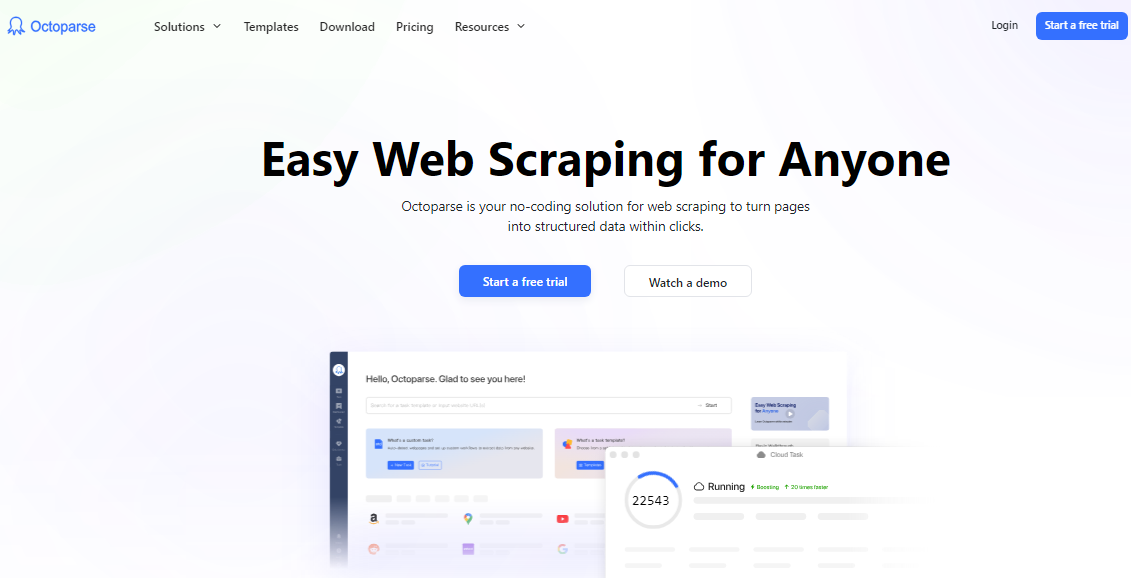
**- https://github.com/scrapy/scrapy.**

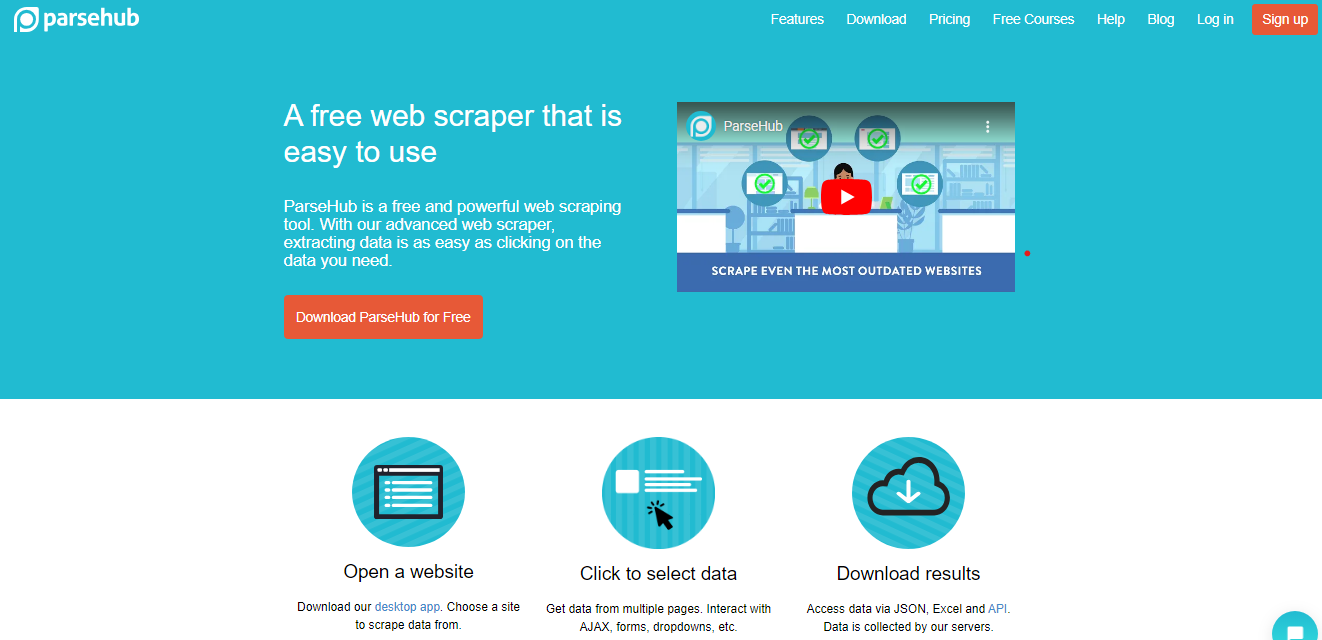
**- https://scrapy.org/.**

**- https://docs.scrapy.org/en/latest/**

**Ethical Considerations**

* **Respect Robots.txt**: Check and respect the website's robots.txt file, which indicates allowed and disallowed scraping paths.
* **Avoid Overloading Servers**: Implement rate limiting to avoid excessive requests that could overwhelm the server.
* **Legal Compliance**: Ensure scraping activities comply with legal regulations and the website's terms of service.
* **Privacy Considerations**: Avoid collecting personal data without consent and adhere to privacy laws such as GDPR.





# **Google Dorks and Advanced Search Queries**

**What Are Google Dorks?**

Google Dorks, also known as Google hacking, refer to using advanced search queries to find information that is not easily accessible via standard search methods. These queries take advantage of Google's powerful indexing capabilities to uncover hidden data, vulnerabilities, and files on web servers.

**Advanced Google Search Queries**

1. Basic Operators
2. Filetype Operator
3. Intitle Operator
4. Inurl Operator
5. Site Operator
6. Cache Operator
7. Link Operator
8. Related Operator
9. Intext Operator
10. Allintitle, Allinurl, Allintext Operators

**1. Basic Operators**

* **AND**: Finds results that include both terms.

example AND test

* **OR**: Finds results that include either term.

