

Vikash Kumar Deo

N15708475

Vkd225

Network Security (CS 6823)

Lab -2

Due: 09/24/2015

NMAP Scan

Nmap command to find the required details

Command: `nmap -sV -O 10.10.111.0/24 -o nmapscan.txt`

Options:

- p- is to scan all the ports from 1-65535. (It does take some time but scans all the port).
- A turns on Advanced and Aggressive feature such as Operating System and service detection.
- o is to write the contents into a file. (File can be downloaded using sftp protocol).

Using nmap scan we can see all the open ports and the corresponding operating systems they are running. When nmap command is used, probes are sent and responses to this probe are used to classify the ports into three categories: either open, closed or filtered.

There are six machines running.

IP Address: 10.10.111.1

Open Ports: 53/tcp, 111/tcp

OS: Linux 2.6.X

```
root@bt:~# nmap -p- -A 10.10.111.0/24 -o nmapscan.txt
Starting Nmap 5.51 ( http://nmap.org ) at 2015-09-24 21:41 EDT
Nmap scan report for 10.10.111.1
Host is up (0.0017s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
53/tcp    open  domain ISC BIND 9.5.1-P3
111/tcp   open  rpcbind 2 (rpc #100000)
MAC Address: 02:00:58:5F:08:02 (Unknown)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.24
Network Distance: 1 hop

TRACEROUTE
HOP RTT ADDRESS
1 1.73 ms 10.10.111.1
```

IP Address: 10.10.111.2

Open Ports: 53/tcp, 111/tcp

OS: Linux 2.6.X

```
Nmap scan report for 10.10.111.2
Host is up (0.0017s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
53/tcp    open  domain ISC BIND 9.5.1-P3
111/tcp   open  rpcbind 2 (rpc #100000)
MAC Address: 02:00:58:D0:00:01 (Unknown)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.24
Network Distance: 1 hop
```

TRACEROUTE

HOP	RTT	ADDRESS
1	1.74 ms	10.10.111.2

IP Address: 10.10.111.106

Open Ports: 22/tcp

OS: Linux 2.6.X

```
Nmap scan report for 10.10.111.106
Host is up (0.0019s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.1p1 Debian 5 (protocol 2.0)
| ssh-hostkey: 1024 af:0d:05:e1:87:a5:53:c8:5c:aa:ff:50:7c:3d:16:c6 (DSA)
| 2048 c3:7d:70:74:f8:0d:d4:16:47:39:0b:41:10:0a:2f:46 (RSA)
111/tcp   open  rpcbind 2 (rpc #100000)
MAC Address: 02:00:58:97:0C:01 (Unknown)
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
```

```
OS:SCAN(V=5.51%D=9/24%OT=22%CT=1%CU=38135%PV=Y%D=1%DC=D%G=Y%M=020058%TM=56
OS:04A728%P=i686-pc-linux-gnu)SEQ(SP=CA%GCD=1%ISR=CB%TI=Z%CI=Z%II=I%TS=9)SE
OS:Q(SP=CA%GCD=2%ISR=CB%TI=Z%CI=Z%II=I%TS=9)OPS(O1=M5B4ST11NW6%O2=M5B4ST11N
OS:W6%O3=M5B4NNT11NW6%O4=M5B4ST11NW6%O5=M5B4ST11NW6%O6=M5B4ST11)WIN(W1=16A0
OS:%W2=16A0%W3=16A0%W4=16A0%W5=16A0%W6=16A0)ECN(R=Y%DF=Y%T=40%W=16D0%O=M5B4
OS:NNSNW6%CC=N%Q=)T1(R=Y%DF=Y%T=40%S=0%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y
OS:%T=40%W=16A0%S=0%A=S+%F=AS%O=M5B4ST11NW6%RD=0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=
OS:A%A=Z%F=R%O=0%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=0%RD=0%Q=)T6(R=
OS:Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=0%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=A
OS:R%O=0%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%R
OS:UD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

Network Distance: 1 hop
Service Info: OS: Linux

TRACEROUTE

HOP	RTT	ADDRESS
1	1.90 ms	10.10.111.106

IP Address: 10.10.111.107
Open Ports: 111/tcp, 54615/tcp
OS: Linux 2.6.X

