Lab 17 – Dig Command

Lab Purpose:

- You're going to learn how to use `dig`, a **DNS lookup tool**.
- **DNS** stands for **Domain Name System** — it's like the phonebook of the internet. When you type a website (like `google.com`) into your browser, DNS translates that name into an IP address (like `142.250.190.78`), which is what computers actually use to communicate.
- `dig` helps you look up that information **directly** from the command line.

Lab Tool:

- You'll be using **Kali Linux**.
  (If you're using a normal Linux machine or Mac, that's fine too.)

Lab Topology:

- You don't need any complex network setup. Just your **Kali Linux machine** and a terminal window.

Lab Walkthrough:

Task 1: Checking if `dig` is installed and running your first lookup

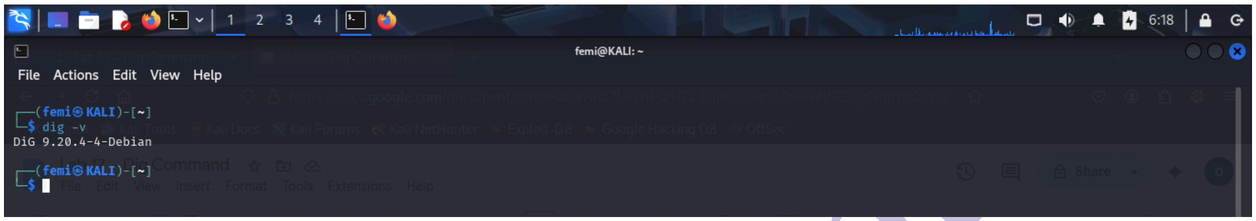
Step 1: Open your terminal
(Shortcut: Press `Ctrl + Alt + T`)

Step 2: Check if dig is installed by typing: *dig -v*

Explanation:

- `-v` stands for **version**.
- This command simply asks `dig`: “Hey, what version are you?”

✅ If you see something like `DiG 9.16.1`, it means it's installed and ready to use.



```
┌──(femi㉿KALI)-[~]
└─$ dig -v
DiG 9.20.4-4-Debian

┌──(femi㉿KALI)-[~]
└─$ 
```

Step 3: Run your first `dig` command:

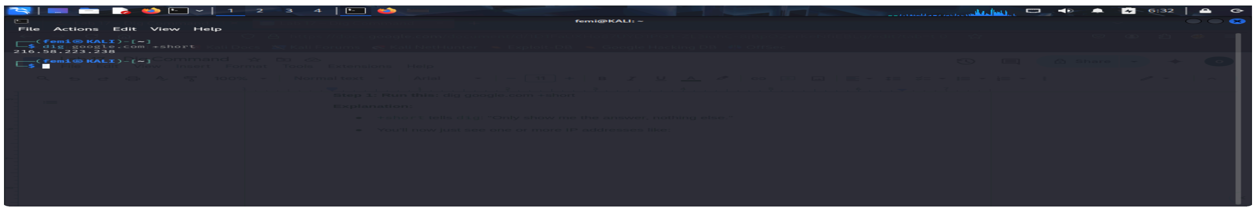
dig google.com

Task 2: Getting a simple, clean result

Step 1: Run this: dig google.com +short

Explanation:

- `+short` tells `dig`: “Only show me the answer, nothing else.”
- You'll now just see one or more IP addresses like:



```
┌──(femi㉿KALI)-[~]
└─$ dig google.com +short
216.58.223.238

┌──(femi㉿KALI)-[~]
└─$ 
```

Step 2: Run this: dig google.com +noall +answer

What does it mean?

- `dig google.com` → Asks for DNS info about `google.com`.
- `+noall` → Tells `dig`:

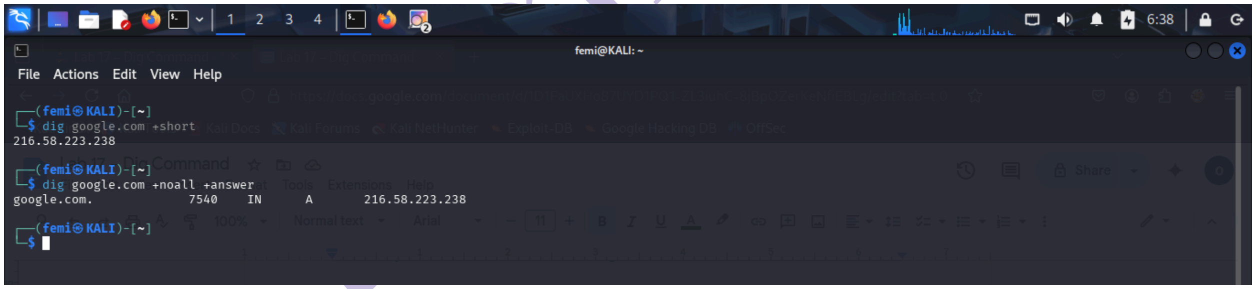
  "Don't show me all the extra stuff (like headers, query times, etc.)."