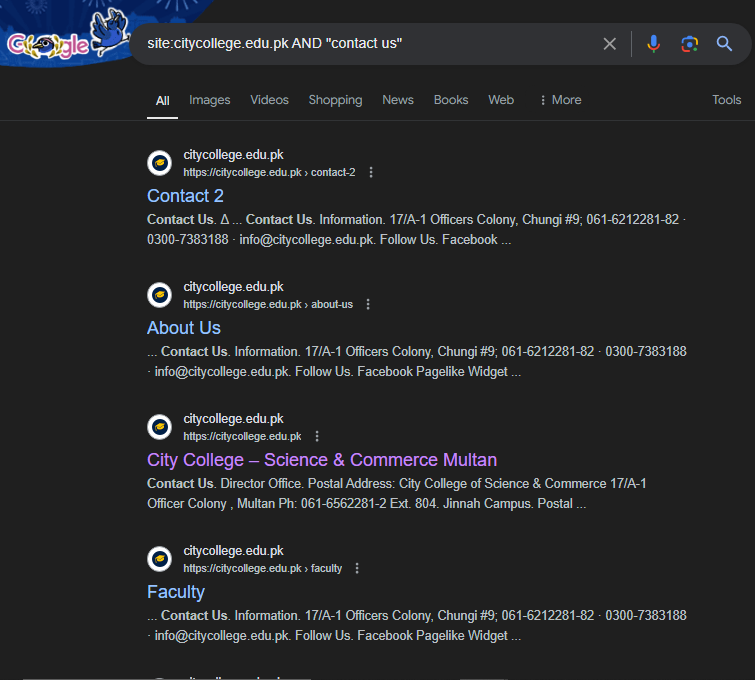
example OR test

* **- (Minus)**: Excludes results containing the term.

example -test

**Example:**

site:citycollege.edu.pk AND "contact us"



**2. Filetype Operator**

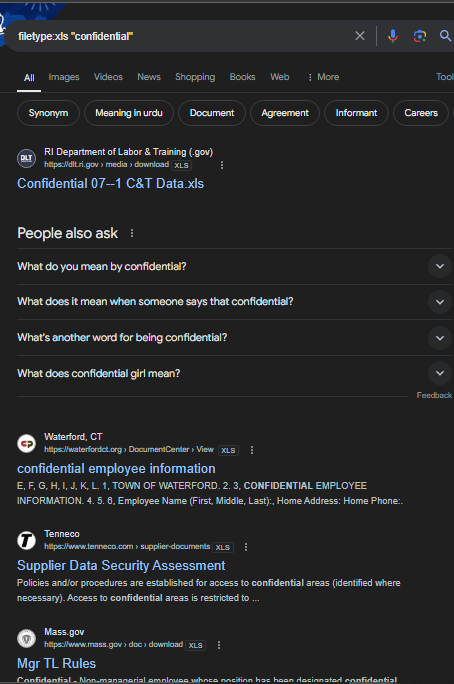
Searches for specific file types.

* **filetype:**: Finds files of a particular type.

filetype:pdf "annual report"

**Example:**

filetype:xls "confidential"



**3. Intitle Operator**

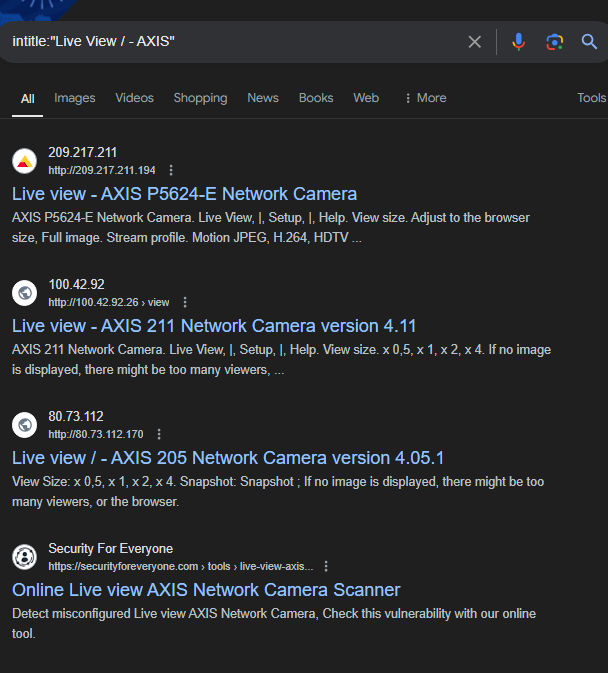
Searches for pages with a specific word in the title.

* **intitle:**: Finds pages with the term in the title.

intitle:"index of"

**Example:**

**intitle:"Live View / - AXIS"**



**4. Inurl Operator**

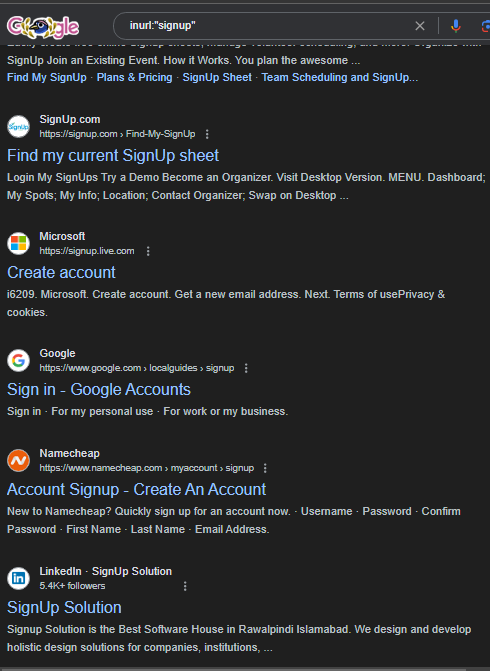
Searches for pages with a specific word in the URL.

* **inurl:**: Finds pages with the term in the URL.

inurl:admin

**Example:**

inurl:"signup"



**5. Site Operator**

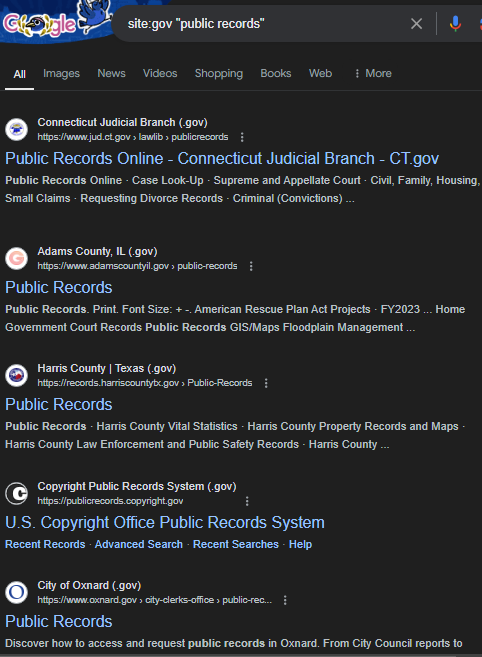
Limits search results to a specific site or domain.

* **site:**: Searches within a specific site.

site:example.com

**Example:**

site:gov "public records"



**6. Cache Operator**

Shows the cached version of a webpage.

* **cache:**: Displays Google's cached version of a page.

cache:example.com

**Example:**

cache:example.com

**7. Link Operator**

Finds pages that link to a specific URL.

* **link:**: Finds pages linking to the specified URL.

link:example.com

**Example:**

link:example.com

**8. Related Operator**

Finds sites related to a specified URL.