```
Nmap scan report for 10.10.111.107
Host is up (0.0046s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
111/tcp    open  rpcbind 2 (rpc #1000000)
54615/tcp  open  status  1 (rpc #100024)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.19 - 2.6.36
Network Distance: 0 hops
```

IP Address: 10.10.111.109
Open Ports: 135/tcp, 139/tcp, 445/tcp, 1025/tcp, 5000/tcp
OS: Microsoft Windows 2000|XP

```
Nmap scan report for 10.10.111.109
Host is up (0.0028s latency).
Not shown: 65530 closed ports
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows XP microsoft-ds
445/tcp    open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp   open  msrpc        Microsoft Windows RPC
5000/tcp   open  upnp         Microsoft Windows UPnP
MAC Address: 02:00:58:7B:0A:01 (Unknown)
Device type: general purpose
Running: Microsoft Windows 2000|XP
OS details: Microsoft Windows 2000 SP0/SP1/SP2 or Windows XP SP0/SP1
Network Distance: 1 hop
Service Info: OS: Windows

Host script results:
|_ nbstat: NetBIOS name: VICTIM1, NetBIOS user: POLY, NetBIOS MAC: 02:00:58:7b:0a:01 (unknown)
|_ smb2-enabled: Server doesn't support SMBv2 protocol
|_ smb-os-discovery:
|   OS: Windows XP (Windows 2000 LAN Manager)
|   Name: WORKGROUP\VICTIM1
|_ System time: 2015-09-25 04:47:40 UTC-7

TRACEROUTE
HOP RTT      ADDRESS
1   2.80 ms  10.10.111.109
```

IP Address: 10.10.111.110
Open Ports: 631/tcp, 3306/tcp
OS: Linux 2.6.x

```
Nmap scan report for 10.10.111.110
Host is up (0.0016s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
631/tcp   open ipp      CUPS 1.1
| http-methods: Potentially risky methods: PUT
|_ See http://nmap.org/nsedoc/scripts/http-methods.html
3306/tcp  open  mysql    MySQL (unauthorized)
MAC Address: 02:00:58:EC:02:01 (Unknown)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.31
Network Distance: 1 hop
```

TRACEROUTE