- `+answer` → Tells `dig`:

  "ONLY show me the actual **answer**."

✅ **Together:**
You are saying:

"Only show me the **final answer**. Nothing else. No extra noise."



```
┌──(femi㉿KALI)-[~]
└─$ dig google.com +short
216.58.223.238

┌──(femi㉿KALI)-[~]
└─$ dig google.com +noall +answer
google.com.             7540    IN      A       216.58.223.238

┌──(femi㉿KALI)-[~]
└─$ 
```

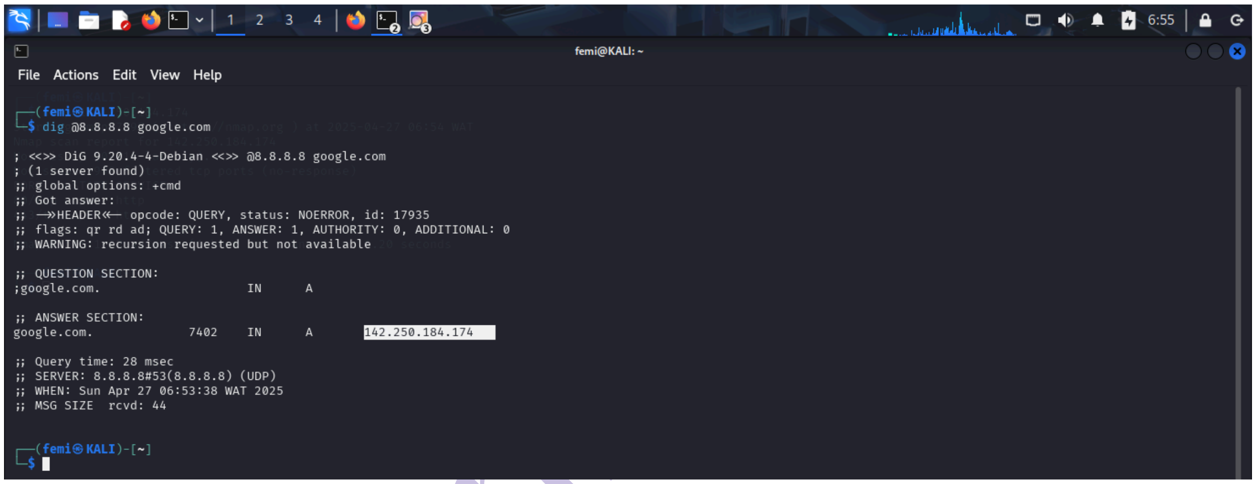
Task 3: Using a specific DNS server

Step 1: Run this: dig @8.8.8.8 google.com

Explanation:

- `@8.8.8.8` tells `dig`: “Ask Google's public DNS server, not the default one from my ISP (Internet Service Provider).”
- 8.8.8.8 is Google's free, public DNS server.

✅ Useful if you want to verify results against a **different nameserver**!



```
┌──(femi㉿KALI)-[~]
└─$ dig @8.8.8.8 google.com

; <<>> DiG 9.20.4-4-Debian <<>> @8.8.8.8 google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17935
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             7402    IN      A       142.250.184.174

;; Query time: 28 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Sun Apr 27 06:53:38 WAT 2025
;; MSG SIZE  rcvd: 44


┌──(femi㉿KALI)-[~]
└─$ 
```

Task 4: Querying all available DNS records (ANY)

