ADVANCED IDLE SCANNING

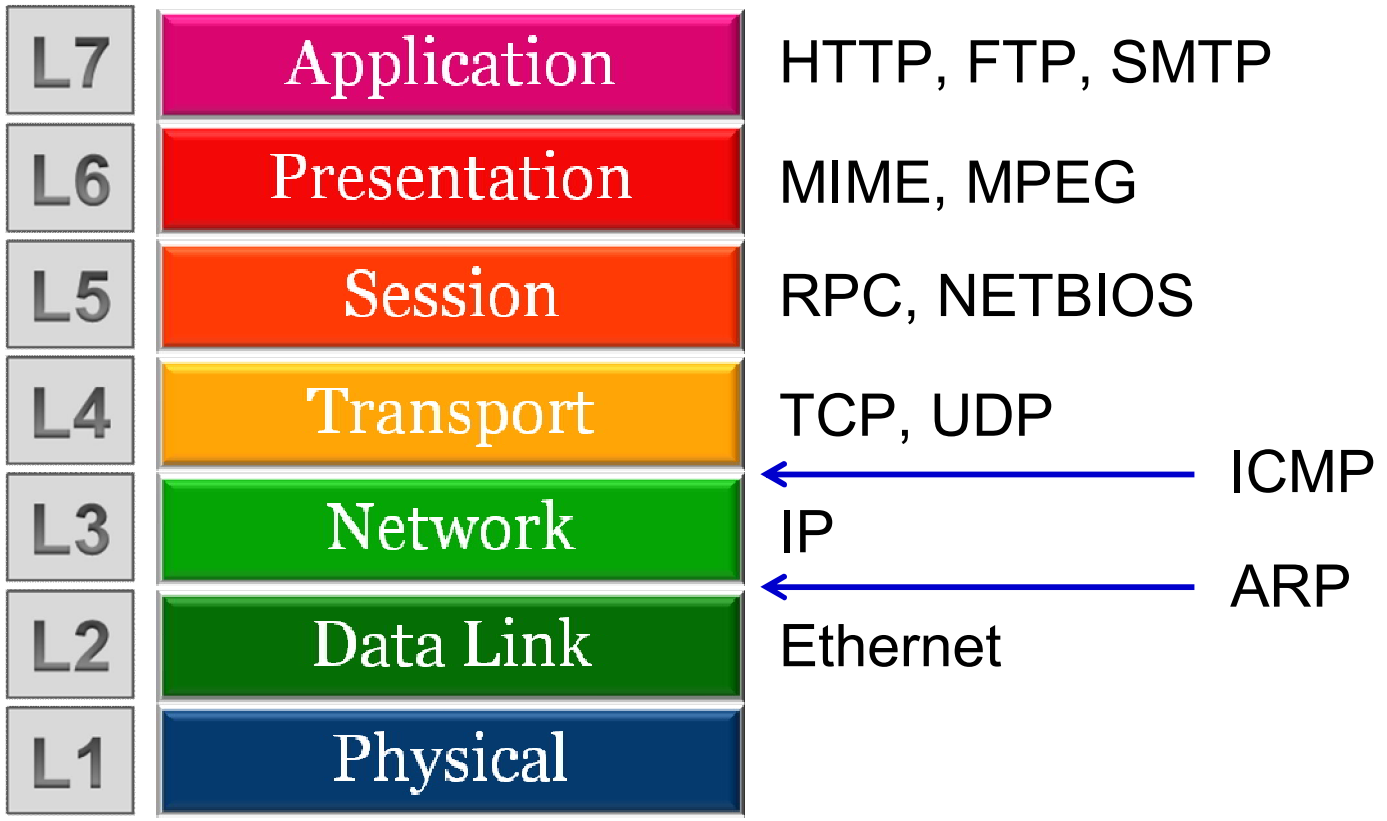
by Demetris Papapetrou

Contents

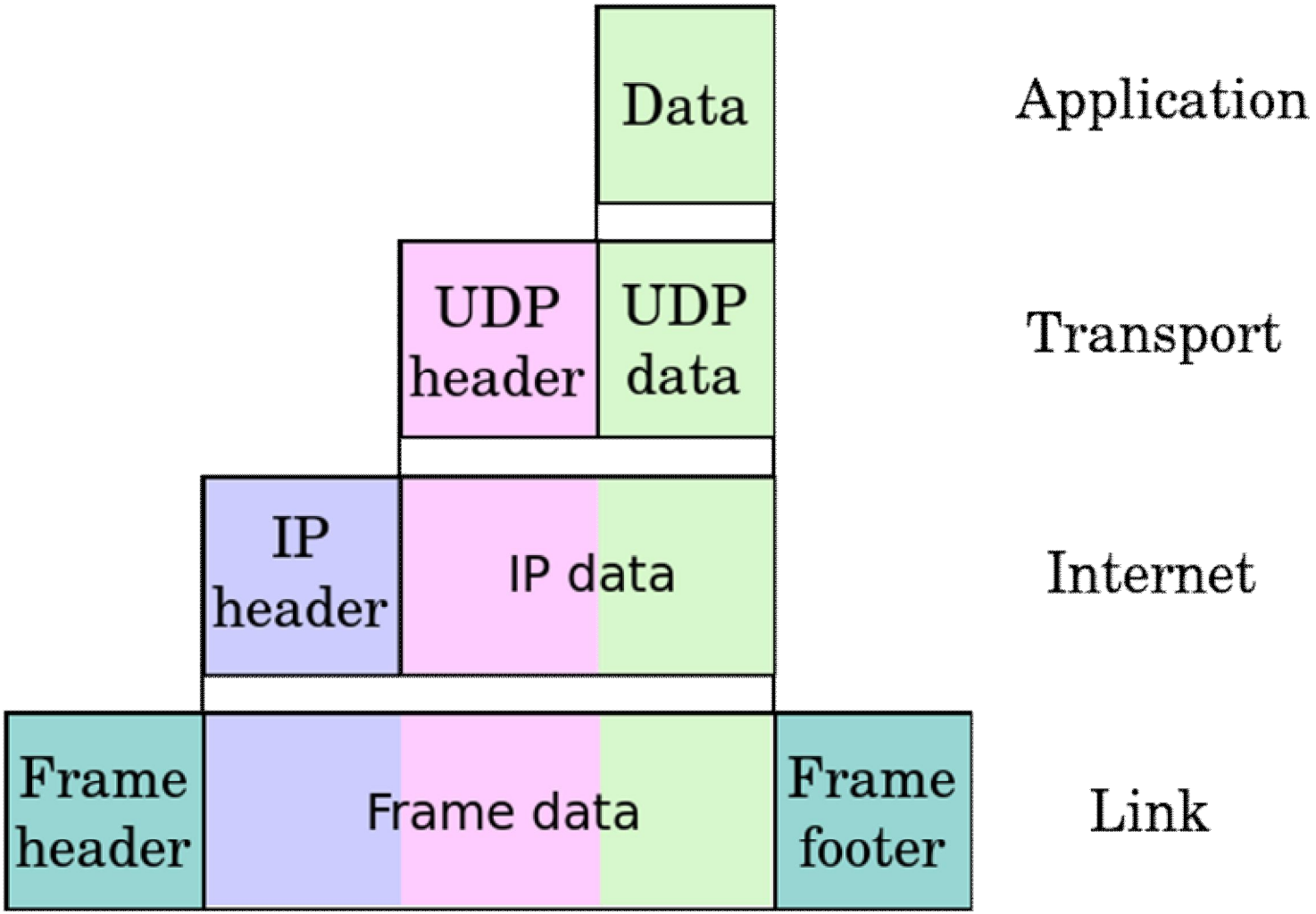
- Introduction to TCP/IP Networks
- Port Scanning Techniques
- Idle Scanning
- Advanced Idle Scanning
- Revealing Trust Relationships

