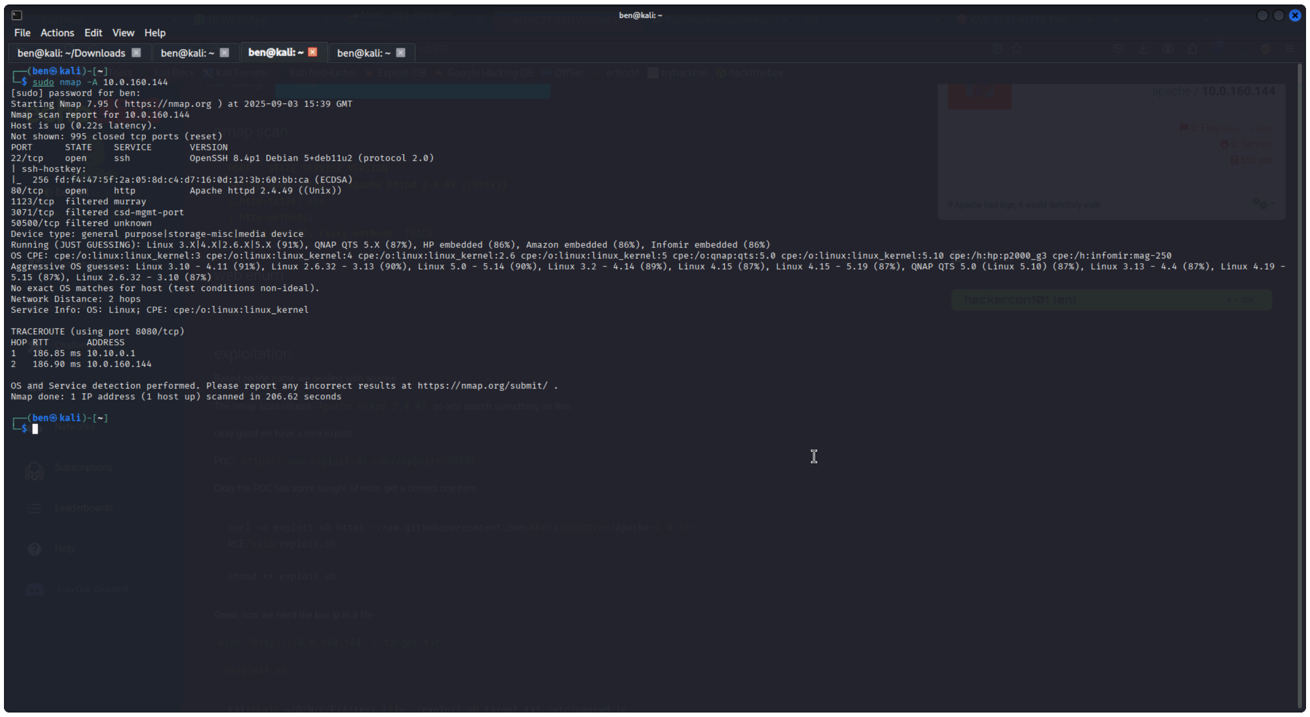
NMAP AND SCAPY USAGE

```
NMAP USAGE
```

Nmap (Network Mapper) is an open-source network scanning tool used to discover devices, identify open ports, detect services and their versions, and gather information about operating systems on a network. It is widely used for network auditing, security assessments, and troubleshooting. Nmap allows users to do a bunch of things that are related to a wide range of network-related tasks.

1. **Network Discovery:** With Nmap, users can scan networks and discover devices and hosts on a network, allowing network admins to understand the network more efficiently.
2. **Port Scanning:** It can determine which ports are open and which services are running on those ports, which is critical for security assessments and vulnerability scanning.
3. **OS Fingerprinting:** Nmap can attempt to identify the operating system running on a target host by analyzing various characteristics of network packets.
4. **Vulnerability Assessment:** It's a valuable tool for identifying potential vulnerabilities in systems and services, aiding in proactive security measures.
5. **Network Monitoring:** Nmap can be used for continuous monitoring to detect changes in the network environment.
   Its a tool that helps individuals especially security people scan multiple networks or a wide range of networks to gain some basic information about them. Some of these info are opened ports, available services, operating system used and many more.