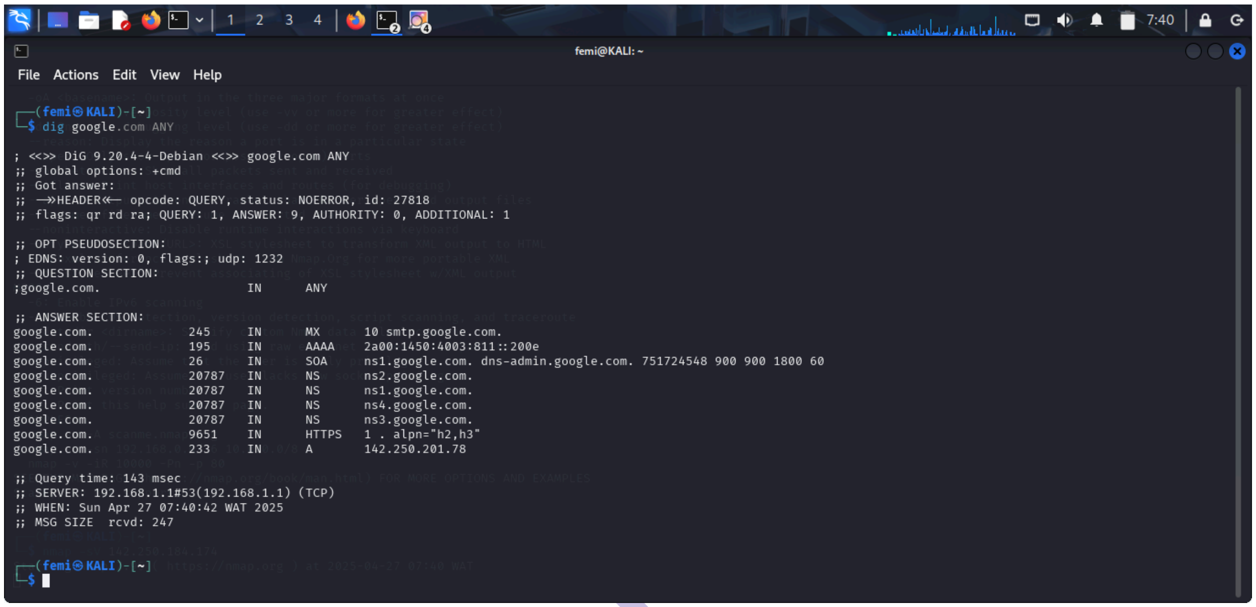
Step 1: Run this: dig google.com ANY

Explanation:

- `ANY` asks: “Show me **everything** you know about `google.com` — every type of record.”
- You might see:
  - **A** records (normal IP addresses)
  - **MX** records (mail servers)
  - **NS** records (nameservers)

- **TXT** records (miscellaneous text, sometimes used for domain ownership verification)

✅ Handy for gathering **all possible information** about a domain.



```
┌──(femi㉿KALI)-[~]
└─$ dig google.com ANY

; <<>> DiG 9.20.4-4-Debian <<>> google.com ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27818
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;google.com.                    IN      ANY

;; ANSWER SECTION:
google.com.             245     IN      MX      10 smtp.google.com.
google.com.             195     IN      AAAA    2a00:1450:4003:811::200e
google.com.             26      IN      SOA     ns1.google.com. dns-admin.google.com. 751724548 900 900 1800 60
google.com.             20787   IN      NS      ns2.google.com.
google.com.             20787   IN      NS      ns1.google.com.
google.com.             20787   IN      NS      ns4.google.com.
google.com.             20787   IN      NS      ns3.google.com.
google.com.             9651    IN      HTTPS   1 . alpn="h2,h3"
google.com.             233     IN      A       142.250.201.78

;; Query time: 143 msec
;; SERVER: 192.168.1.1#53(192.168.1.1) (TCP)
;; WHEN: Sun Apr 27 07:40:42 WAT 2025
;; MSG SIZE  rcvd: 247

┌──(femi㉿KALI)-[~]
└─$ 
```

Task 5: Looking up specific types of records

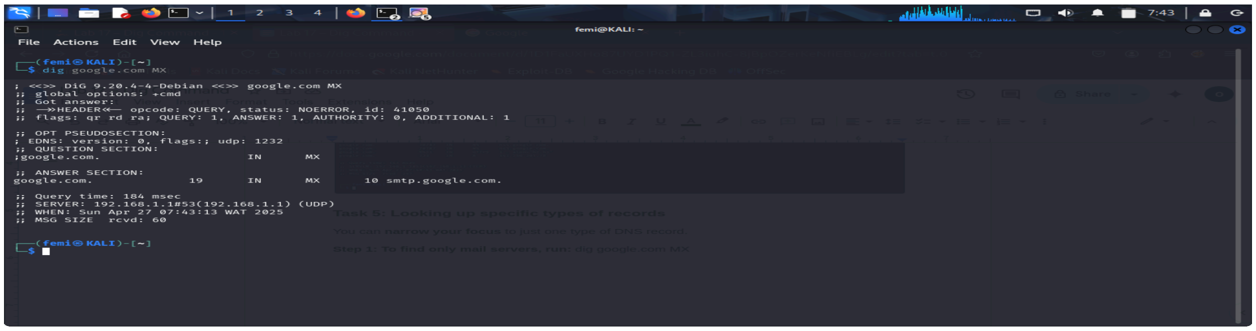
You can narrow your focus to just one type of DNS record.