INTRODUCTION TO TCP/IP NETWORKS

OSI 7 Layer Model



Encapsulation



IP Header

IP Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

TCP Header

TCP Header Format

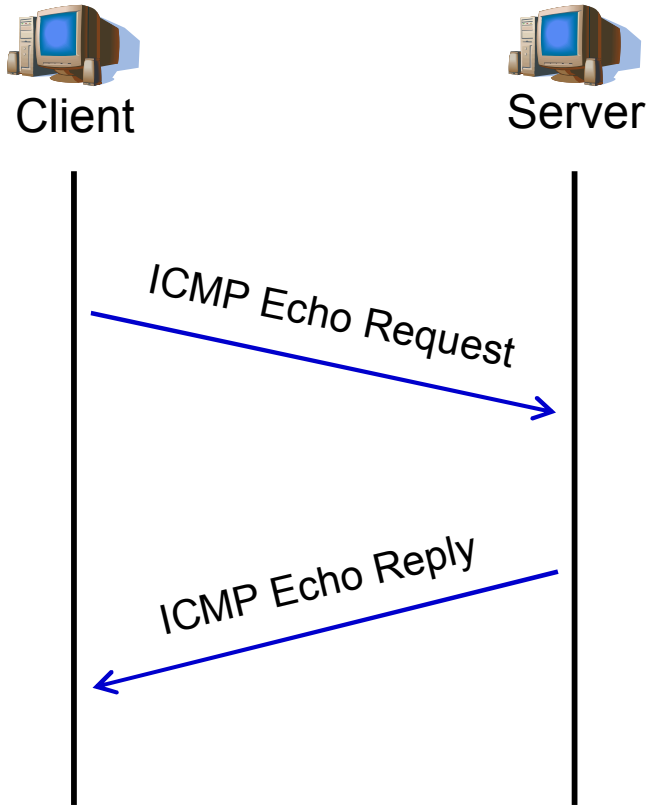
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Ping Sweep

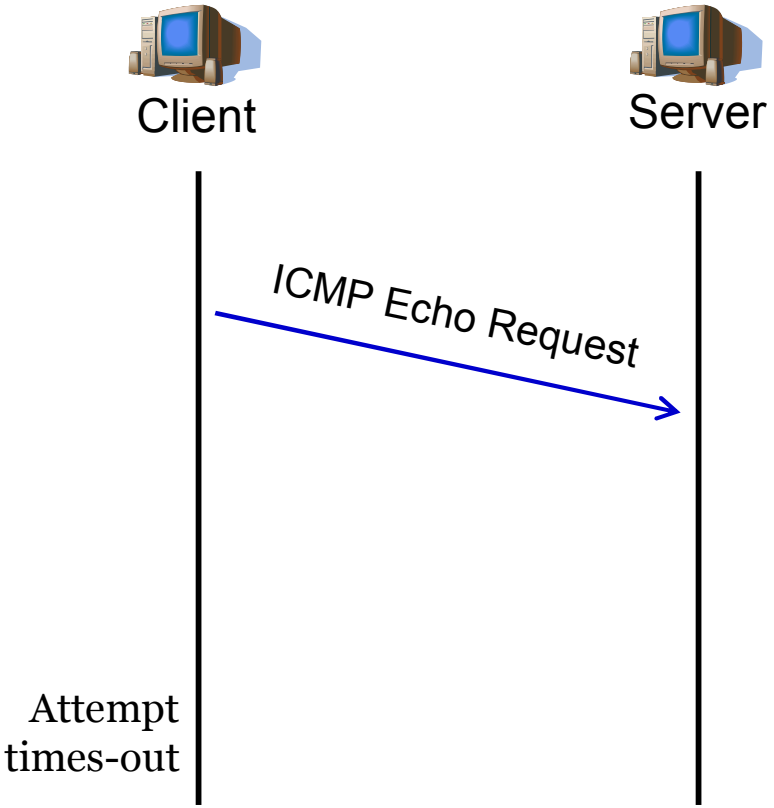
- It is a technique to detect live hosts
- Pings are not 100% reliable
  - On LANs we can use ARP pings
- They use ICMP Echo Requests and Echo Replies
- They can be blocked by firewalls very easily