```
HOP RTT      ADDRESS
1   1.57 ms  10.10.111.110
```

OS and Service detection performed. Please report any incorrect results at <http://nmap.org/submit/>.

Nmap done: 256 IP addresses (6 hosts up) scanned in 341.29 seconds

Nessus Vulnerability Scan

Starting Nessus:



Nessus opening in Firefox:



Nessus scan for XP Machine (IP address: 10.10.111.109)

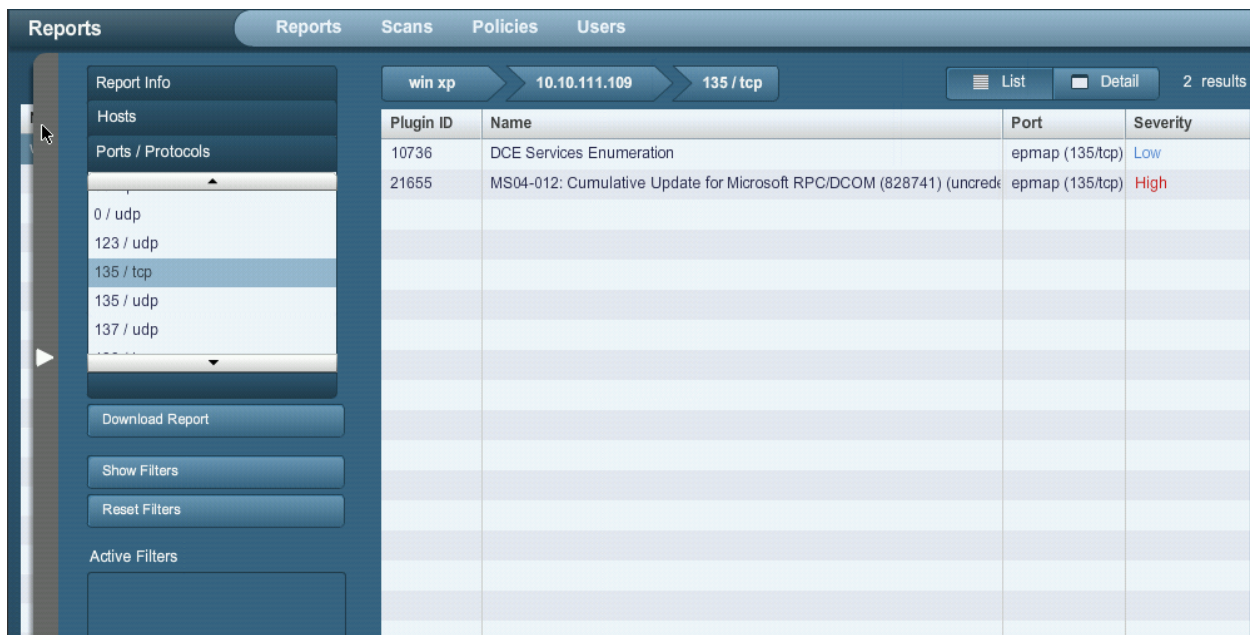
The picture shows the vulnerabilities present in the win XP machine (10.10.111.109) associated with each port. The vulnerabilities are present as severity of 'High', 'Medium', 'Low' and 'Open Ports'.



The screenshot shows the Nessus Reports page for host 10.10.111.109. The left sidebar contains 'Report Info', 'Hosts', and 'Active Filters'. The main table displays 13 results with columns for Port, Protocol, SVC Name, Total, High, Medium, Low, and Open Port. The data is as follows:

Port	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	icmp	general	1	0	0	1	0
0	tcp	general	6	1	0	5	0
0	udp	general	1	0	0	1	0
123	udp	ntp	1	0	0	1	0
135	tcp	epmap	3	1	0	1	1
135	udp	epmap?	1	1	0	0	0
137	udp	netbios-ns	1	0	0	1	0
139	tcp	smb	2	0	0	1	1
445	tcp	cifs	19	10	1	7	1
1025	tcp	dce-rpc	3	1	0	1	1
1027	udp	dce-rpc	1	0	0	1	0
1900	udp	upnp-client	1	0	0	1	0
5000	tcp	www	5	0	0	4	1

The picture shows the vulnerabilities associated with port 135 and it has 1 'Low' type severity and 1 'High' type severity and 1 open port present.



The screenshot shows the Nessus Reports page with the filter '135 / tcp' applied. The left sidebar shows 'Ports / Protocols' with a dropdown menu. The main table displays 2 results with columns for Plugin ID, Name, Port, and Severity. The data is as follows:

Plugin ID	Name	Port	Severity
10736	DCE Services Enumeration	epmap (135/tcp)	Low
21655	MS04-012: Cumulative Update for Microsoft RPC/DCOM (828741) (uncred	epmap (135/tcp)	High

Further looking into the vulnerability, we can see the synopsis, Description and Solution present for the vulnerability. For an example, we are looking at the high type vulnerability.

The screenshot shows the Nessus Reports interface. The left sidebar contains a 'Ports / Protocols' list with '135 / tcp' selected. The main panel displays details for a vulnerability on host '10.10.111.109' at port '135 / tcp'. The vulnerability is identified as 'Plugin ID: 21655' and 'Plugin Name: MS04-012: Cumulative Update for Microsoft RPC/DCOM (828741) (unauthenticated check)'. The severity is 'High'. The synopsis states: 'Arbitrary code can be executed on the remote host.' The description mentions: 'The remote host has multiple bugs in its RPC/DCOM implementation (828741). An attacker may exploit one of these flaws to execute arbitrary code on the remote system.' The solution notes: 'Microsoft has released a set of patches for Windows NT, 2000, XP and 2003 : <http://www.microsoft.com/technet/security/bulletin/ms04-012.msp>'. The risk factor is 'Critical' and the CVSS Base Score is '10.0 (CVSS2#AV:N/AC:L/Au:N/C/I:C/A:C)'.

For another example, we can see the vulnerabilities associated with port 445 and it has 10 'High' type, 1 'Medium' type and 7 'Low' type severity and 1 open port present.

The screenshot shows the Nessus Reports interface with a list of vulnerabilities on host '10.10.111.109' at port '445 / tcp'. The left sidebar shows '445 / tcp' selected in the 'Ports / Protocols' list. The main panel displays a table of 18 results.

Plugin ID	Name	Port	Severity
11011	Microsoft Windows SMB Service Detection	cifs (445/tcp)	Low
10736	DCE Services Enumeration	cifs (445/tcp)	Low
10785	Microsoft Windows SMB NativeLanManager Remote System Information D	cifs (445/tcp)	Low
10394	Microsoft Windows SMB Log In Possible	cifs (445/tcp)	Low