Step 1: To find only mail servers, run: dig google.com MX

Explanation:

- MX = Mail eXchange.
- Shows mail servers handling emails for that domain.

✅ Useful if you're investigating how a company handles email.



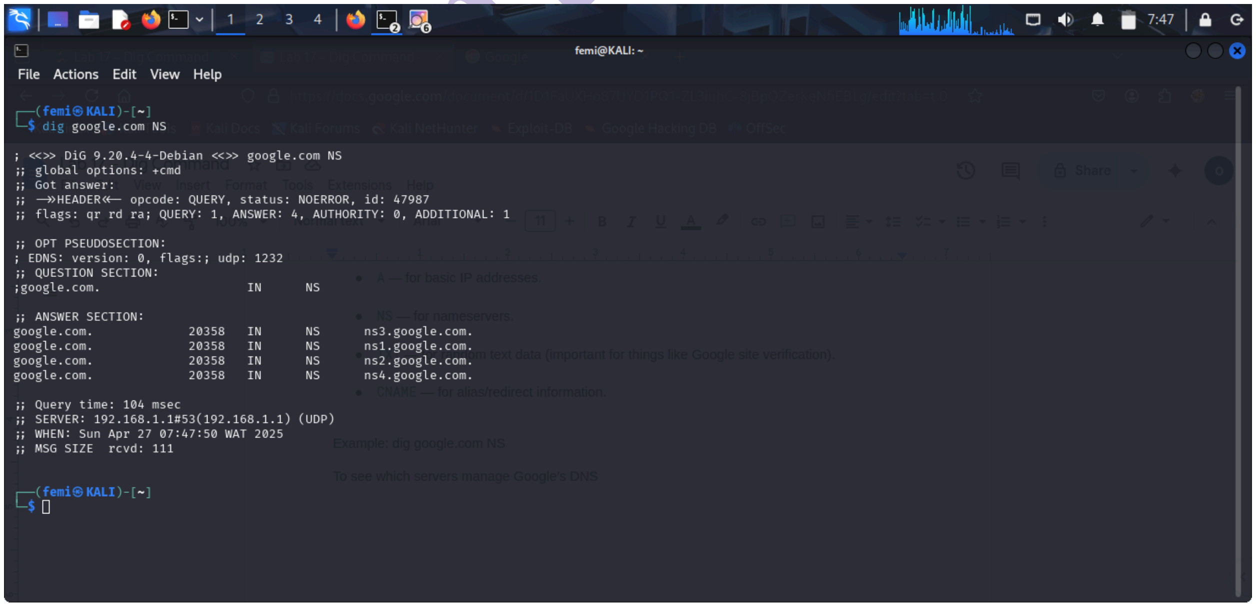
```
┌──(femi㉿KALI)-[~]
└─$ dig google.com MX

; <<>> DiG 9.20.4-4-Debian <<>> google.com MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41050
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;google.com.                    IN      MX

;; ANSWER SECTION:
google.com.             19      IN      MX      10 smtp.google.com.

;; Query time: 184 msec
;; SERVER: 192.168.1.1#53(192.168.1.1) (UDP)
;; WHEN: Sun Apr 27 07:43:13 WAT 2025
;; MSG SIZE  rcvd: 60

┌──(femi㉿KALI)-[~]
└─$ 
```

Other types you can specify instead of MX:

- `A` — for basic IP addresses.
- `NS` — for nameservers.
- `TXT` — for random text data (important for things like Google site verification).
- `CNAME` — for alias/redirect information.