* **related:**: Finds sites similar to the specified URL.

related:example.com

**Example:**

related:example.com

**9. Intext Operator**

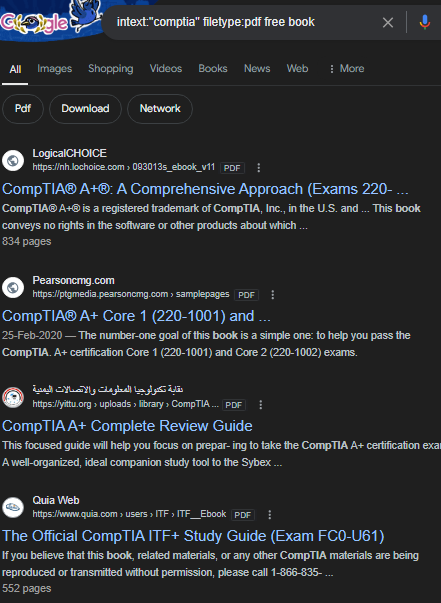
Searches for pages with a specific word in the text.

* **intext:**: Finds pages with the term in the text.

intext:"security policy"

**Example:**

intext:"vulnerabilities"



**10. Allintitle, Allinurl, Allintext Operators**

Searches for multiple terms in specific parts of the webpage.

* **allintitle:**: Finds pages with all the specified terms in the title.

allintitle:”login” “password”

* **allinurl:**: Finds pages with all the specified terms in the URL.

allinurl:admin login

* **allintext:**: Finds pages with all the specified terms in the text.

allintext:username password

**Examples:**

allintitle:"username" "password"

allinurl:admin "dashboard"

allintext:"database" "password"

**Practical Examples of Google Dorks**

1. **Finding Login Pages**

inurl:login

1. **Discovering Sensitive Files**

filetype:docx "confidential"

1. **Locating Publicly Accessible Cameras**

intitle:"Live View / - AXIS"

1. **Searching for Exposed Email Lists**

filetype:xls inurl:"email list"

1. **Finding Vulnerable Web Pages**

intitle:"index of" inurl:admin

1. **Exploring Site-Specific Directories**

site:example.com inurl:admin

1. **Accessing Cached Versions of Websites**

cache:example.com

**Conclusion**

Google Dorks provide a powerful way to uncover hidden information on the web. By leveraging advanced search queries, users can find sensitive data, locate specific files, and identify vulnerabilities. However, it's crucial to use these techniques ethically and within legal boundaries to avoid unauthorized access and potential legal repercussions.

# **Google Hacking Database (GHDB)**

#### What is GHDB?

The Google Hacking Database (GHDB) is a collection of search queries (commonly known as "Google Dorks") that can be used to find sensitive information and vulnerabilities in websites through the use of advanced Google search operators. It was created by Johnny Long in 2002 and is maintained by Offensive Security.

#### Purpose of GHDB

* **Security Research**: Used by security researchers and penetration testers to identify potential security weaknesses in websites.
* **Vulnerability Assessment**: Helps in discovering exposed sensitive data, misconfigurations, and other security issues.
* **Information Gathering**: Assists in gathering detailed information about a target during the reconnaissance phase of penetration testing.

#### Types of Queries in GHDB

* **Sensitive Directories**: Queries that find directories that should not be publicly accessible.

intitle:"index of" site:example.com

* **Error Messages**: Queries that locate error messages revealing sensitive information.

intext:"phpinfo()" "published by the PHP Group"

* **Configuration Files**: Queries that find configuration files containing sensitive data.

filetype:cfg inurl:settings

* **Sensitive Data**: Queries that uncover files containing sensitive data like passwords and credit card numbers.

filetype:xls inurl:"email list"

#### Example of a GHDB Entry

**Query**:

intitle:"index of" "backup"

**Purpose**: Finds directory listings of backup files, which can contain sensitive data and should not be publicly accessible.

# **Dorkbot**

#### What is Dorkbot?

Dorkbot is an automated tool designed to perform Google Dorks-based reconnaissance. It is a Python script that utilizes the power of the Google Hacking Database to find vulnerable targets and sensitive information on the web.

#### Purpose of Dorkbot

* **Automation**: Automates the process of searching for vulnerabilities using Google Dorks.
* **Efficiency**: Saves time by running multiple Google Dorks queries simultaneously.
* **Comprehensive Scanning**: Provides detailed reports of findings, helping security professionals assess the security posture of targets.

#### Features of Dorkbot

* **Modular Design**: Allows users to add custom modules and queries.
* **Multiple Search Engines**: Supports multiple search engines in addition to Google.
* **Output Options**: Provides various output formats for reports, including HTML and CSV.
* **Customization**: Users can customize queries and target specific types of information.

#### Example Usage of Dorkbot

1. **Install Dorkbot**:

git clone https://github.com/BlueTeamLabs/dorkbot

cd dorkbot

pip install -r requirements.txt

1. **Run Dorkbot**:

python dorkbot.py -q "intitle:'index of' 'backup'" -e google

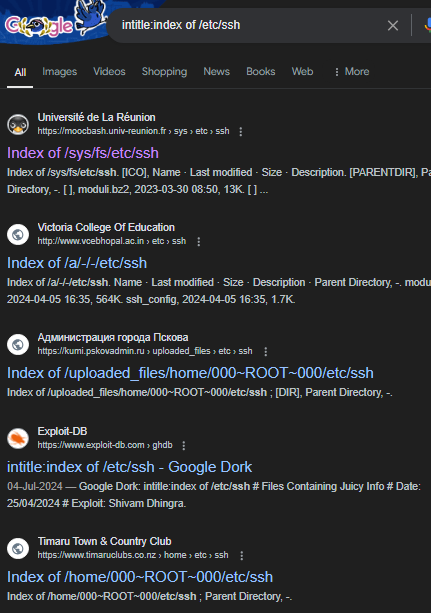
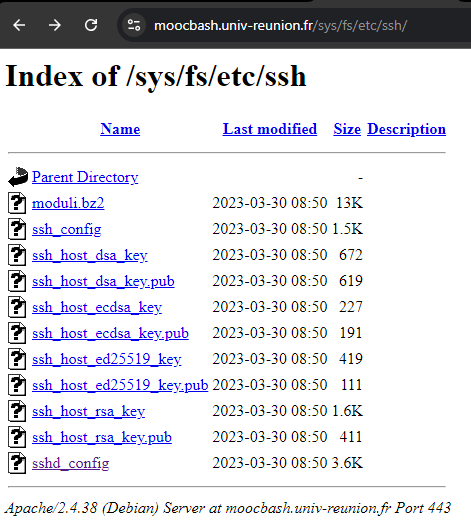
**Explanation**: This command runs Dorkbot with a Google Dork query that searches for directory listings of backup files.