22034	MS06-035: Vulnerability in Server Service Could Allow Remote Code Execu	cifs (445/tcp)	High
26917	Microsoft Windows SMB Registry : Nessus Cannot Access the Windows Re	cifs (445/tcp)	Low
22194	MS06-040: Vulnerability in Server Service Could Allow Remote Code Execu	cifs (445/tcp)	High
10395	Microsoft Windows SMB Shares Enumeration	cifs (445/tcp)	Low
35362	MS09-001: Microsoft Windows SMB Vulnerabilities Remote Code Execution	cifs (445/tcp)	High
26920	Microsoft Windows SMB NULL Session Authentication	cifs (445/tcp)	Low
19407	MS05-043: Vulnerability in Printer Spooler Service Could Allow Remote Cod	cifs (445/tcp)	High
18502	MS05-027: Vulnerability in SMB Could Allow Remote Code Execution (8964	cifs (445/tcp)	High
16337	MS05-007: Vulnerability in Windows Could Allow Information Disclosure (88	cifs (445/tcp)	Medium
11835	MS03-039: Microsoft RPC Interface Buffer Overrun (824146) (uncredential	cifs (445/tcp)	High
12209	MS04-011: Security Update for Microsoft Windows (835732) (uncredential	cifs (445/tcp)	High
12054	MS04-007: ASN.1 Vulnerability Could Allow Code Execution (828028) (uncr	cifs (445/tcp)	High
11110	MS02-045: Microsoft Windows SMB Protocol SMB_COM_TRANSACTION E	cifs (445/tcp)	High

Further looking into the vulnerability, we can see the synopsis, Description and Solution present for the vulnerability. For an example, we are looking at the high type and a low type vulnerability associated with port 445.

The screenshot shows the Nessus interface with the 'Reports' tab selected. The left sidebar contains a 'Ports / Protocols' list with '445 / tcp' selected. The main panel displays details for a vulnerability with the following information:

- Plugin ID:** 10395
- Port / Service:** cifs (445/tcp)
- Severity:** Low
- Plugin Name:** Microsoft Windows SMB Shares Enumeration
- Synopsis:** It is possible to enumerate remote network shares.
- Description:** By connecting to the remote host, Nessus was able to enumerate the network share names.
- Solution:** N/A
- Risk Factor:** None
- Plugin Output:** Here are the SMB shares available on the remote host when logged as a NULL session:
 - IPC\$
 - ADMIN\$
 - C\$
- Plugin Publication Date:** 2000/05/09

The screenshot shows the Nessus interface with the 'Reports' tab selected. The left sidebar contains a 'Ports / Protocols' list with '445 / tcp' selected. The main panel displays details for a vulnerability with the following information:

- Plugin ID:** 22034
- Port / Service:** cifs (445/tcp)
- Severity:** High
- Plugin Name:** MS06-035: Vulnerability in Server Service Could Allow Remote Code Execution (917159) (uncred...
- Synopsis:** Arbitrary code can be executed on the remote host due to a flaw in the 'Server' service.
- Description:** The remote host is vulnerable to heap overflow in the 'Server' service that may allow an attacker to execute arbitrary code on the remote host with 'SYSTEM' privileges. In addition to this, the remote host is also affected by an information disclosure vulnerability in SMB that may allow an attacker to obtain portions of the memory of the remote host.
- Solution:** Microsoft has released a set of patches for Windows 2000, XP and 2003 :
<http://www.microsoft.com/technet/security/bulletin/ms06-035.msp>
- Risk Factor:** High

Python/Scapy programming

1

a: - Built a TCP packet with no payload and encapsulated the packet in IP packet and called it as a 'layer2'.

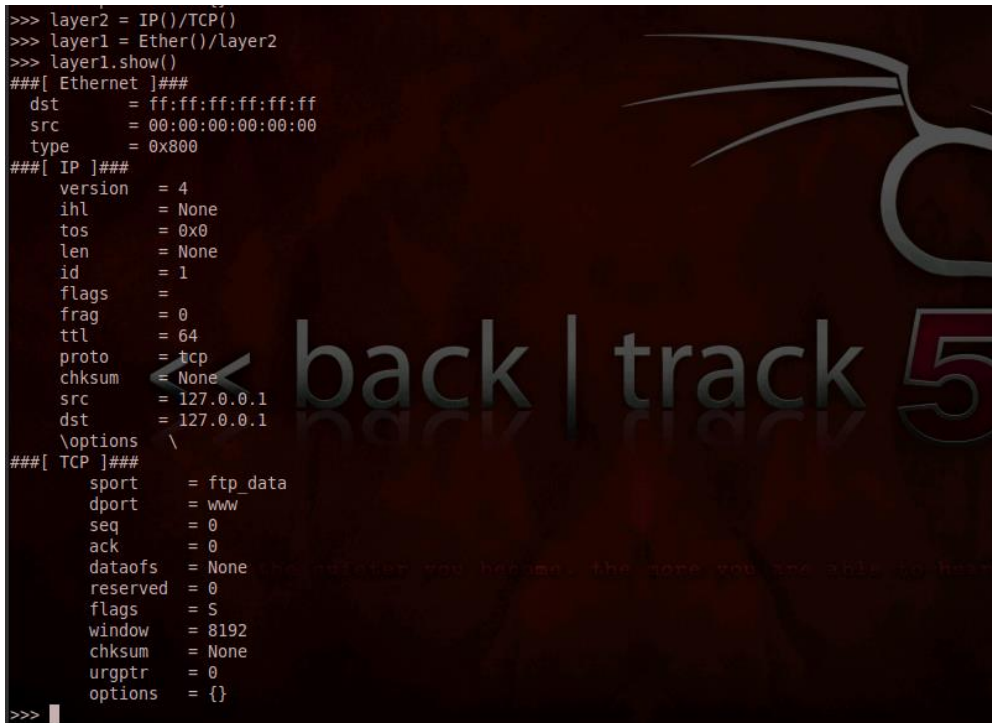
layer2= IP()/TCP()

- Encapsulated the IP packet (layer2) into an Ethernet packet and called it as 'layer1'.

layer1= Ether()/layer2

layer1.show()

Showing layer1 shows a TCP packet encapsulated in an IP packet (layer2), which is further encapsulated in an Ethernet packet and shows destination and source addresses (MAC and IP addresses).