Example: dig google.com NS

To see which servers manage Google's DNS



```
┌──(femi㉿KALI)-[~]
└─$ dig google.com NS

; <<>> DiG 9.20.4-4-Debian <<>> google.com NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47987
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;google.com.                    IN      NS

;; ANSWER SECTION:
google.com.             20358   IN      NS      ns3.google.com.
google.com.             20358   IN      NS      ns1.google.com.
google.com.             20358   IN      NS      ns2.google.com.
google.com.             20358   IN      NS      ns4.google.com.

;; Query time: 104 msec
;; SERVER: 192.168.1.1#53(192.168.1.1) (UDP)
;; WHEN: Sun Apr 27 07:47:50 WAT 2025
;; MSG SIZE  rcvd: 111

┌──(femi㉿KALI)-[~]
└─$ 
```

Task 6: Tracing the DNS resolution path

Step 1: Run this: dig +short +trace google.com

Explanation:

- `+trace` → First, trace the path (root DNS → .com → google.com).
- `+short` → After tracing, show only the short clean output (IP addresses, etc.)

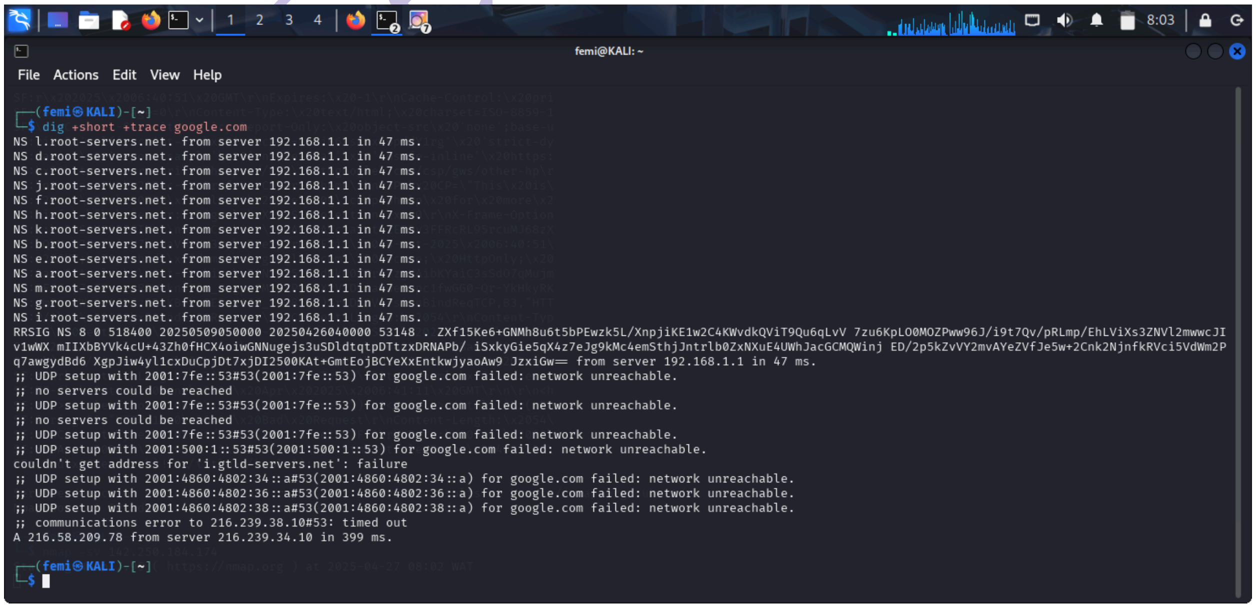
BUT there's a little important thing:

- `+trace` forces `dig` to show full tracing info.
- `+short` does not really affect how `+trace` looks because tracing needs full details to show the path.

So when you use both together (`+short +trace`):

- You still see lots of tracing output (because tracing can't be short).
- Only the final answer part may be shortened a little.