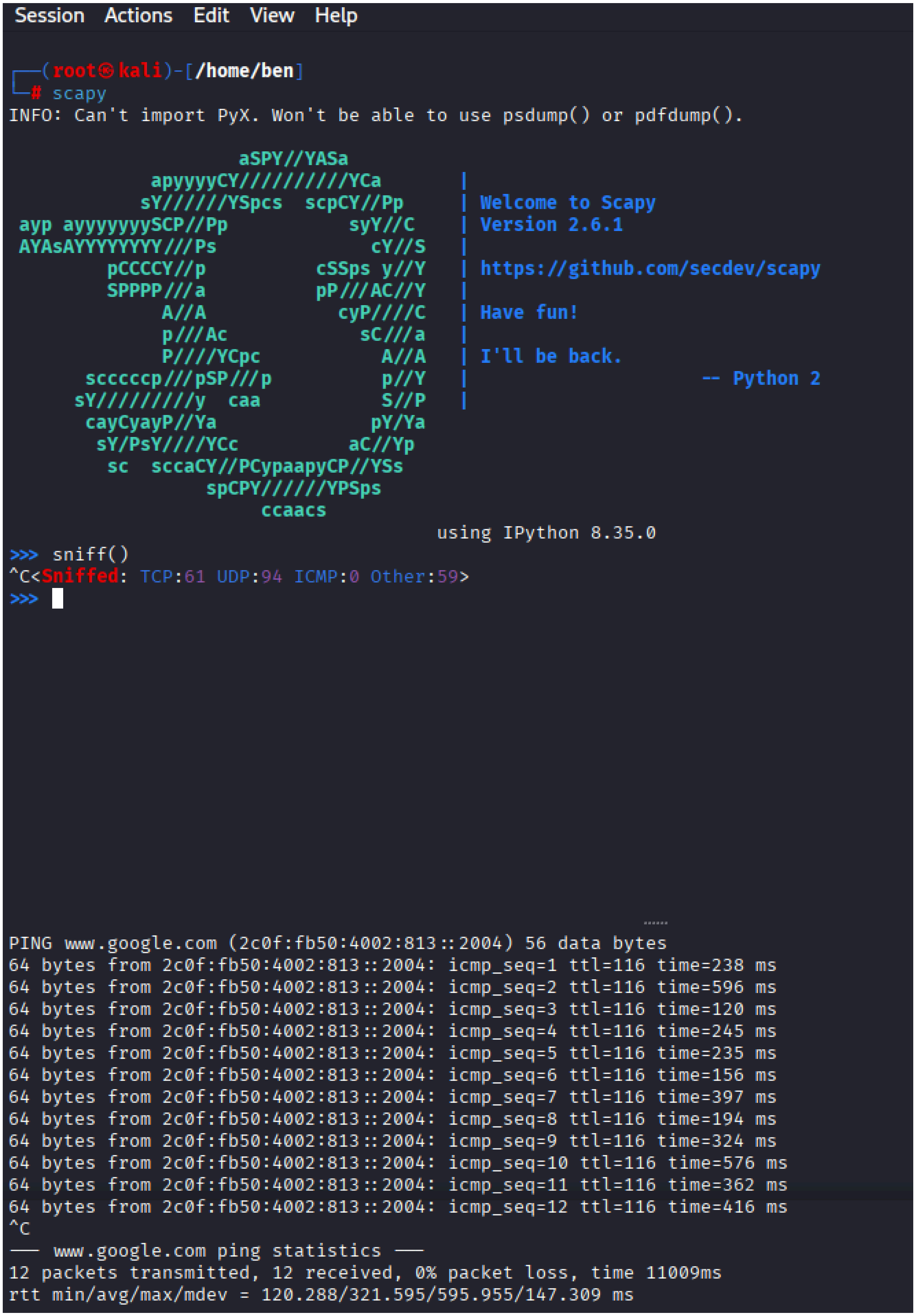


```
┌──(ben㉿kali)-[~]
└─$ sudo nmap -A 10.0.160.144
[sudo] password for ben:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-03 15:39 GMT
Nmap scan report for 10.0.160.144
Host is up (0.22s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE    SERVICE       VERSION
22/tcp    open     ssh           OpenSSH 8.4p1 Debian 5+deb11u2 (protocol 2.0)
| ssh-hostkey:
|_  256 fd:f4:47:5f:2a:05:8d:c4:d7:16:0d:12:3b:60:bb:ca (ECDSA)
80/tcp    open     http          Apache httpd 2.4.49 ((Unix))
1123/tcp  filtered murray
3071/tcp  filtered csd-mgmt-port
50500/tcp filtered unknown
Device type: general purpose|storage-misc|media device
Running (JUST GUESSING): Linux 3.X|4.X|2.6.X|5.X (91%), QNAP QTS 5.X (87%), HP embedded (86%), Amazon embedded (86%), Infomir embedded (86%)
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:5 cpe:/o:qnap:qts:5.0 cpe:/o:linux:linux_kernel:5.10 cpe:/h:hp:p2000_g3 cpe:/h:infomir:mag-250
Aggressive OS guesses: Linux 3.10 - 4.11 (91%), Linux 2.6.32 - 3.13 (90%), Linux 5.0 - 5.14 (90%), Linux 3.2 - 4.14 (89%), Linux 4.15 (87%), Linux 4.15 - 5.19 (87%), QNAP QTS 5.0 (Linux 5.10) (87%), Linux 3.13 - 4.4 (87%), Linux 4.19 -
5.15 (87%), Linux 2.6.32 - 3.10 (87%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 8080/tcp)
HOP RTT       ADDRESS