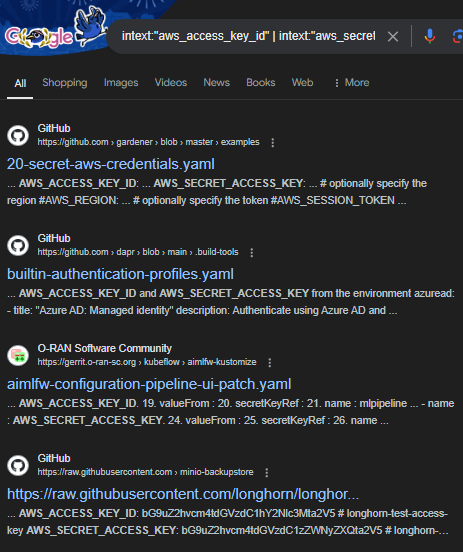
### Conclusion

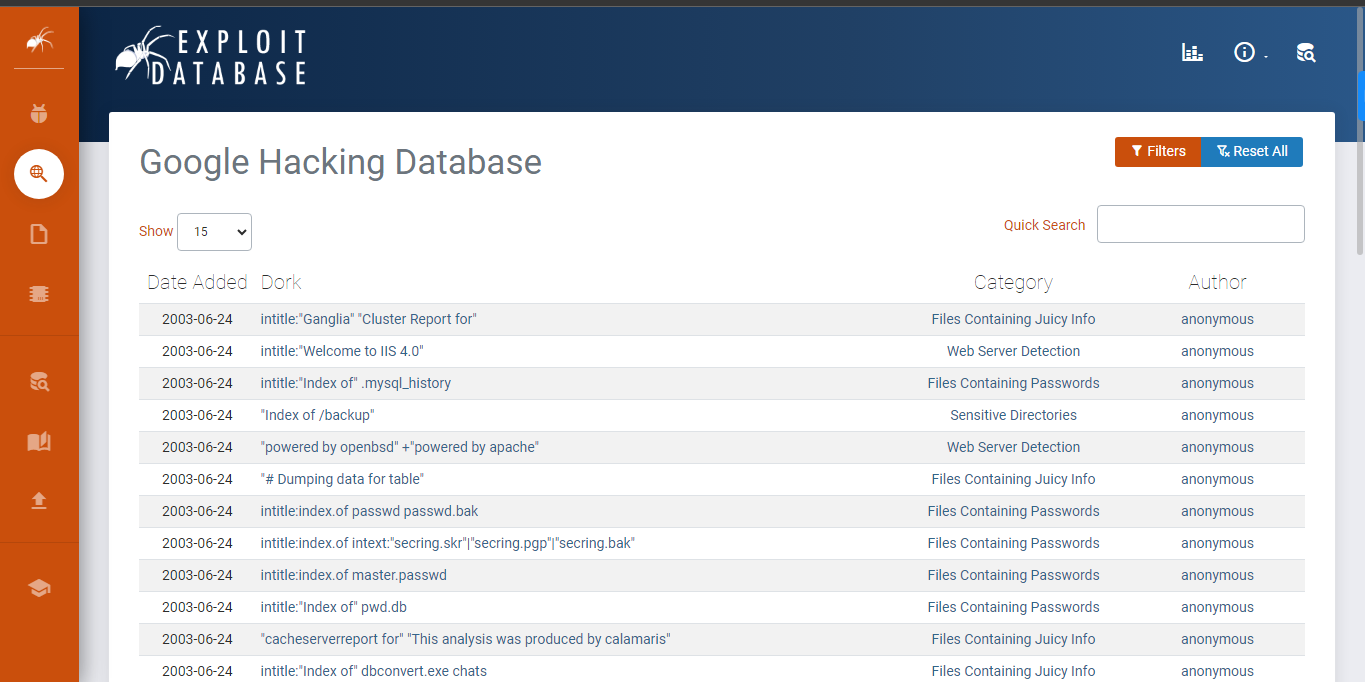
The Google Hacking Database (GHDB) and tools like Dorkbot are valuable resources for security professionals, enabling them to uncover vulnerabilities and sensitive information through advanced search techniques. While these tools are powerful, it's crucial to use them ethically and within the bounds of the law, obtaining proper authorization before testing any systems or networks that do not belong to you.

**URL:**

* <https://www.exploit-db.com/google-hacking-database>
* <https://github.com/chr3st5an/Google-Dorking>
* <https://security.utexas.edu/dorkbot>







# **Geolocation and IP Tracing**

**What is Geolocation?**

Geolocation refers to the process of identifying the physical location of a device (e.g., computer, smartphone) connected to the internet. This is typically done using the device's IP address, GPS data, Wi-Fi positioning, or cell tower triangulation.

#### What is IP Tracing?

IP tracing involves tracking the origin of an IP address to determine its geographical location, ISP (Internet Service Provider), and other related information. IP tracing tools and techniques can provide insights into the approximate location of the device using the IP address.

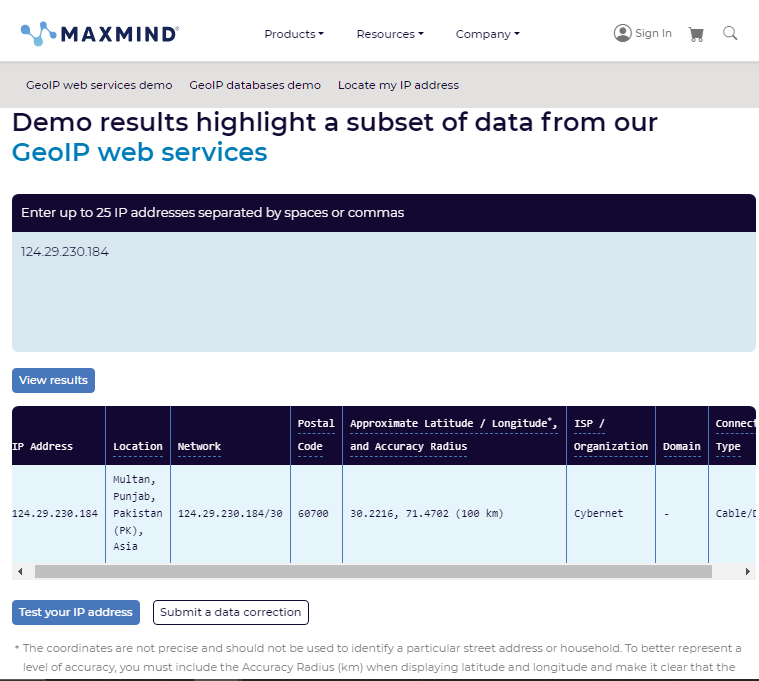
### Applications of Geolocation and IP Tracing

1. **Cybersecurity**: Identifying the location of potential cyber threats and attacks.
2. **Law Enforcement**: Locating individuals involved in illegal activities.
3. **Marketing**: Delivering region-specific content and advertisements.
4. **Compliance**: Ensuring compliance with regional data protection laws.
5. **Network Management**: Diagnosing network issues and managing traffic.

