# Section 1:

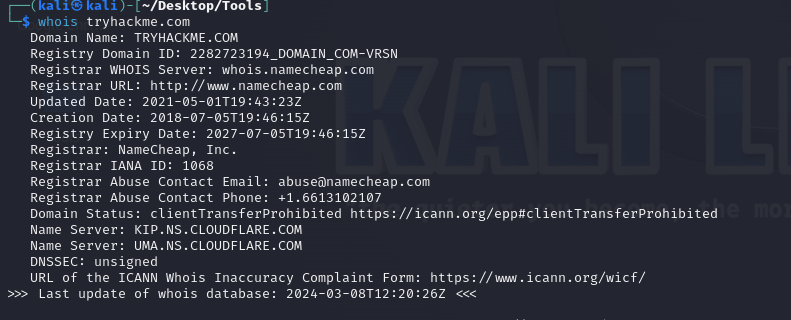
Whois

It is a request and response protocol which is considered a very helpful command-line utility.

It is extremely helpful for looking up information about a **domain**. The domain registrar is responsible for maintaining the whois records for the domain names it’s leasing. The whois server replies with different information related to the domain requested.

|  |  |
| --- | --- |
| **Registrar** | Via which entity/registrar was the domain registered? |
| **Contact information of registrant** | Name, organization, address, phone, among other things. (unless made hidden via a privacy service) (tryhackme.com) |
| **Name Server** | The server which is responsible for **resolving** the domain name. |
| **Important Dates** | Date of registration of the domain name  Date of expiration of the domain name  Data of last update of the domain name |

whois tryhackme.com



Website Fingerprinting

Open-source intelligence: use online resources to gather information about the target without interacting directly with the target.

Example: Netcraft: analyzes the given website

<https://sitereport.netcraft.com/?url=http://tryhackme.com>

Through this website, you can see the hosting history and the versions used.

BIGIP: It’s a firewall used to detect malicious software.

Havebeenpwnd: Used to see if my email has been leaked or not

IntelligenceX: Are there leaks? What are they?

Google Dorking

**The cheat sheet**:

Use operators!

<https://gist.github.com/sundowndev/283efaddbcf896ab405488330d1bbc06>

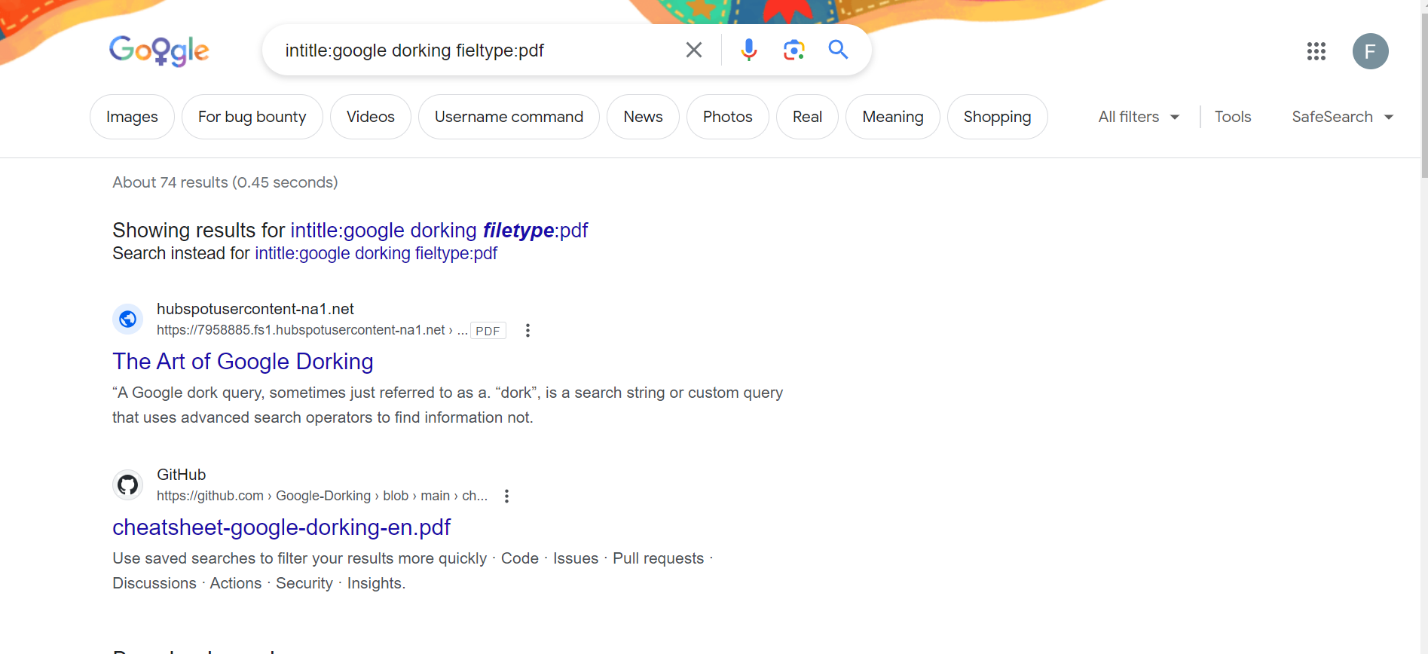
A screenshot of a black screen

Description automatically generated

1. site: tryhackme.com A screenshot of a computer

   Description automatically generated
2. site: tryhackme.com intitle: reconnaissanceA screenshot of a computer