✅ So `+short` doesn't do much when combined with `+trace`.
✅ It matters more without tracing.



```
┌──(femi㉿KALI)-[~]
└─$ dig +short +trace google.com
NS l.root-servers.net. from server 192.168.1.1 in 47 ms.
NS d.root-servers.net. from server 192.168.1.1 in 47 ms.
NS c.root-servers.net. from server 192.168.1.1 in 47 ms.
NS j.root-servers.net. from server 192.168.1.1 in 47 ms.
NS f.root-servers.net. from server 192.168.1.1 in 47 ms.
NS h.root-servers.net. from server 192.168.1.1 in 47 ms.
NS k.root-servers.net. from server 192.168.1.1 in 47 ms.
NS b.root-servers.net. from server 192.168.1.1 in 47 ms.
NS e.root-servers.net. from server 192.168.1.1 in 47 ms.
NS a.root-servers.net. from server 192.168.1.1 in 47 ms.
NS m.root-servers.net. from server 192.168.1.1 in 47 ms.
NS g.root-servers.net. from server 192.168.1.1 in 47 ms.
NS i.root-servers.net. from server 192.168.1.1 in 47 ms.
RRSIG NS 8 0 518400 20250509050000 20250426040000 53148 . ZXf15Ke6+GNMh8u6t5bPEwzk5L/XnpjiKE1w2C4KWvdkQViT9Qu6qLvV 7zu6KpLO0MOZPww96J/i9t7Qv/pRLmp/EhLViXs3ZNVl2mwwcJI
v1wWX mIIXbBYVk4cU+43Zh0fHCX4oiwGNNugejs3uSDldtqtpDTtzxDRNAPb/ iSxkyGie5qX4z7eJg9kMc4emSthjJntrlb0ZxNXuE4UWhJacGCMQWinj ED/2p5kZvVY2mvAYeZVfJe5w+2Cnk2NjnfkRVci5VdWm2P
q7awgydBd6 XgpJiw4yl1cxDuCpjDt7xjDI2500KAt+GmtEojBCYeXxEntkwjyaoAw9 JzxiGw== from server 192.168.1.1 in 47 ms.
;; UDP setup with 2001:7fe::53#53(2001:7fe::53) for google.com failed: network unreachable.
;; no servers could be reached
;; UDP setup with 2001:7fe::53#53(2001:7fe::53) for google.com failed: network unreachable.
;; no servers could be reached
;; UDP setup with 2001:7fe::53#53(2001:7fe::53) for google.com failed: network unreachable.
;; UDP setup with 2001:500:1::53#53(2001:500:1::53) for google.com failed: network unreachable.
couldn't get address for 'i.gtld-servers.net': failure
;; UDP setup with 2001:4860:4802:34::a#53(2001:4860:4802:34::a) for google.com failed: network unreachable.
;; UDP setup with 2001:4860:4802:36::a#53(2001:4860:4802:36::a) for google.com failed: network unreachable.
;; UDP setup with 2001:4860:4802:38::a#53(2001:4860:4802:38::a) for google.com failed: network unreachable.
;; communications error to 216.239.38.10#53: timed out
A 216.58.209.78 from server 216.239.34.10 in 399 ms.

┌──(femi㉿KALI)-[~]
└─$ 
```

Task 7: Reverse DNS lookup

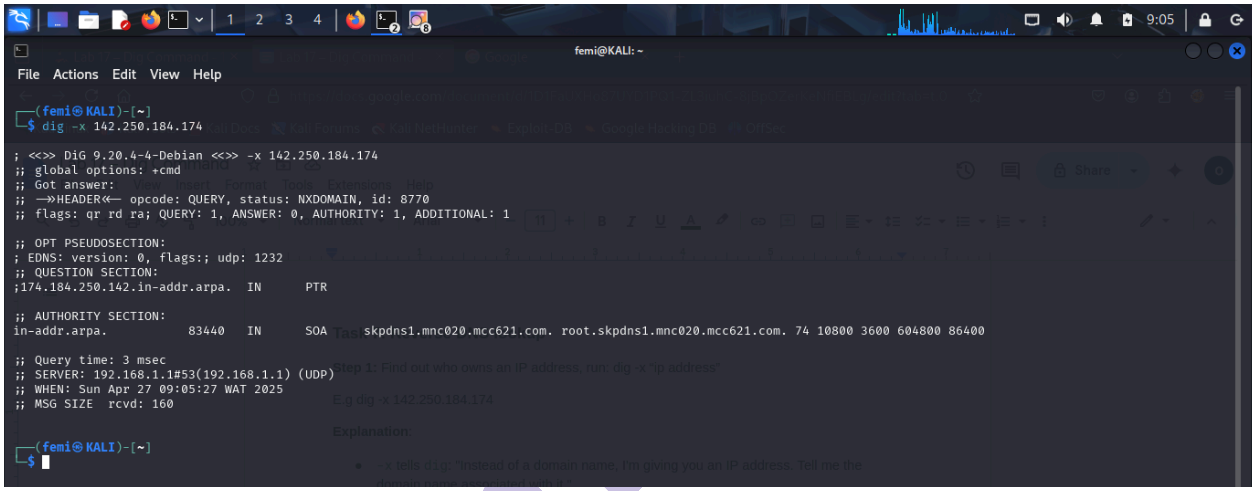
Step 1: Find out who owns an IP address, run: dig -x “ip address”

E.g dig -x 142.250.184.174

Explanation:

- `-x` tells `dig`: "Instead of a domain name, I'm giving you an IP address. Tell me the domain name associated with it."