Ping Sweep

Host is up and replies to Pings



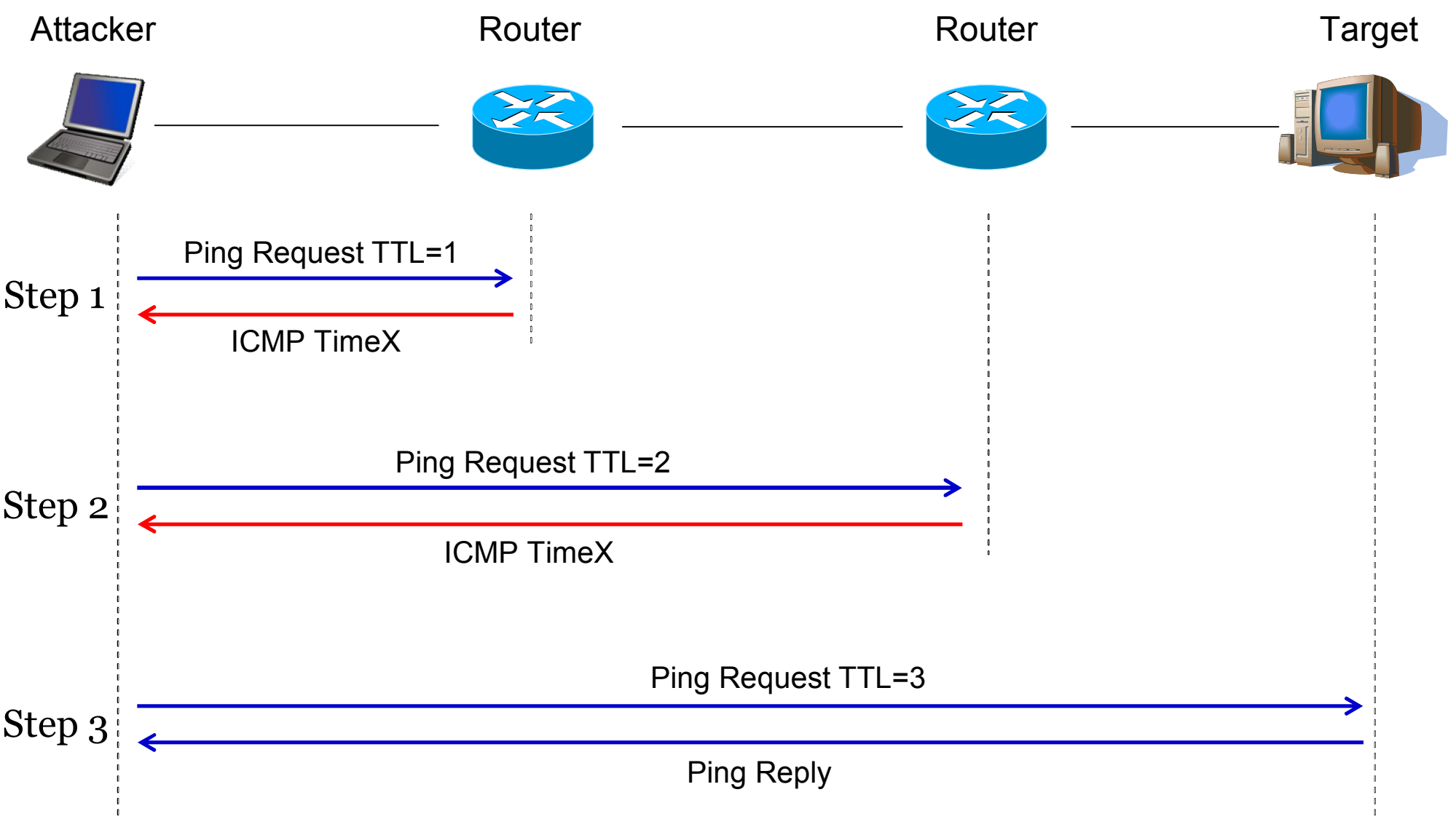
Host is down or Pings are blocked by a Firewall



Traceroute

- It is a technique to trace the network nodes/hops that our packets traverse until they reach their destination
- A clever replacement of IP Record Route
- We usually send ICMP or UDP packets
  - Hackers use TCP as well
- And receive ICMP Time Exceeded in Transit messages
- They are used by hackers to map the target network
- To block them drop outbound ICMP Time Exceeded

Traceroute


Attacker
Router
Router
Target
Step 1
Ping Request TTL=1
ICMP TimeX
Step 2
Ping Request TTL=2
ICMP TimeX
Step 3
Ping Request TTL=3
Ping Reply