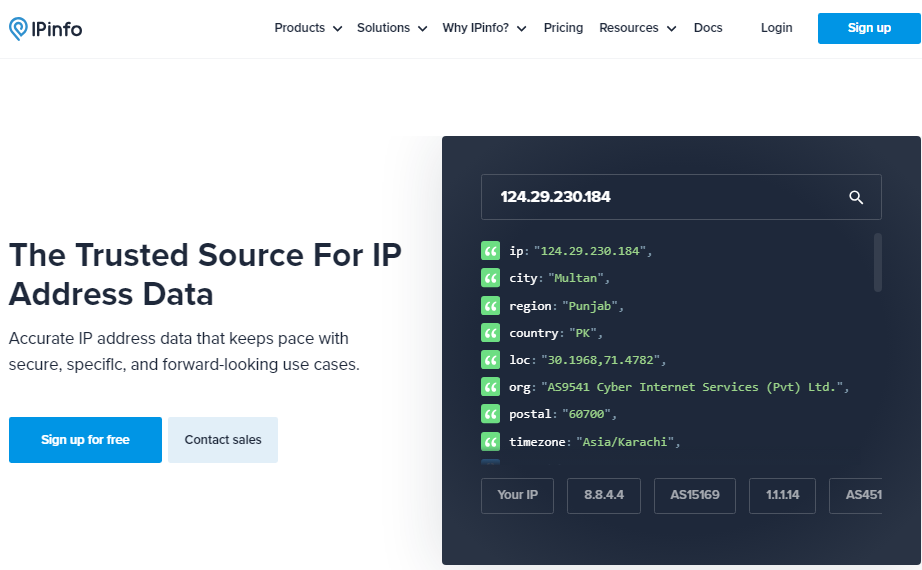
### Tools for Geolocation and IP Tracing

1. **MaxMind GeoIP**
2. **IPinfo**
3. **ipapi**
4. **IP Geolocation API**
5. **Traceroute and Traceroute Tools**

#### 1. MaxMind GeoIP

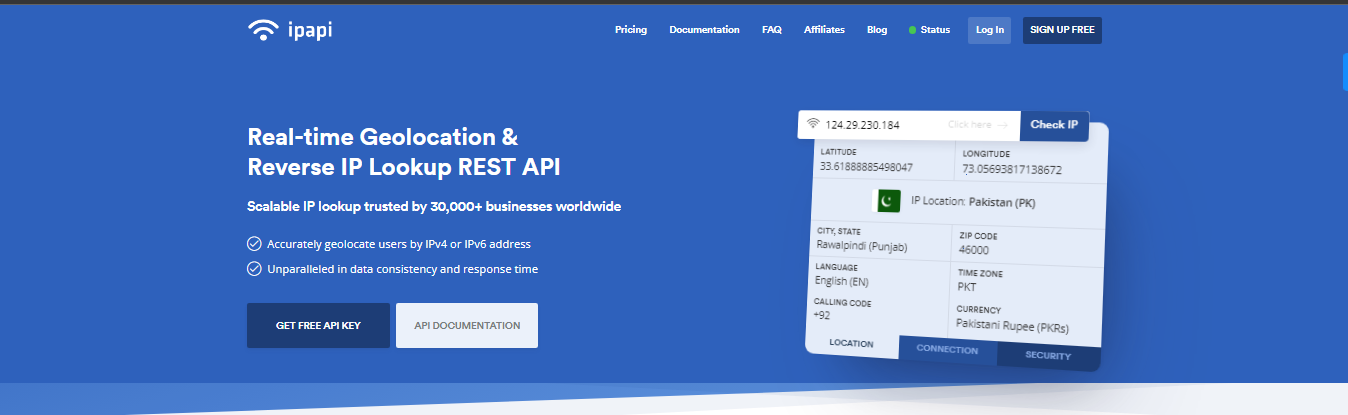
* **Description**: MaxMind provides GeoIP databases and web services that allow users to look up the geographical location of an IP address.
* **Features**: Provides detailed information such as country, city, latitude, longitude, ISP, and more.
* **Usage**: Commonly used in cybersecurity, fraud detection, and content localization. 

#### 2. IPinfo

* **Description**: IPinfo provides an API for IP geolocation and related data.
* **Features**: Includes data such as IP location, ASN (Autonomous System Number), and company information.
* **Usage**: Useful for network security, fraud detection, and personalization. 