- Example result:



```
┌──(femi㉿KALI)-[~]
└─$ dig -x 142.250.184.174

; <<>> DiG 9.20.4-4-Debian <<>> -x 142.250.184.174
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 8770
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;174.184.250.142.in-addr.arpa.  IN      PTR

;; AUTHORITY SECTION:
in-addr.arpa.           83440   IN      SOA     skpdns1.mnc020.mcc621.com. root.skpdns1.mnc020.mcc621.com. 74 10800 3600 604800 86400

;; Query time: 3 msec
;; SERVER: 192.168.1.1#53(192.168.1.1) (UDP)
;; WHEN: Sun Apr 27 09:05:27 WAT 2025
;; MSG SIZE  rcvd: 160


┌──(femi㉿KALI)-[~]
└─$ 
```

Task 8: Lookup multiple domains at once

Step 1: Create a file called `domain_names.txt and type in domain names one per line`

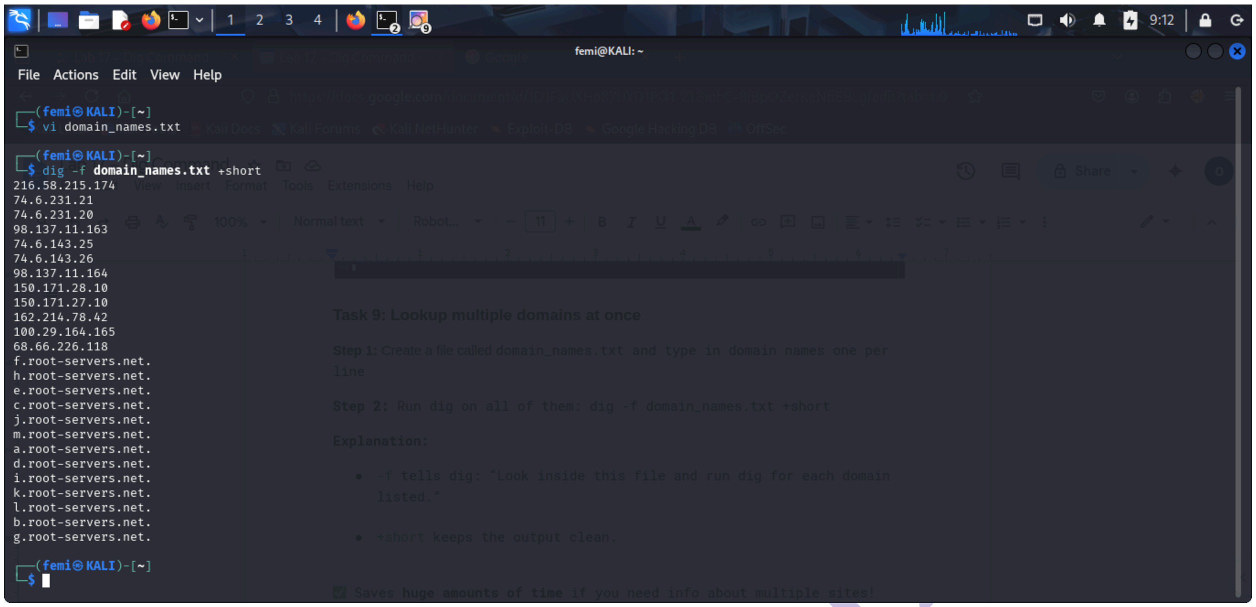
`Step 2:` `Run dig on all of them: dig -f domain_names.txt +short`

`Explanation:`

- `-f tells dig: “Look inside this file and run dig for each domain listed.”`

- `+short keeps the output clean.`

✅ `Saves` **`huge amounts of time`** `if you need info about multiple sites!`



```
┌──(femi㉿KALI)-[~]
└─$ vi domain_names.txt

┌──(femi㉿KALI)-[~]
└─$ dig -f domain_names.txt +short
216.58.215.174
74.6.231.21
74.6.231.20
98.137.11.163
74.6.143.25
74.6.143.26
98.137.11.164
150.171.28.10
150.171.27.10
162.214.78.42
100.29.164.165
68.66.226.118
f.root-servers.net.
h.root-servers.net.
e.root-servers.net.
c.root-servers.net.
j.root-servers.net.
m.root-servers.net.
a.root-servers.net.
d.root-servers.net.
i.root-servers.net.
k.root-servers.net.
l.root-servers.net.
b.root-servers.net.
g.root-servers.net.

┌──(femi㉿KALI)-[~]
└─$ 
```