Sample Traceroute Output

Command Prompt

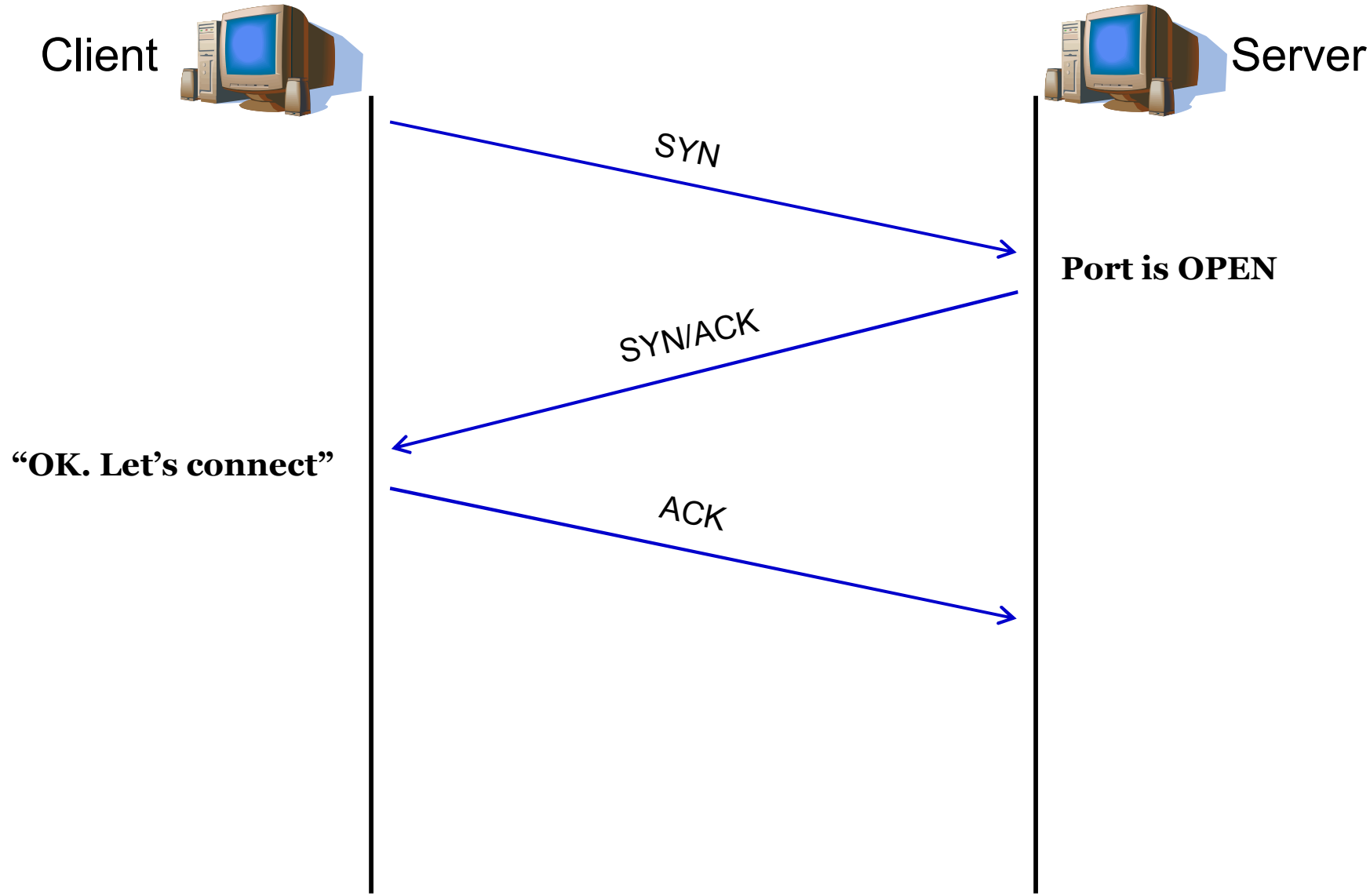
```
C:\>tracert mediacollege.com

Tracing route to mediacollege.com [66.246.3.197]
over a maximum of 30 hops:

  1    <10 ms    <10 ms    <10 ms  192.168.1.1
  2    240 ms    421 ms     70 ms  219-88-164-1.jetstream.xtra.co.nz [219.88.164.1]
  3     20 ms     30 ms     30 ms  210.55.205.123
  4      *         *         *     Request timed out.
  5     30 ms     30 ms     40 ms  202.50.245.197
  6     30 ms     40 ms     40 ms  g2-0-3.tkbr3.global-gateway.net.nz [202.37.245.140]
  7     30 ms     30 ms     40 ms  so-1-2-1-0.akbr3.global-gateway.net.nz [202.50.116.161]
  8    160 ms    161 ms    160 ms  p1-3.sjbr1.global-gateway.net.nz [202.50.116.178]
  9    160 ms    171 ms    160 ms  so-1-3-0-0.pabr3.global-gateway.net.nz [202.37.245.230]
 10    160 ms    161 ms    170 ms  pao1-br1-g2-1-101.gnaps.net [198.32.176.165]
 11    180 ms    181 ms    180 ms  lax1-br1-p2-1.gnaps.net [199.232.44.5]
 12    170 ms    170 ms    171 ms  lax1-br1-ge-0-1-0.gnaps.net [199.232.44.50]
 13    240 ms    241 ms    240 ms  nyc-m20-ge2-2-0.gnaps.net [199.232.44.21]
 14    240 ms    251 ms    250 ms  ash-m20-ge1-0-0.gnaps.net [199.232.131.36]
 15    241 ms    240 ms    250 ms  0503.ge-0-0-0.gbr1.ash.nac.net [207.99.39.157]
 16    251 ms    260 ms    250 ms  0.so-2-2-0.gbr2.nwr.nac.net [209.123.11.29]
 17    250 ms    260 ms    261 ms  0.so-0-3-0.gbr1.oct.nac.net [209.123.11.233]
 18    250 ms    260 ms    261 ms  209.123.182.243
 19    250 ms    260 ms    261 ms  sol.yourhost.co.nz [66.246.3.197]

Trace complete.

C:\>
```