   Description automatically generated
3. site: \*.aast.edu intitle: loginA screenshot of a computer

   Description automatically generated
4. intitle: google dorking fieltype:pdf



1. Google Dorks for Bug Bounty: <https://taksec.github.io/google-dorks-bug-bounty/> A screenshot of a computer program

   Description automatically generated

Subdomains

TOOLS

amass

Sublist3r

sub finder

* These tools can do active and passive reconnaissance. However, they do passive recon by default.
* Project Discovery created sub finder.
* Sub finder can find more websites/subdomains than sub lister
* Subfinder🡪

1. ./subfinder -d tryhackme.com

A screen shot of a computer

Description automatically generated

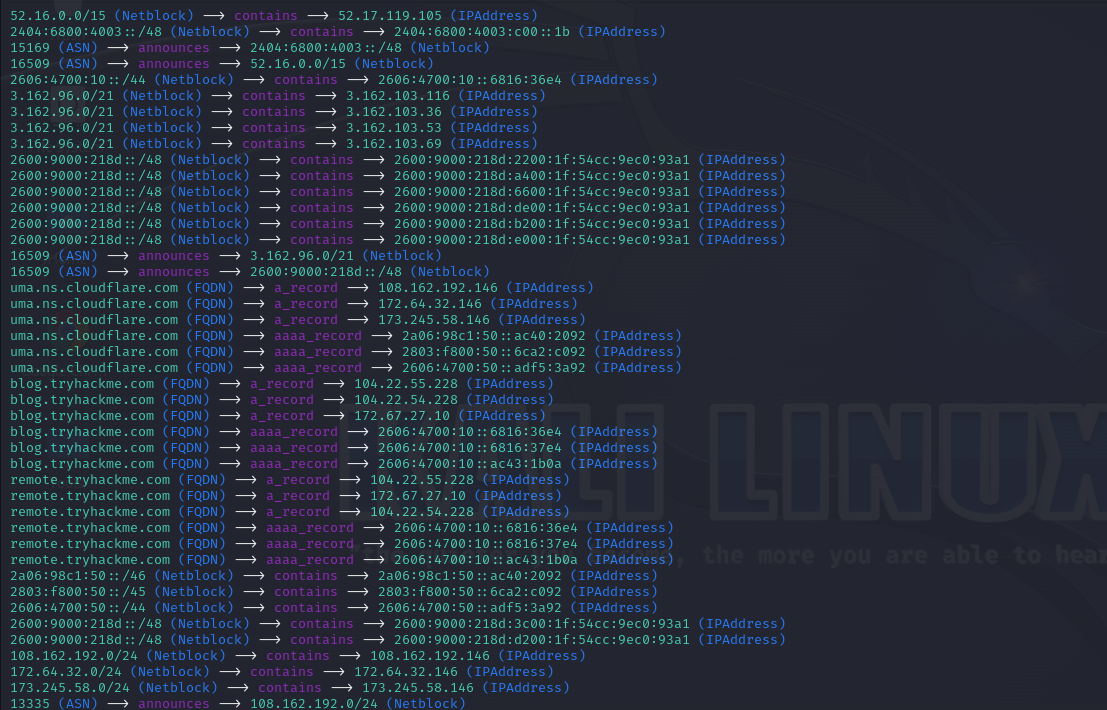
1. It’s faster and gets more results than sublist3er

* Sublist3r: Created by an Egyptian Programmer

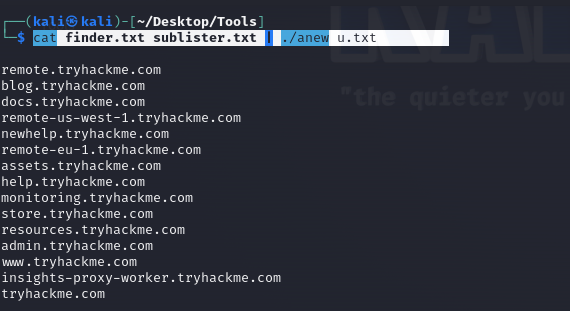
1. -d: for domain inspection
2. -b: for brute force
3. -p: scan ports active recon
4. -v: to see what is happening
5. -e: specify the search engine you’re going to use
6. -o: to specify where you want to save the result or output
7. python3 sublist3r.py -d tryhackme.com: used to google dork to find subdomains.
8. python3 sublist3r.py -d tryhackme.com -o sublister.txt: to output results.
9. Sub finder uses GO and sub lister uses python
10. Go is compiled whereas python is interpreted
11. cat sublister.text 🡪 Reads the items inside the text
12. wc -l sublister.txt🡪13 sublister.txt ( wc gives us the number of lines through which enables us to know how many subdomains was the sublister able to find

* amass: queries,subdomain,etc..