```
>>> layer2 = IP()/TCP()
>>> layer1 = Ether()/layer2
>>> layer1.show()
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 00:00:00:00:00:00
  type     = 0x800
###[ IP ]###
  version  = 4
  ihl      = None
  tos      = 0x0
  len      = None
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = tcp
  chksum   = None
  src      = 127.0.0.1
  dst      = 127.0.0.1
  \options = \
###[ TCP ]###
  sport    = ftp_data
  dport    = www
  seq      = 0
  ack      = 0
  dataofs  = None
  reserved = 0
  flags    = S
  window   = 8192
  chksum   = None
  urgptr   = 0
  options  = {}
>>>
```

b: - Built a UDP packet with no payload and encapsulated the packet in IP packet and called it as a 'layer2'.

layer2= IP()/UDP()

- Encapsulated the IP packet (layer2) into an Ethernet packet and called it as 'layer1'.

layer1= Ether()/layer2

layer1.show()

Showing layer1 shows a UDP packet encapsulated in an IP packet (layer2), which is further encapsulated in an Ethernet packet and shows destination and source addresses (MAC and IP addresses).

```

>>> layer2 = IP()/UDP()
>>> layer1 = Ether()/layer2
>>> layer1.show()
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 00:00:00:00:00:00
  type     = 0x800
###[ IP ]###
  version  = 4
  ihl      = None
  tos      = 0x0
  len      = None
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = udp
  chksum   = None
  src      = 127.0.0.1
  dst      = 127.0.0.1
  \options \
###[ UDP ]###
  sport    = domain
  dport    = domain
  len      = None
  chksum   = None
>>>

```

2: Set of packets for IP address 10.20.111.109/30 and subnet and port number 80 and 53 are generated and sent.

Pck = IP(dst = "10.20.111.109/24")
pckport = TCP (dport = [80,53])
[k for k in pck/port]