PORT SCANNING TECHNIQUES

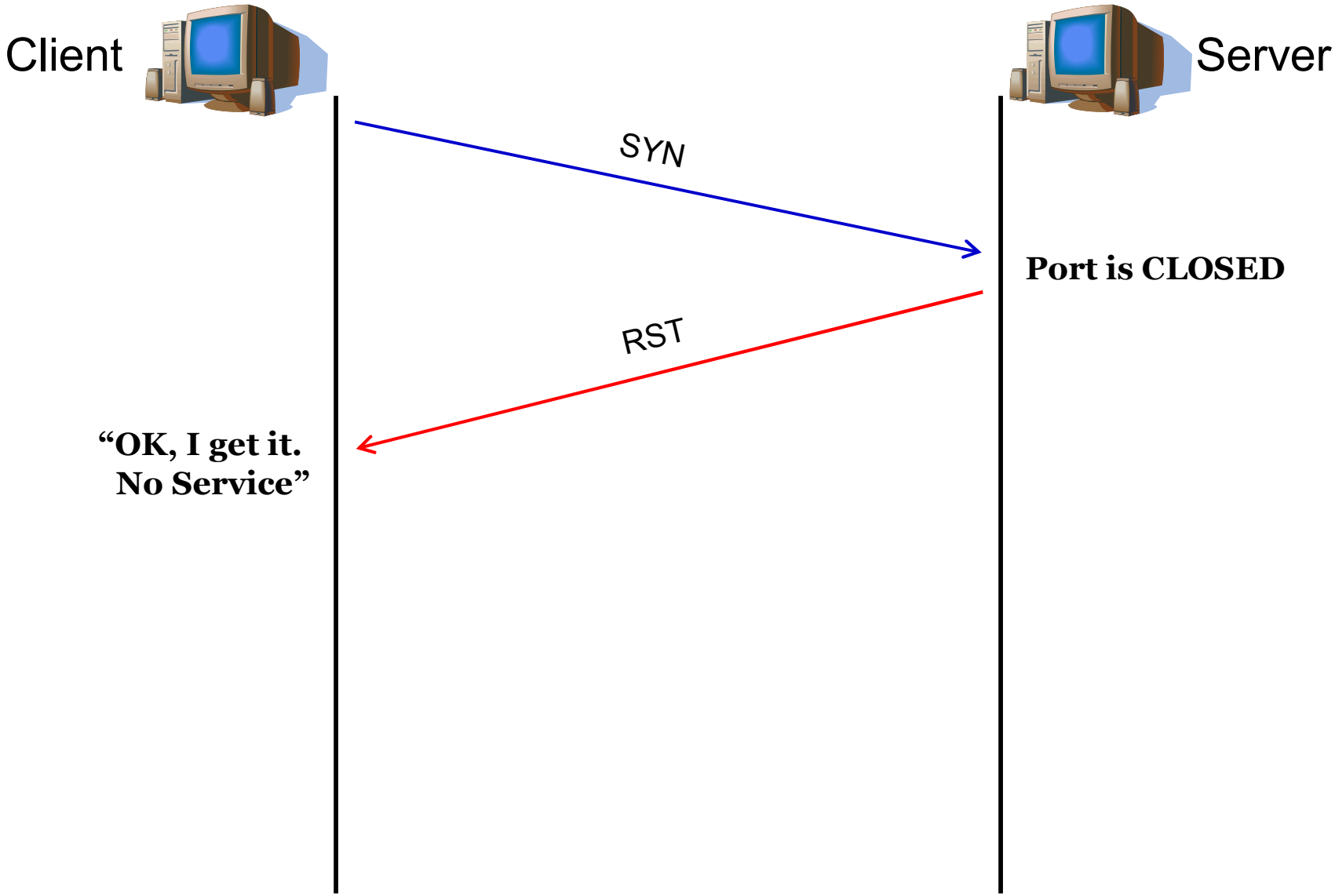
TCP Port Scanning

- It is a technique to detect which TCP enabled services are running/listening on the target host
- Based on the Three-Way-Handshake and TCP Flags
  - SYN, FIN, URG, PSH, RST, ACK
- A port can be:
  - Open
  - Closed
  - Filtered