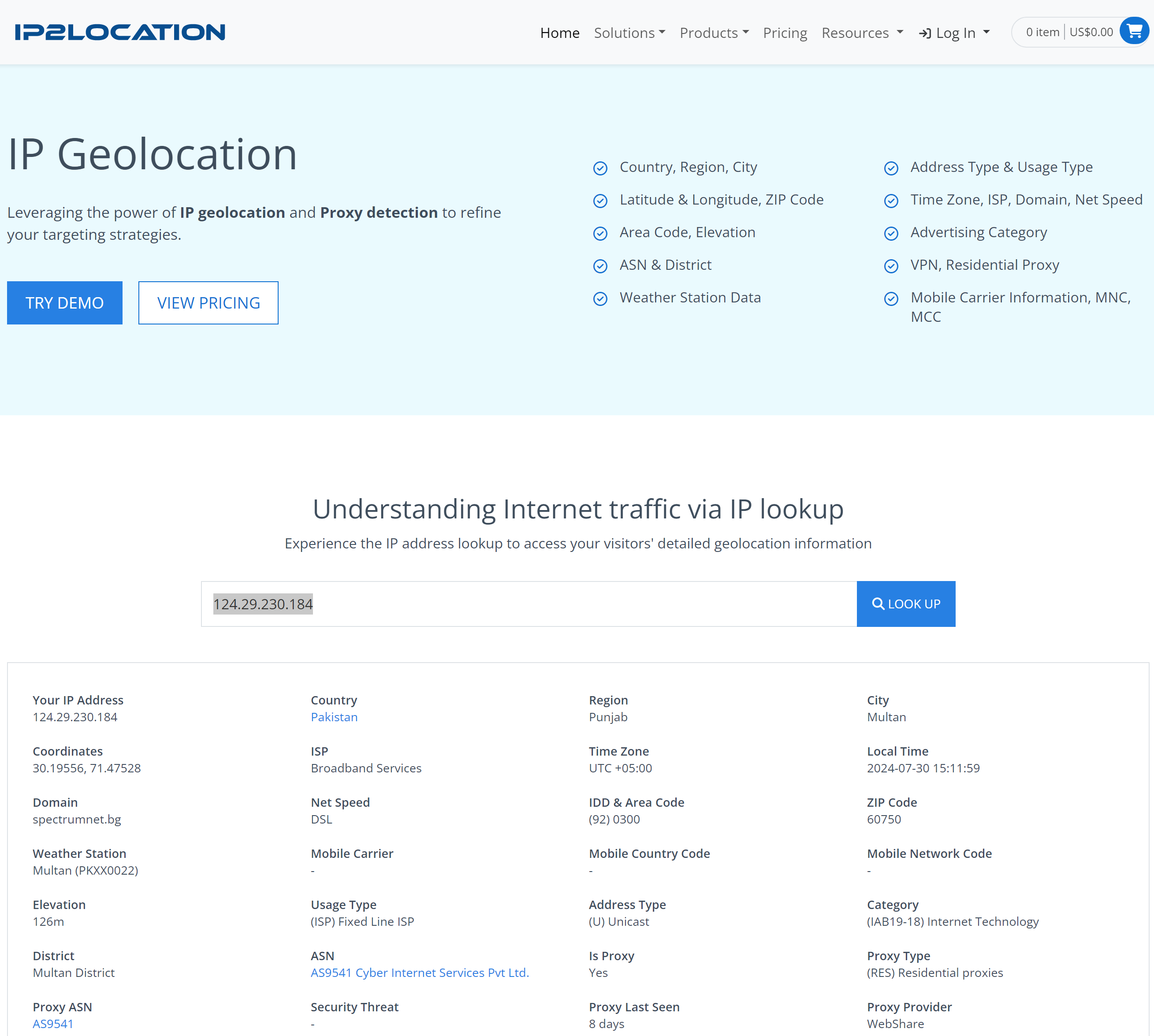
#### 3. ipapi

* **Description**: ipapi provides a real-time API service for IP geolocation.
* **Features**: Delivers geolocation data along with information on timezone, currency, and connection details.
* **Usage**: Ideal for location-based content and analytics.



#### 4. IP Geolocation API

* **Description**: IP Geolocation API provides comprehensive geolocation data for an IP address.
* **Features**: Includes information such as continent, country, state, city, zip code, latitude, longitude, and more.



### 5. Traceroute and Tracert:

#### What is Traceroute?

**Traceroute** is a network diagnostic tool used to track the path packets take from one computer to another across an IP network. It helps in identifying the route and measuring the transit delays of packets across the network. Traceroute sends a sequence of Internet Control Message Protocol (ICMP) echo requests, or User Datagram Protocol (UDP) packets, with incrementing Time-To-Live (TTL) values to each router along the path to the destination. When a packet's TTL reaches zero, the router returns an ICMP "Time Exceeded" message, revealing the router's IP address.

* **Purpose**: Diagnosing network issues, understanding the path to a destination, and identifying bottlenecks or points of failure.
* **Output**: A list of routers (hops) the packets pass through, with each hop showing the response time.

**Example Command (Linux)**:

**traceroute google.com**

#### What is Tracert?

**Tracert** (short for "trace route") is the Windows equivalent of traceroute. It performs the same function but uses ICMP echo requests instead of UDP packets by default.

* **Purpose**: Same as traceroute – diagnosing network issues, understanding the network path, and identifying points of delay or failure.
* **Output**: A list of routers (hops) the packets pass through, with each hop showing the response time.

**Example Command (Windows)**:

tracert example.com

### Detailed Explanation and Comparison

#### How Traceroute and Tracert Work

1. **Initiation**:
   * Both tools send packets to the destination with an initial TTL value of 1.
2. **TTL Increment**:
   * Each router along the path decreases the TTL by 1. When the TTL reaches zero, the router returns an ICMP "Time Exceeded" message.
3. **Incrementing TTL**:
   * The tools then increment the TTL value and send another packet, repeating the process until the destination is reached or the maximum TTL is exceeded.
4. **Final Response**:
   * When the destination is reached, an ICMP "Echo Reply" (for tracert) or a response from the target port (for traceroute) is returned.

#### Output Interpretation

* **Hop Number**: The sequence number of the hop along the path.
* **Router IP Address**: The IP address of the router that responded.
* **Response Times**: Typically, three response times are shown for each hop, representing the round-trip time (RTT) for three probe packets.

**Example Output (Linux Traceroute)**:

Traceroute google.com

traceroute to example.com (93.184.216.34), 30 hops max, 60 byte packets

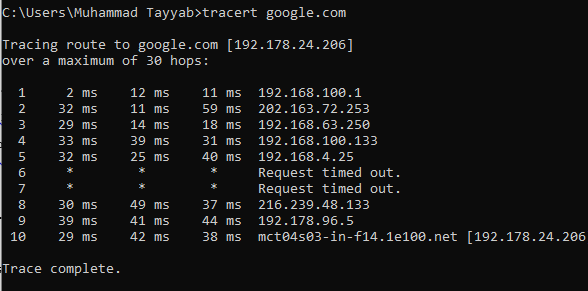
1 192.168.1.1 (192.168.1.1) 1.456 ms 1.218 ms 1.072 ms

2 10.0.0.1 (10.0.0.1) 2.144 ms 2.025 ms 1.931 ms

3 \* \* \*

4 93.184.216.34 (93.184.216.34) 10.384 ms 10.282 ms 10.174 ms

**Example Output (Windows Tracert)**:



### Differences Between Traceroute and Tracert

* **Operating Systems**:
  + **Traceroute**: Primarily used on Unix-like systems (Linux).
  + **Tracert**: Used on Windows systems.
* **Default Protocol**:
  + **Traceroute**: Uses UDP packets by default (can be configured to use ICMP or TCP).
  + **Tracert**: Uses ICMP echo requests by default.
* **Command Options**:
  + **Traceroute**: Offers more options for customizing the probes (e.g., protocol type, port number).
  + **Tracert**: Generally simpler with fewer customization options.

### 6. MTR

#### Overview

**MTR (My Traceroute)** is a network diagnostic tool that combines the functionality of both ping and traceroute. It provides real-time, continuous network route and performance analysis. MTR is used to diagnose and troubleshoot network issues by reporting the round-trip time (RTT), packet loss, and latency of each hop along the path from the source to the destination.

#### Key Features

1. **Combines Ping and Traceroute**: MTR provides the functionality of both ping and traceroute, offering continuous and real-time network route analysis.
2. **Real-Time Monitoring**: Continuously sends packets to the destination, updating results in real time.
3. **Comprehensive Metrics**: Reports latency, packet loss, and standard deviation for each hop.
4. **User-Friendly Interface**: Provides both text-based and graphical interfaces for ease of use.

#### How MTR Works

1. **Initialization**: MTR sends a sequence of packets to the destination with an initial TTL (Time-To-Live) value of 1.
2. **Incrementing TTL**: For each hop, the TTL value is incremented, allowing the packet to travel one more hop.
3. **Response Collection**: Each hop sends back an ICMP "Time Exceeded" message when the TTL reaches zero. This process continues until the destination is reached.
4. **Real-Time Updates**: MTR continuously sends packets, updating the results for each hop in real-time, showing variations in latency and packet loss over time.

#### Example Output

MTR provides a detailed report for each hop, showing the following information:

* **Host**: The IP address or hostname of each hop.
* **Loss%**: The percentage of packets lost at each hop.
* **Snt**: The number of packets sent to each hop.
* **Last**: The round-trip time of the last packet.
* **Avg**: The average round-trip time.
* **Best**: The shortest round-trip time.
* **Wrst**: The longest round-trip time.
* **StDev**: The standard deviation of round-trip times.