1   186.85 ms 10.10.0.1
2   186.90 ms 10.0.160.144

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 206.62 seconds

┌──(ben㉿kali)-[~]
└─$ 
```

In the diagram above the capital A that was used represents an aggressive scan meaning it will generate all the possible info about that ip address. Therefore there is no need to add the sC,sV and the others since the introduction of the capital A will generate all the information.

```
SCAPY USAGE
```

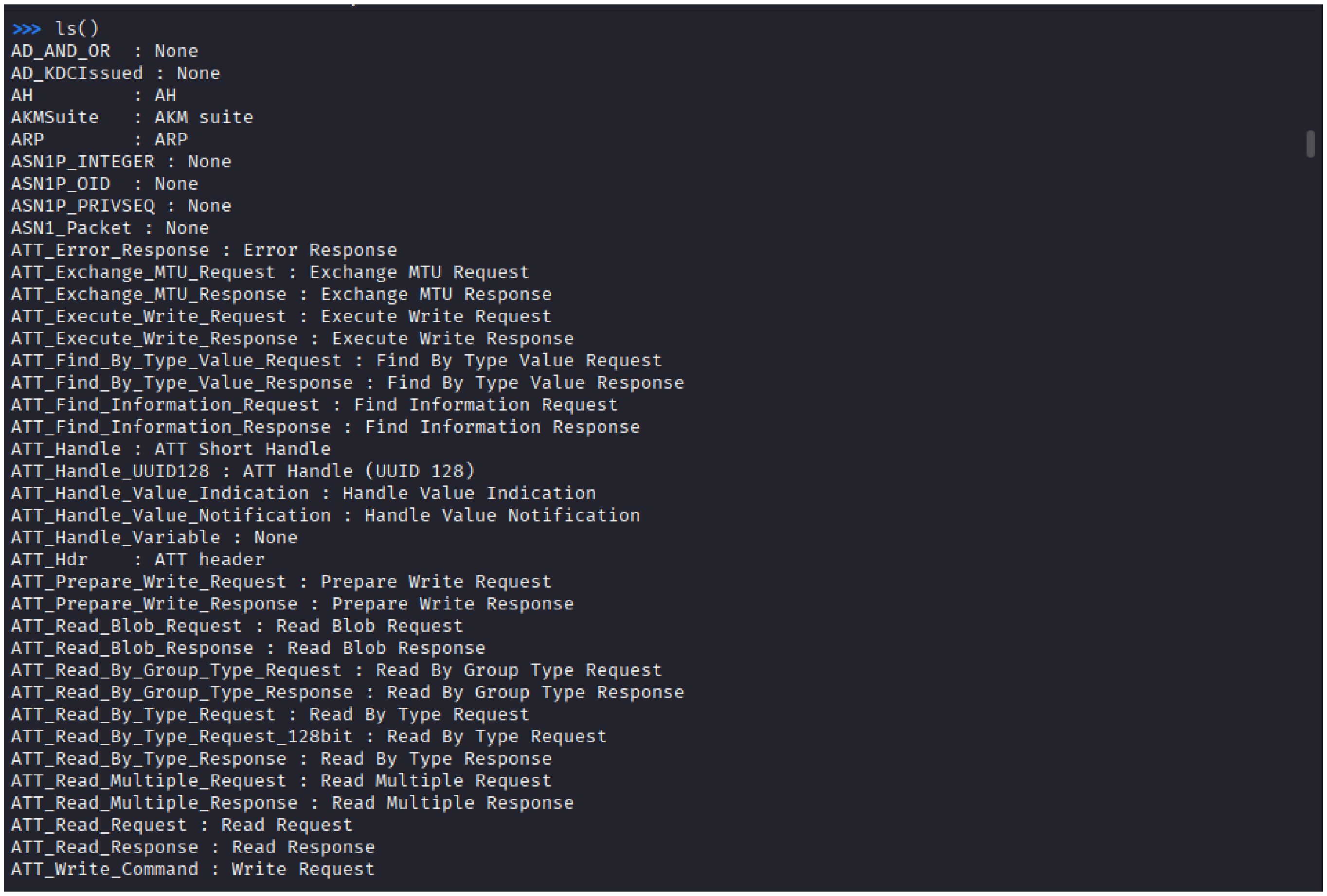
Scapy is a powerful Python-based interactive packet manipulation program and library. It is capable of forging or decoding packets of a wide number of protocols, sending them on the wire, capturing them, and much more. With Scapy, you can craft custom packets, sniff network traffic, perform network scans, and automate tasks that involve network interaction.



```
┌──(root㉿kali)-[/home/ben]
└─# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.6.1
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | I'll be back.
       scccccp///pSP///p          p//Y   |                   -- Python 2
      sY/////////y  caa           S//P   |
       cayCyayP//Ya              pY/Ya
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 8.35.0
>>> sniff()
^C<Sniffed: TCP:61 UDP:94 ICMP:0 Other:59>
>>> 
```