1. amass enum -passive -d tryhackme.com
2. amass is really slow
3. However, it gives a handful of information about each subdomain



* cat finder.txt sublister.txt | ./anew u.txt



* wc -l u.txt 🡪 Counts how many subdomain exist

Intro to Active Recon

What firewall is the domain using?

Wafw00f

Wafw00f: This tool is used to get information about firewalls.

Syntax:

* wafw00f aast.edu actively connecting with aast.edu, analyzes the presence of firewall and what firewall does it use. The firewall is provided by AWS in this case A screenshot of a computer

  Description automatically generated
* waf00f -l lists of firewalls that it can identify.

A screen shot of a computer

Description automatically generated

* wafw00f -h helps you with the syntax for all the tools of wafW00f

A screenshot of a computer screen

Description automatically generated

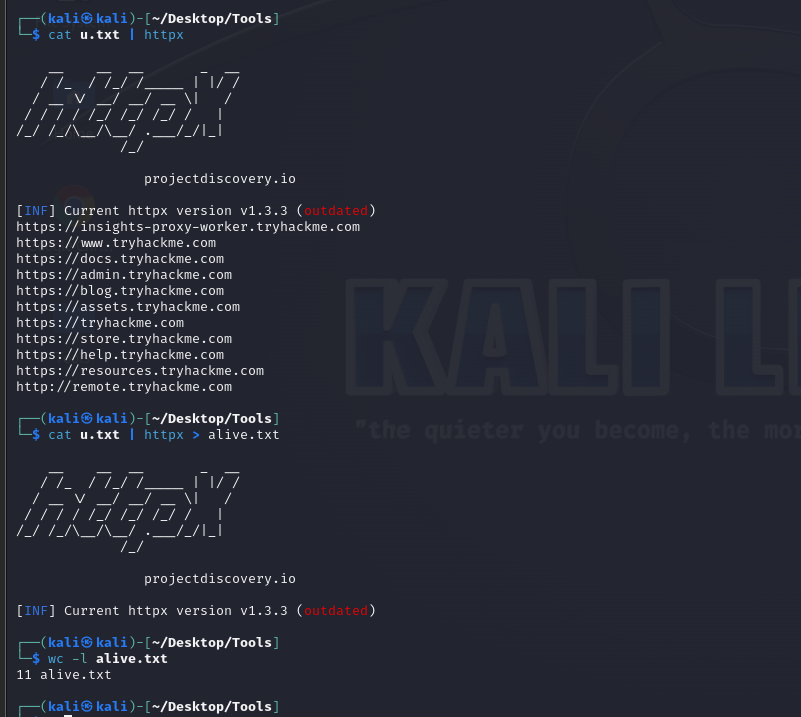
Httpx:

1)cat u.txt | httpx

2) cat u.txt | httpx > alive.txt

3) wc -l alive.txt

These three preceding commands use httpx to see which commands are alive and working then saves them in alive.txt. Then, the number of subdomains which is printed using word count🡪 wc



# Section 2:

Local Host Discovery:

**What do I want to Know?**

1. **How many devices are running on the network?**
2. **Regarding each device, what is the port number of each one?**
3. **What is the version of the service the port number is using?**

**Case 1: I have your IP address and we are in the same network.**

Example: You’re using http service on Port 80 and I want to send you a GET Request. Your IP address is <http://10.10.1.2>.

I need its MAC address so I can send this IP address a message.

First: Use whois to send an ARP Request

🡪 whois 10.10.1.2🡪 the ARP Response will be something like this 🡪 10.10.1.2 is at 2E:56: 74:

This means that this IP address is indeed in the network and is active.

Nmap can do that. It will send an ARP Request to figure out if the device is working or not and is able to port scan.

I want to use **nmap** to figure out the number of devices in the network.

Host Discovery without port scan:

You’re supposed to find the MAC address of every IP address that responded to you.

**🡪Nmap 10.10.1/24 -sn**

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Host Discovery (outside the network):

Port Scanning

Scanning

Udp port scanning

Tcp port scanning

TCP Port:

1)Stealth Scan -Ss (it’s faster than connect scan, because there is no 3 way handshaking)

1. Send

2) Send ACK

3) Reset