**Example Command (Linux)**:

mtr example.com

**Example Output**:

My traceroute [v0.86]

example.com (93.184.216.34) 2024-07-29T12:34:56+0000

Keys: Help Display mode Restart statistics Order of fields quit

Packets Pings

Host Loss% Snt Last Avg Best Wrst StDev

1. 192.168.1.1 0.0% 10 1.2 1.3 1.1 1.5 0.1

2. 10.0.0.1 0.0% 10 2.0 2.2 2.0 2.5 0.2

3. \* 100.0 10 0.0 0.0 0.0 0.0 0.0

4. 93.184.216.34 0.0% 10 10.3 10.2 10.1 10.4 0.1

#### Usage and Options

MTR can be run with various options to customize its behavior:

* **-r**: Report mode. Outputs statistics in a report format.
* **-c [count]**: Limits the number of pings sent.
* **-i [interval]**: Sets the interval between pings.
* **-w [timeout]**: Sets the timeout for each ping.
* **-b**: Displays both IP and hostname.
* **-4**: Forces MTR to use IPv4.
* **-6**: Forces MTR to use IPv6.

**Example Command with Options**:

mtr -r -c 100 example.com

This command runs MTR in report mode, sending 100 packets to the destination and then displaying the results.

### Conclusion

MTR is a powerful and versatile network diagnostic tool that combines the capabilities of ping and traceroute. By providing real-time updates and comprehensive metrics for each hop, MTR helps network administrators and engineers diagnose and troubleshoot network issues more effectively. Its ability to show variations in latency and packet loss over time makes it particularly useful for identifying intermittent network problems.

# **Wireshark**

**Wireshark** is a widely-used network protocol analyzer that lets you capture and interactively browse the traffic running on a computer network. It is an open-source tool that network administrators, security experts, and developers use to troubleshoot network issues, analyze traffic, and ensure network security. Wireshark supports a wide range of network protocols, making it a versatile tool for network analysis.

### Key Features of Wireshark

1. **Packet Capture**: Captures live network traffic from various types of networks.
2. **Deep Packet Inspection**: Examines the details of the data packets to provide in-depth analysis.
3. **Filtering**: Powerful filtering capabilities to isolate specific traffic or packets.
4. **User Interface**: Graphical user interface (GUI) for interactive exploration of network data.
5. **Protocol Support**: Supports hundreds of protocols, with more being added continually.
6. **Export Data**: Allows exporting of packet data to various file formats for further analysis.
7. **Decryption Support**: Can decrypt many protocols, including HTTPS, if the appropriate keys are provided.

### Using Wireshark

#### Interface Overview

When you open Wireshark, you'll see the following main sections:

1. **Interface List**: Lists available network interfaces on your machine for capturing traffic.
2. **Packet List Pane**: Displays the captured packets.
3. **Packet Details Pane**: Shows the details of the selected packet.
4. **Packet Bytes Pane**: Shows the raw data of the selected packet in hexadecimal and ASCII format.

#### Capturing Traffic

1. **Start Capture**: Select an interface from the list and click the "Start Capturing Packets" button (or press Ctrl+E).
2. **Stop Capture**: Click the "Stop Capturing Packets" button (or press Ctrl+E again).

#### Analyzing Packets

* **Packet List Pane**: Displays a summary of captured packets with columns for number, time, source, destination, protocol, length, and info.
* **Packet Details Pane**: Expandable tree view of the packet's protocol layers.
* **Packet Bytes Pane**: Hexadecimal and ASCII view of the packet's raw data.

#### Filtering Packets

Wireshark provides powerful filtering capabilities to help you focus on specific packets.

* **Display Filters**: Used to filter the packets displayed in the packet list. Examples:
  + ip.addr == 192.168.1.1: Show packets to/from IP address 192.168.1.1.
  + http: Show only HTTP packets.
  + tcp.port == 80: Show packets using TCP port 80.
* **Capture Filters**: Applied before capturing packets, filtering the traffic that Wireshark captures. Examples:
  + host 192.168.1.1: Capture traffic to/from IP address 192.168.1.1.
  + port 80: Capture traffic on port 80.

#### Exporting Data

Wireshark allows you to export packet data for further analysis or sharing.

* **File > Export Packet Dissections**: Export packet summaries in various formats.
* **File > Export Specified Packets**: Save a subset of packets to a new capture file.

### Advanced Features

#### Decryption

Wireshark can decrypt certain protocols, such as HTTPS, if you provide the necessary keys (e.g., pre-master secrets for SSL/TLS).

#### Customizing Wireshark

* **Coloring Rules**: Highlight packets based on specific criteria to improve readability.
* **Custom Columns**: Add columns to the packet list to display specific fields from the packets.