```

>>> pck = IP(dst = "10.10.111.109/30")
>>> pckport = TCP (dport = [53,80])
>>> [k for k in pck/pckport]
[<IP frag=0 proto=tcp dst=10.10.111.108 |<TCP dport=domain |>>, <IP frag=0 proto=tcp dst=10.10.111.108 |<TCP dport=www |>>, <IP frag=0 proto=tcp dst=10.10.111.109 |<TCP dport=domain |>>, <IP frag=0 proto=tcp dst=10.10.111.109 |<TCP dport=www |>>, <IP frag=0 proto=tcp dst=10.10.111.110 |<TCP dport=domain |>>, <IP frag=0 proto=tcp dst=10.10.111.110 |<TCP dport=www |>>, <IP frag=0 proto=tcp dst=10.10.111.111 |<TCP dport=domain |>>, <IP frag=0 proto=tcp dst=10.10.111.111 |<TCP dport=www |>>]
>>>

```

pckport.show()

```

>>> pckport.show()
###[ TCP ]###
  sport    = ftp_data
  dport    = ['domain', 'www']
  seq      = 0
  ack      = 0
  dataofs  = None
  reserved = 0
  flags    = S
  window   = 8192
  chksum   = None
  urgptr   = 0
  options  = {}
>>>

```

The destination port for 53 is 'domain' (Domain Name System) and 80 is www (HTTP).

3:

An ICMP packet is generated and is called PACKET with a destination address 10.10.111.109 (Windows XP machine) and is encapsulated in IP layer packet.

PACKET = IP (dst="10.10.111.109")/ICMP()

The PACKET was sent and received using sr1(PACKET) function which is for sending packets and receiving answers. It receives only one packet that answered the sent packet.

SENREC = sr1(PACKET)

```
>>> PACKET = IP (dst="10.10.111.109")/ICMP()
>>> SENREC = sr1(PACKET)
Begin emission:
Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
>>>
```

Pck = sr(IP(dst = "10.10.11.109")/ICMP())

ans, unans = sr(IP(dst = "10.10.11.109")/ICMP())

ans.summary() shows the source IP and destination IP address for ICMP request and reply.

```
>>> pck = sr(IP(dst= "10.10.111.109")/ICMP())
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> ans,unans = sr(IP(dst= "10.10.111.109")/ICMP())
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> ans.summary()
IP / ICMP 10.10.111.107 > 10.10.111.109 echo-request 0 ==> IP / ICMP 10.10.111.109 > 10.10.111.107 echo-reply 0 / Padding
>>>
```


4. Traceroute without built in command for traceroute

```
ans, unans = sr(IP(dst = "10.10.111.2", ttl=(4,25), id = RandShort())/TCP(flags=0x2))
```

```
>>> ans,unans = sr(IP(dst= "10.10.111.2", ttl = (4,25),id = RandShort())/TCP(flags=0x2))
Begin emission:
.*****.*****Finished to send 22 packets.
*
Received 24 packets, got 22 answers, remaining 0 packets
```

```
for snd, rcv in ans:
```

```
print snd.ttl, rcv.src, isinstance(rcv.payload, TCP)
```

```
>>> for snd,rcv in ans:
...     print snd.ttl, rcv.src, isinstance(rcv.payload, TCP)
...
4 10.10.111.2 True
5 10.10.111.2 True
6 10.10.111.2 True
7 10.10.111.2 True
8 10.10.111.2 True
9 10.10.111.2 True
10 10.10.111.2 True
11 10.10.111.2 True
12 10.10.111.2 True
13 10.10.111.2 True
14 10.10.111.2 True
15 10.10.111.2 True
16 10.10.111.2 True
17 10.10.111.2 True
18 10.10.111.2 True
19 10.10.111.2 True
20 10.10.111.2 True
21 10.10.111.2 True
22 10.10.111.2 True
23 10.10.111.2 True
24 10.10.111.2 True
25 10.10.111.2 True
>>> 
```