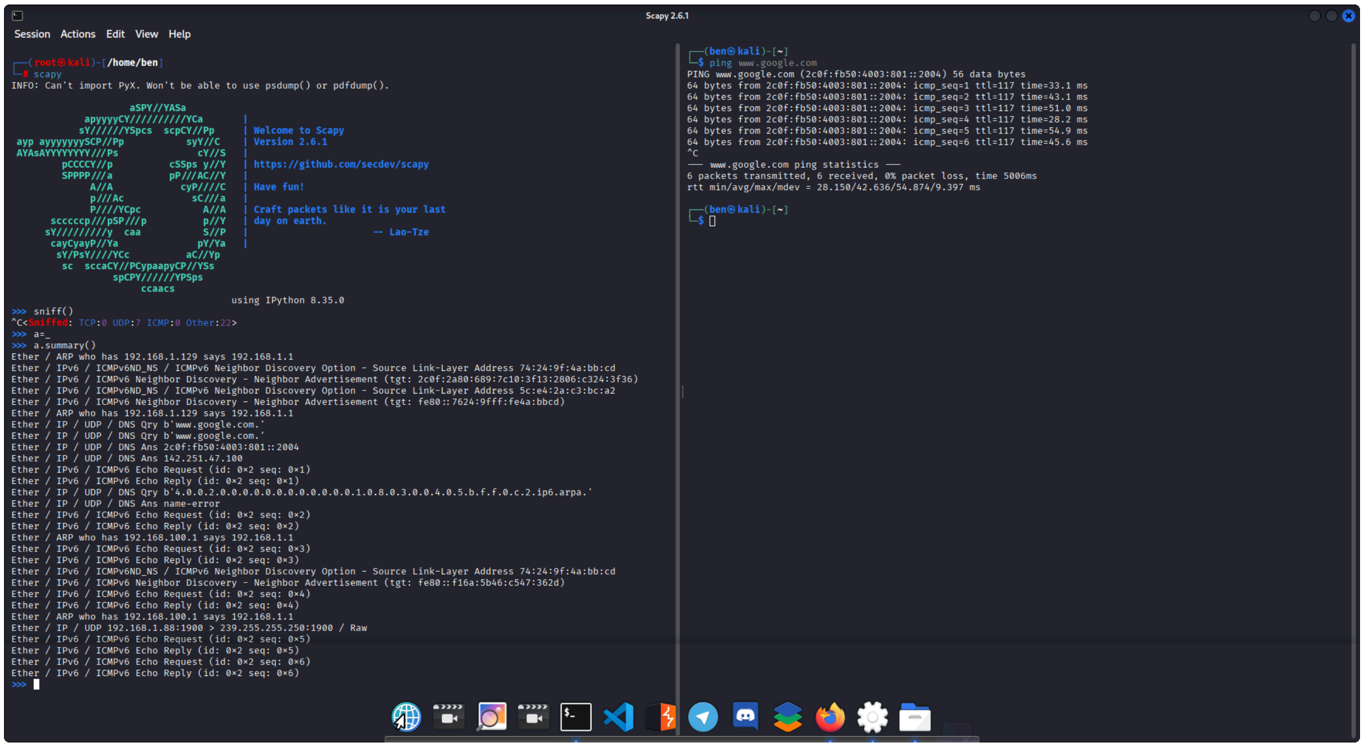
```
PING www.google.com (2c0f:fb50:4002:813::2004) 56 data bytes
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=1 ttl=116 time=238 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=2 ttl=116 time=596 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=3 ttl=116 time=120 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=4 ttl=116 time=245 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=5 ttl=116 time=235 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=6 ttl=116 time=156 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=7 ttl=116 time=397 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=8 ttl=116 time=194 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=9 ttl=116 time=324 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=10 ttl=116 time=576 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=11 ttl=116 time=362 ms
64 bytes from 2c0f:fb50:4002:813::2004: icmp_seq=12 ttl=116 time=416 ms
^C
--- www.google.com ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11009ms
rtt min/avg/max/mdev = 120.288/321.595/595.955/147.309 ms
```

To be able to start scapy u must have root privileges before. BY switching to the root shell you must type sudo su and then type your password.



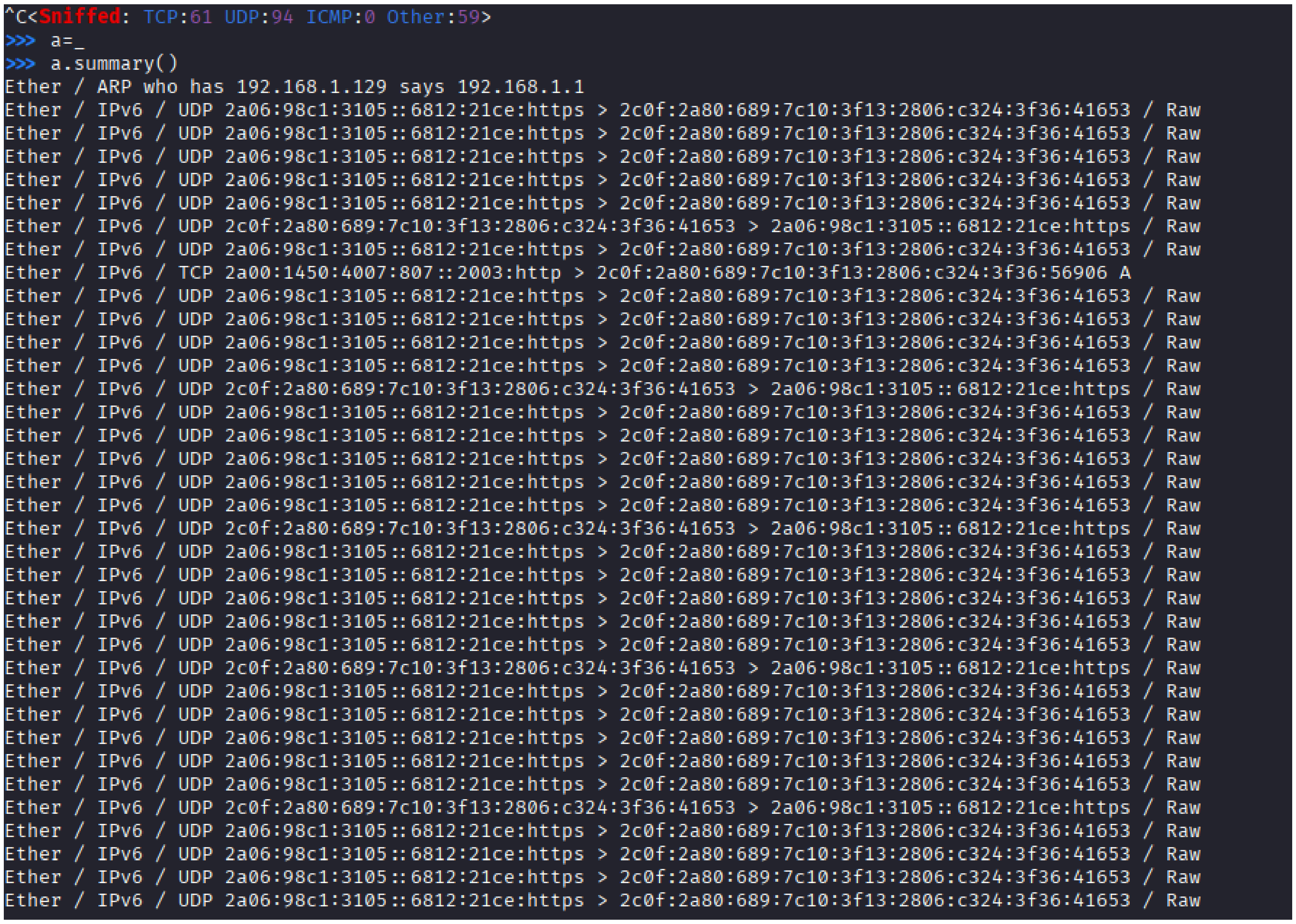
```
>>> ls()