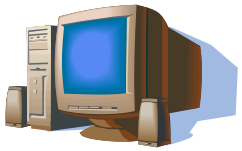
TCP Connect Scan – Open Port

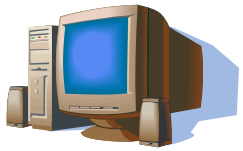


TCP Connect Scan – Closed Port



TCP Connect Scan – Filtered Port





SYN

Port is FILTERED

SYN

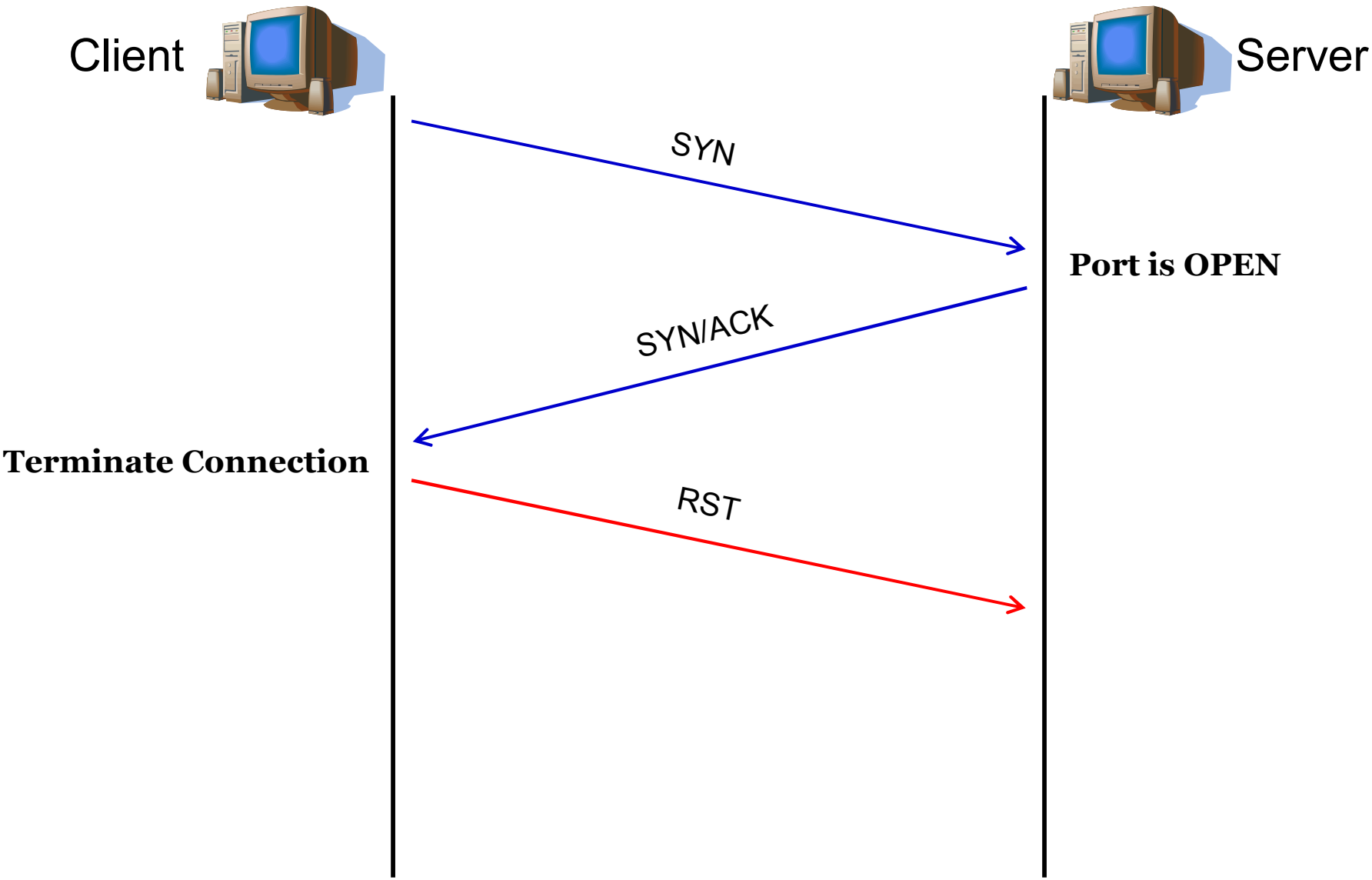
Port is FILTERED

SYN

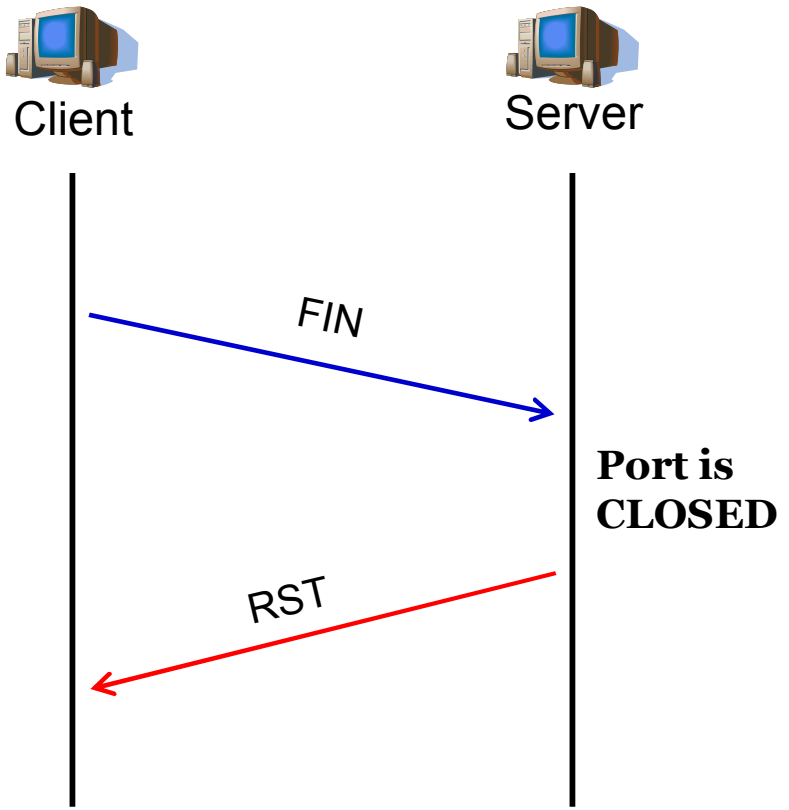
Port is FILTERED

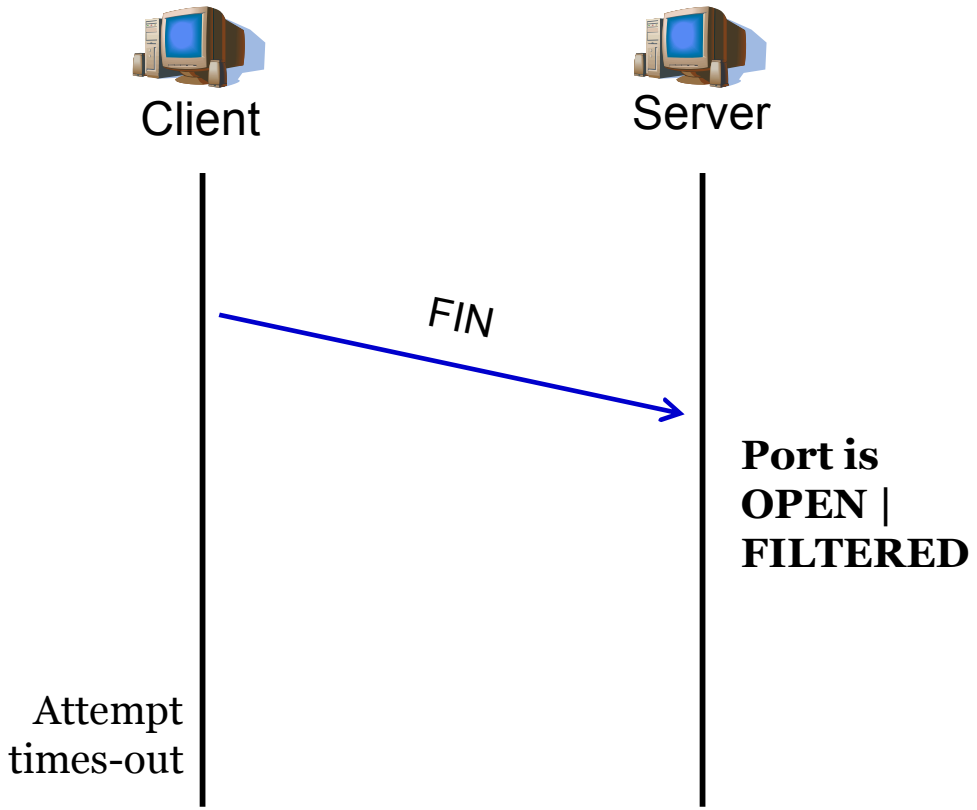
Connection times-out

TCP SYN Scan – Open Port



TCP FIN Scan


CLOSED
Client
Server
FIN
Port is
CLOSED
RST


OPEN | FILTERED
Client
Server
FIN
Port is
OPEN |
FILTERED
Attempt
times-out

TCP XMAS Scan

CLOSED

Client

Server

FIN/PSH/URG

Port is CLOSED

RST

OPEN | FILTERED

Client

Server

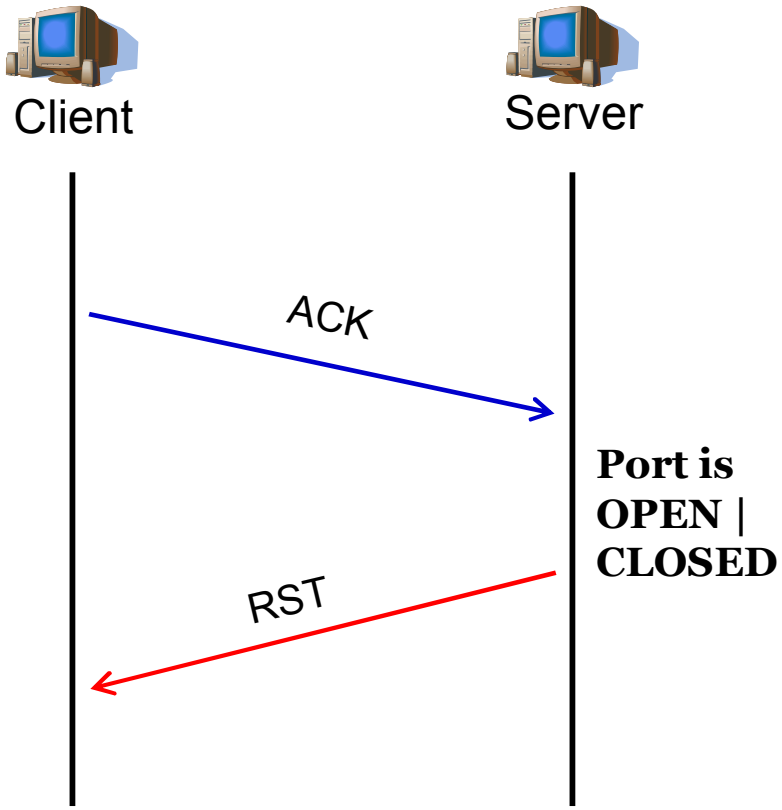