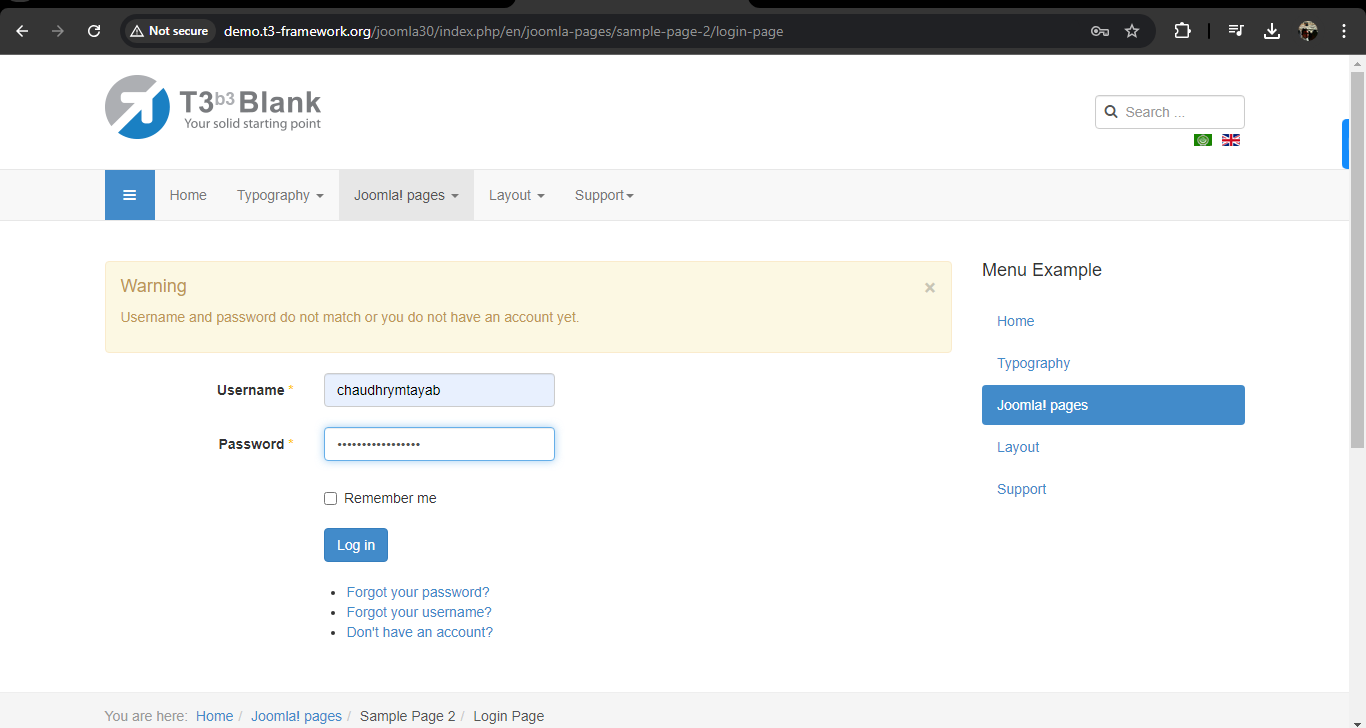
### Use Cases for Wireshark

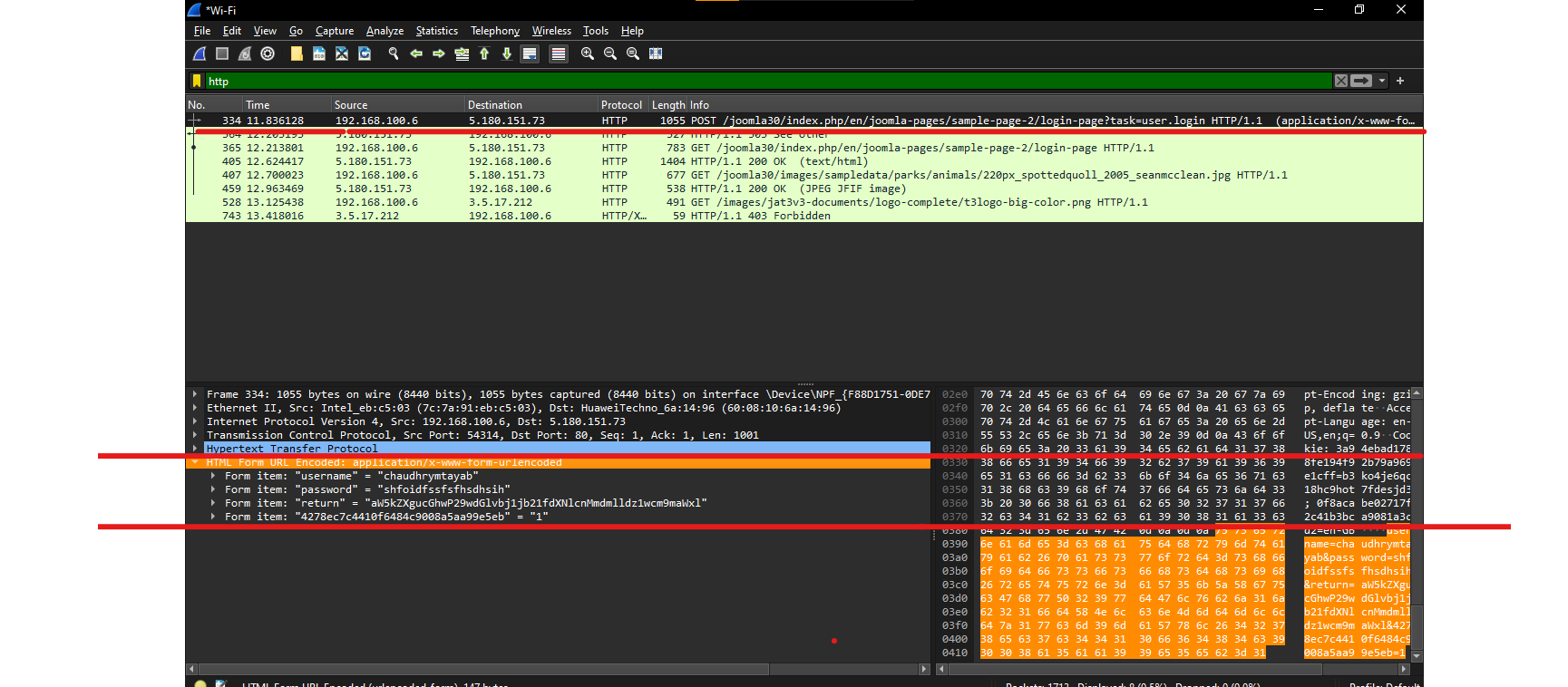
1. **Network Troubleshooting**: Identify and resolve network performance issues, such as latency, packet loss, and congestion.
2. **Security Analysis**: Detect and analyze security incidents, such as suspicious traffic, unauthorized access, and potential attacks.
3. **Protocol Development**: Validate the implementation of network protocols and troubleshoot communication issues.
4. **Educational Purposes**: Learn about network protocols and how data flows through networks.

### Conclusion

Wireshark is a powerful and versatile network analysis tool used by professionals for troubleshooting, security analysis, and protocol development. Whether you're running it on Linux or Windows, Wireshark provides detailed insights into network traffic, helping you to understand and manage your network effectively. Its comprehensive feature set and user-friendly interface make it an essential tool for anyone working with computer networks.

# Screenshot:





In the above picture you can see that in a website I put a username and a password which was caught by the **WIRESHARK** with an “**http**” request.

# Why ncbae.edu.pk is showing private IP [192.185.35.182]?

the answer is it is not a private IP. The ranges of public and private IP’s are:

### IP Classes: Public and Private

#### Overview of IP Address Classes

IP addresses are divided into classes to organize the allocation of addresses. The traditional IP address classes are A, B, C, D, and E, each with a different range and purpose.

#### Class A

* **Range**: 0.0.0.0 to 127.255.255.255
* **Default Subnet Mask**: 255.0.0.0 (or /8 in CIDR notation)
* **Purpose**: Designed for very large networks, such as large companies or ISPs.
* **Public Range**: 1.0.0.0 to 126.255.255.255
* **Private Range**: 10.0.0.0 to 10.255.255.255 (10.0.0.0/8)

#### Class B

* **Range**: 128.0.0.0 to 191.255.255.255
* **Default Subnet Mask**: 255.255.0.0 (or /16 in CIDR notation)
* **Purpose**: Intended for medium-sized networks.
* **Public Range**: 128.0.0.0 to 191.255.255.255
* **Private Range**: 172.16.0.0 to 172.31.255.255 (172.16.0.0/12)

#### Class C

* **Range**: 192.0.0.0 to 223.255.255.255
* **Default Subnet Mask**: 255.255.255.0 (or /24 in CIDR notation)
* **Purpose**: Suitable for small networks.
* **Public Range**: 192.0.0.0 to 223.255.255.255
* **Private Range**: 192.168.0.0 to 192.168.255.255 (192.168.0.0/16)