AD_AND_OR   : None
AD_KDCIssued : None
AH          : AH
AKMSuite    : AKM suite
ARP         : ARP
ASN1P_INTEGER : None
ASN1P_OID   : None
ASN1P_PRIVSEQ : None
ASN1_Packet : None
ATT_Error_Response : Error Response
ATT_Exchange_MTU_Request : Exchange MTU Request
ATT_Exchange_MTU_Response : Exchange MTU Response
ATT_Execute_Write_Request : Execute Write Request
ATT_Execute_Write_Response : Execute Write Response
ATT_Find_By_Type_Value_Request : Find By Type Value Request
ATT_Find_By_Type_Value_Response : Find By Type Value Response
ATT_Find_Information_Request : Find Information Request
ATT_Find_Information_Response : Find Information Response
ATT_Handle : ATT Short Handle
ATT_Handle_UUID128 : ATT Handle (UUID 128)
ATT_Handle_Value_Indication : Handle Value Indication
ATT_Handle_Value_Notification : Handle Value Notification
ATT_Handle_Variable : None
ATT_Hdr     : ATT header
ATT_Prepare_Write_Request : Prepare Write Request
ATT_Prepare_Write_Response : Prepare Write Response
ATT_Read_Blob_Request : Read Blob Request
ATT_Read_Blob_Response : Read Blob Response
ATT_Read_By_Group_Type_Request : Read By Group Type Request
ATT_Read_By_Group_Type_Response : Read By Group Type Response
ATT_Read_By_Type_Request : Read By Type Request
ATT_Read_By_Type_Request_128bit : Read By Type Request