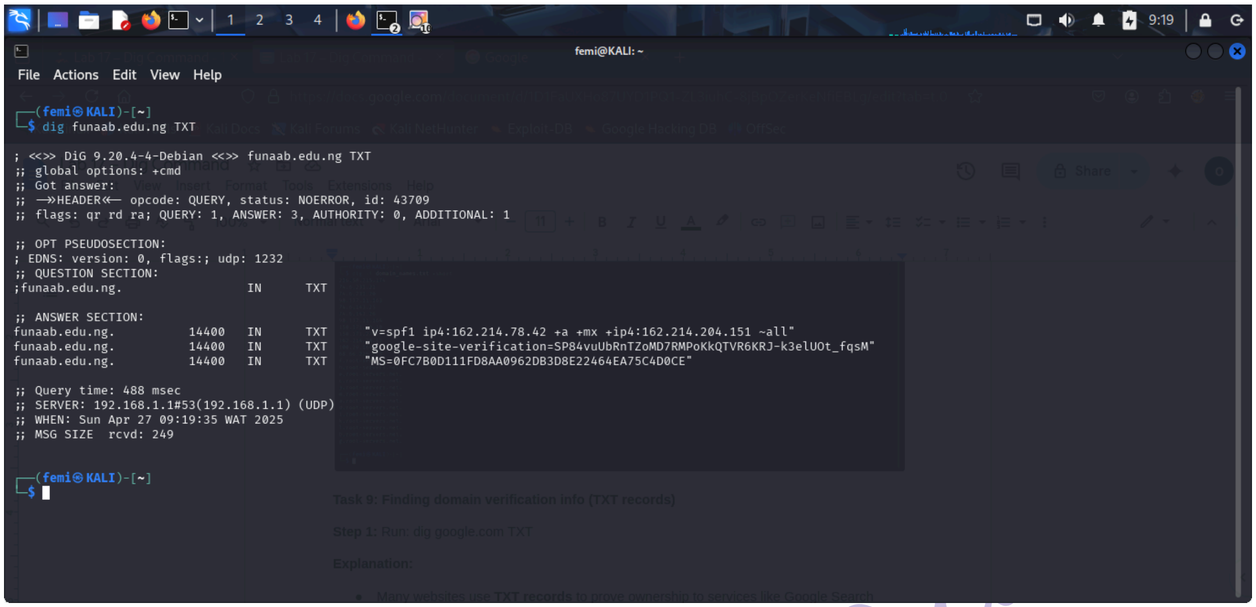
Task 9: Finding domain verification info (TXT records)

Step 1: Run: dig funaab.edu.ng TXT

Explanation:

- Many websites use **TXT records** to prove ownership to services like Google Search Console or to configure email security settings.
- You might see records mentioning things like:
  - **"v=spf1..."** (email security settings)
  - **"google-site-verification=..."** (domain ownership)

✅ **TXT records** are super important when dealing with website verification, email authentication, etc.



```
┌──(femi㉿KALI)-[~]
└─$ dig funaab.edu.ng TXT

; <<>> DiG 9.20.4-4-Debian <<>> funaab.edu.ng TXT
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43709
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;funaab.edu.ng.                 IN      TXT

;; ANSWER SECTION:
funaab.edu.ng.          14400   IN      TXT     "v=spf1 ip4:162.214.78.42 +a +mx +ip4:162.214.204.151 ~all"
funaab.edu.ng.          14400   IN      TXT     "google-site-verification=SP84vuUbRnTZoMD7RMPoKkQTVR6KRJ-k3elUOt_fqsM"
funaab.edu.ng.          14400   IN      TXT     "MS=0FC7B0D111FD8AA0962DB3D8E22464EA75C4D0CE"

;; Query time: 488 msec
;; SERVER: 192.168.1.1#53(192.168.1.1) (UDP)
;; WHEN: Sun Apr 27 09:19:35 WAT 2025
;; MSG SIZE  rcvd: 249


┌──(femi㉿KALI)-[~]
└─$ 
```