FIN/PSH/URG

Port is OPEN | FILTERED

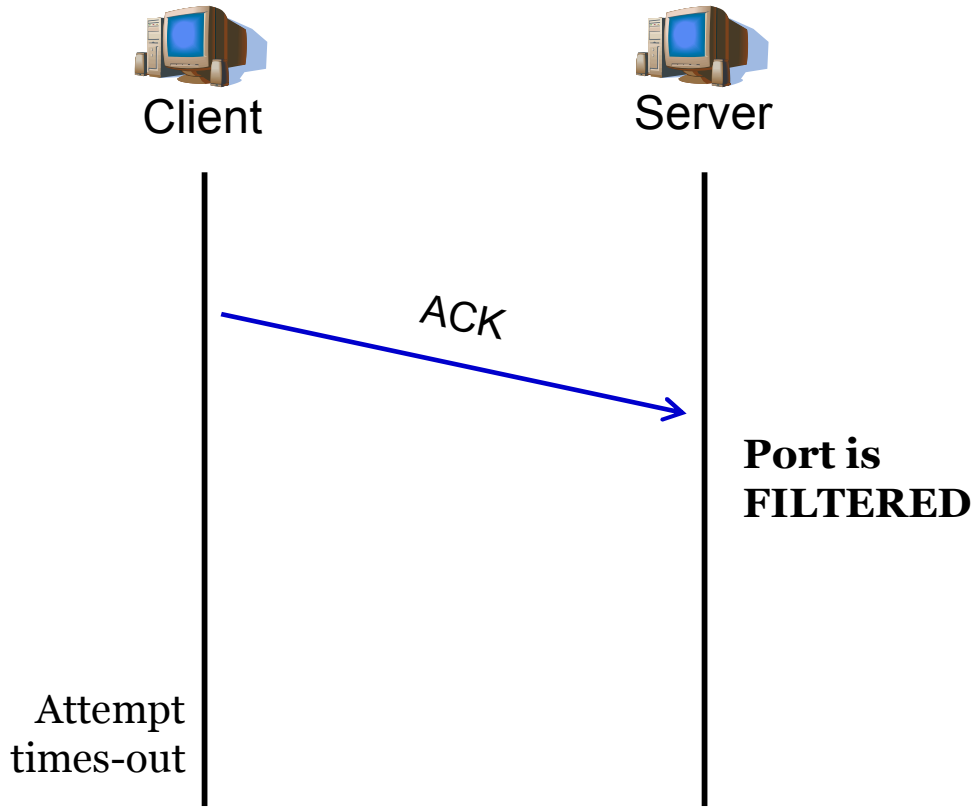
Attempt times-out

TCP ACK Scan

UNFILTERED
Stateless Inspection or
No Firewall



FILTERED
Stateful Inspection



UDP Port Scanning

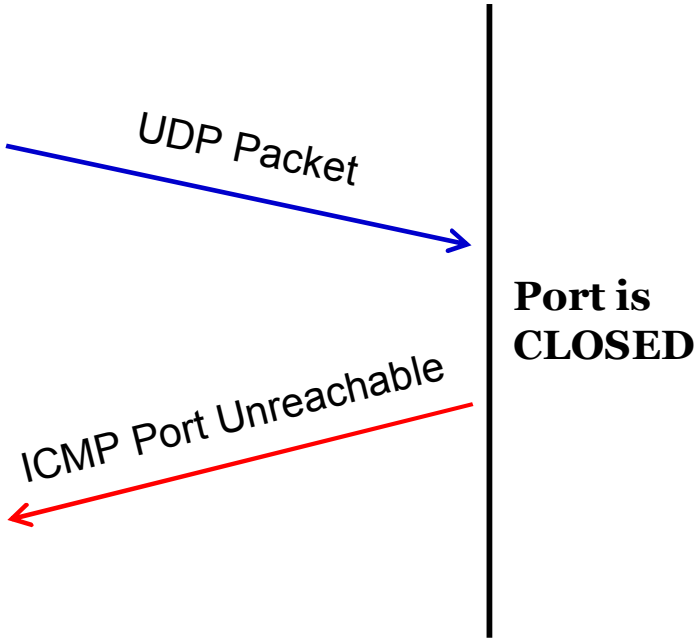
- It is a technique to detect which UDP enabled services are running/listening on the target host
- UDP is a connectionless protocol, hence there is no Three-Way-Handshake and no flags
- The protocol is not responsible to indicate whether a connection has been established. This is the job of upper layer protocols
- Therefore we don’t detect open ports but closed ones
- We can also use application protocols to find open ports