ATT_Read_By_Type_Response : Read By Type Response
ATT_Read_Multiple_Request : Read Multiple Request
ATT_Read_Multiple_Response : Read Multiple Response
ATT_Read_Request : Read Request
ATT_Read_Response : Read Response
ATT_Write_Command : Write Request
```

using ls () in the scapy shell allows you to view some basic things you can do with scapy.



```
┌──(root㉿kali)-[/home/ben]
└─# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.6.1
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft packets like it is your last
       scccccp///pSP///p          p//Y   | day on earth.
      sY/////////y  caa           S//P   |                      -- Lao-Tze
       cayCyayP//Ya              pY/Ya   |
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 8.35.0
>>> sniff()
^C<Sniffed: TCP:0 UDP:7 ICMP:0 Other:22>
>>> a=_
>>> a.summary()
Ether / ARP who has 192.168.1.129 says 192.168.1.1
Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 74:24:9f:4a:bb:cd
Ether / IPv6 / ICMPv6 Neighbor Discovery - Neighbor Advertisement (tgt: 2c0f:2a80:689:7c10:3f13:2806:c324:3f36)
Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 5c:e4:2a:c3:bc:a2
Ether / IPv6 / ICMPv6 Neighbor Discovery - Neighbor Advertisement (tgt: fe80::7624:9fff:fe4a:bbcd)
Ether / ARP who has 192.168.1.129 says 192.168.1.1
Ether / IP / UDP / DNS Qry b'www.google.com.'
Ether / IP / UDP / DNS Qry b'www.google.com.'
Ether / IP / UDP / DNS Ans 2c0f:fb50:4003:801::2004
Ether / IP / UDP / DNS Ans 142.251.47.100
Ether / IPv6 / ICMPv6 Echo Request (id: 0x2 seq: 0x1)
Ether / IPv6 / ICMPv6 Echo Reply (id: 0x2 seq: 0x1)
Ether / IP / UDP / DNS Qry b'4.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.1.0.8.0.3.0.0.4.0.5.b.f.f.0.c.2.ip6.arpa.'
Ether / IP / UDP / DNS Ans name-error
Ether / IPv6 / ICMPv6 Echo Request (id: 0x2 seq: 0x2)
Ether / IPv6 / ICMPv6 Echo Reply (id: 0x2 seq: 0x2)
Ether / ARP who has 192.168.100.1 says 192.168.1.1
Ether / IPv6 / ICMPv6 Echo Request (id: 0x2 seq: 0x3)
Ether / IPv6 / ICMPv6 Echo Reply (id: 0x2 seq: 0x3)
Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 74:24:9f:4a:bb:cd
Ether / IPv6 / ICMPv6 Neighbor Discovery - Neighbor Advertisement (tgt: fe80::f16a:5b46:c547:362d)
Ether / IPv6 / ICMPv6 Echo Request (id: 0x2 seq: 0x4)
Ether / IPv6 / ICMPv6 Echo Reply (id: 0x2 seq: 0x4)
Ether / ARP who has 192.168.100.1 says 192.168.1.1
Ether / IP / UDP 192.168.1.88:1900 > 239.255.255.250:1900 / Raw
Ether / IPv6 / ICMPv6 Echo Request (id: 0x2 seq: 0x5)
Ether / IPv6 / ICMPv6 Echo Reply (id: 0x2 seq: 0x5)
Ether / IPv6 / ICMPv6 Echo Request (id: 0x2 seq: 0x6)
Ether / IPv6 / ICMPv6 Echo Reply (id: 0x2 seq: 0x6)
>>>
```

```
┌──(ben㉿kali)-[~]
└─$ ping www.google.com
PING www.google.com (2c0f:fb50:4003:801::2004) 56 data bytes
64 bytes from 2c0f:fb50:4003:801::2004: icmp_seq=1 ttl=117 time=33.1 ms
64 bytes from 2c0f:fb50:4003:801::2004: icmp_seq=2 ttl=117 time=43.1 ms
64 bytes from 2c0f:fb50:4003:801::2004: icmp_seq=3 ttl=117 time=51.0 ms
64 bytes from 2c0f:fb50:4003:801::2004: icmp_seq=4 ttl=117 time=28.2 ms
64 bytes from 2c0f:fb50:4003:801::2004: icmp_seq=5 ttl=117 time=54.9 ms
64 bytes from 2c0f:fb50:4003:801::2004: icmp_seq=6 ttl=117 time=45.6 ms
^C
--- www.google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 28.150/42.636/54.874/9.397 ms

┌──(ben㉿kali)-[~]
└─$
```



```
^C<Sniffed: TCP:61 UDP:94 ICMP:0 Other:59>
>>> a=_
>>> a.summary()
Ether / ARP who has 192.168.1.129 says 192.168.1.1
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 > 2a06:98c1:3105::6812:21ce:https / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / TCP 2a00:1450:4007:807::2003:http > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:56906 A
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 > 2a06:98c1:3105::6812:21ce:https / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 > 2a06:98c1:3105::6812:21ce:https / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 > 2a06:98c1:3105::6812:21ce:https / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 > 2a06:98c1:3105::6812:21ce:https / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
Ether / IPv6 / UDP 2a06:98c1:3105::6812:21ce:https > 2c0f:2a80:689:7c10:3f13:2806:c324:3f36:41653 / Raw
```

From the two images above u can see that i was able to sniff packets after ping google.com and then typing sniff() in the interactive shell by scapy. After sniffing the packets u have to be able to

view the kind of packet you sniffed, so by doing this u have to assign a variable to the packets.In the second image you can see that the packets were assigned to a variable a and then viewed by typing a.summary()