UDP Scan – Regular

CLOSED

Client

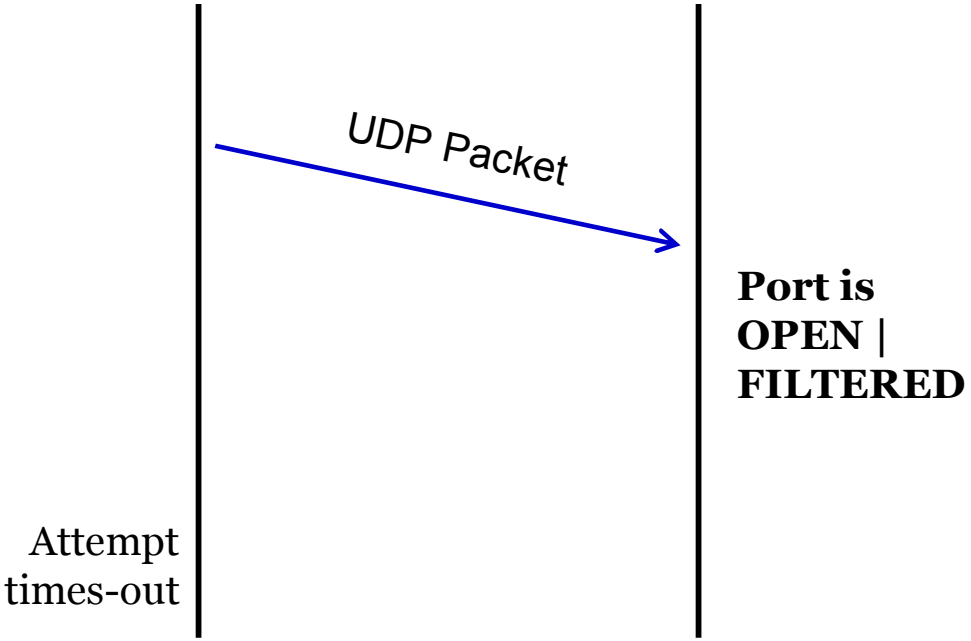
Server



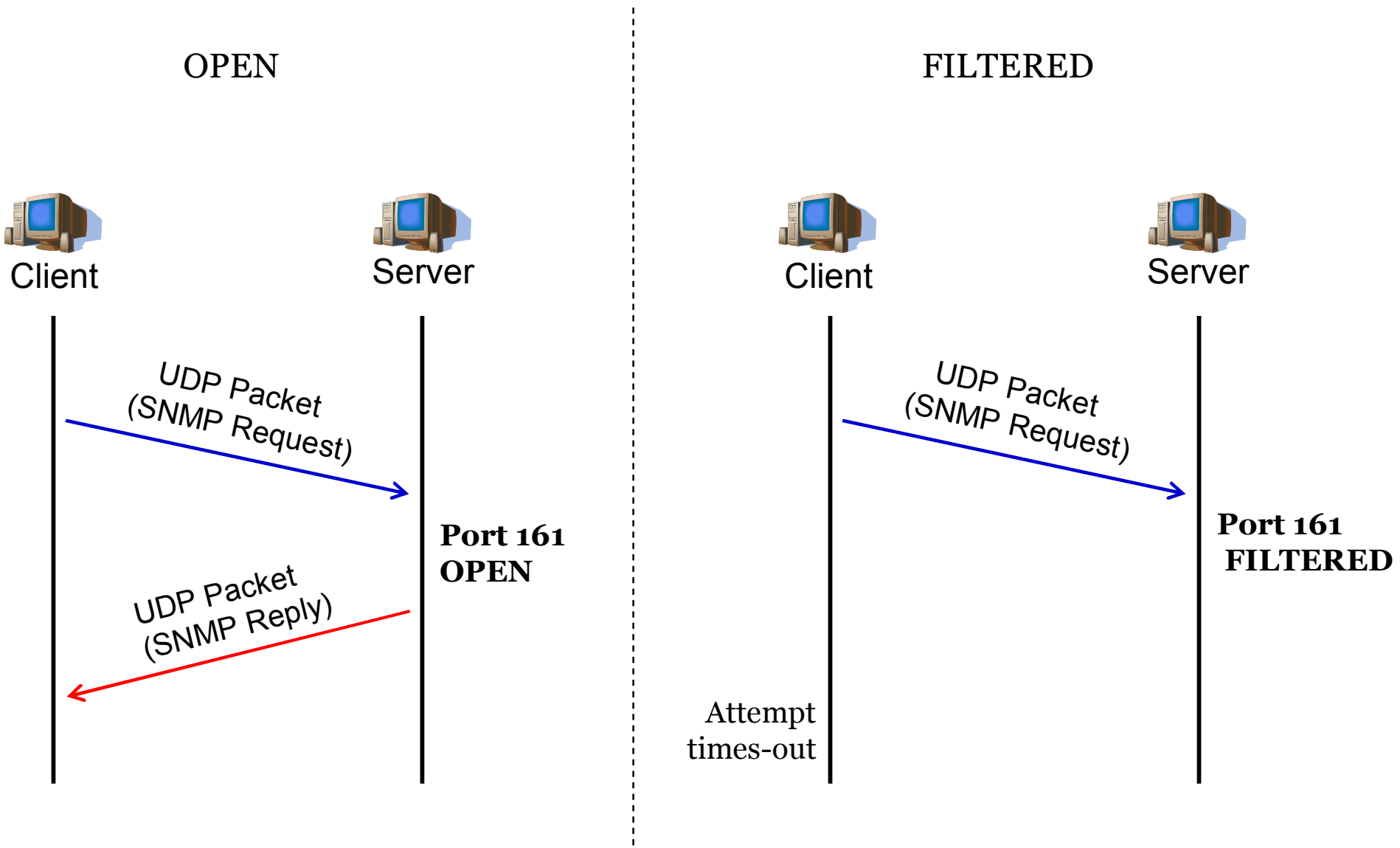
OPEN | FILTERED

Client

Server



UDP Scan – Protocol Specific



IDLE SCANNING

What is it?

- The most stealthy scan technique so far…
  - The attacker never sends traffic to the target using his/her real IP address, but instead..
  - “Instructs” a victim machine (zombie) to do it for him/her
  - If a security administrator or an IPS takes action against the “attacker”, it is the poor zombie who will take the blame (blacklisted or reported to the authorities)
  - The scan traverses the target’s firewall based on the rules that apply for the zombie host (not the attacker’s host)

Prerequisites

- In order for the scan to work properly:
  - The zombie must generate incremental IP IDs globally (not per session/host)
  - The zombie must accept TCP SYN/ACK packets (no stateful filtering for incoming packets)
  - The zombie must be idle (no traffic generated)
  - The attacker must be able to spoof his IP address (no NATing)
  - The attacker must be able to send TCP SYN/ACK packets (no stateful filtering on his end, for outgoing packets)

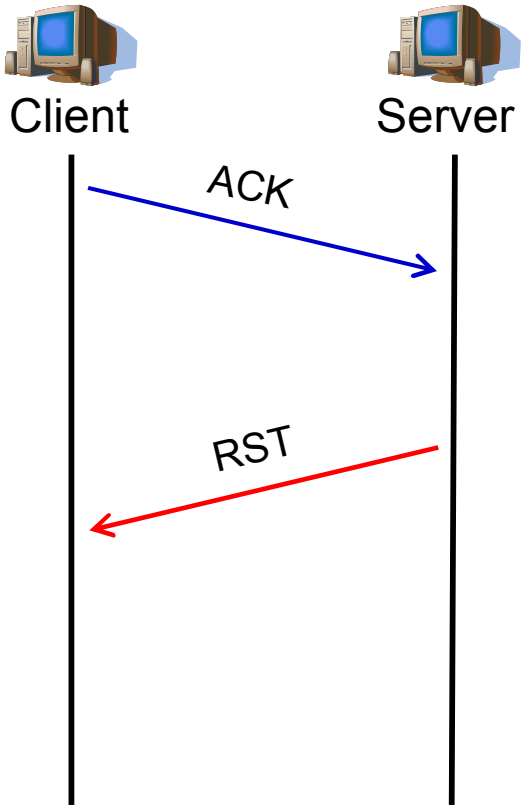
IP Header

IP Header Format

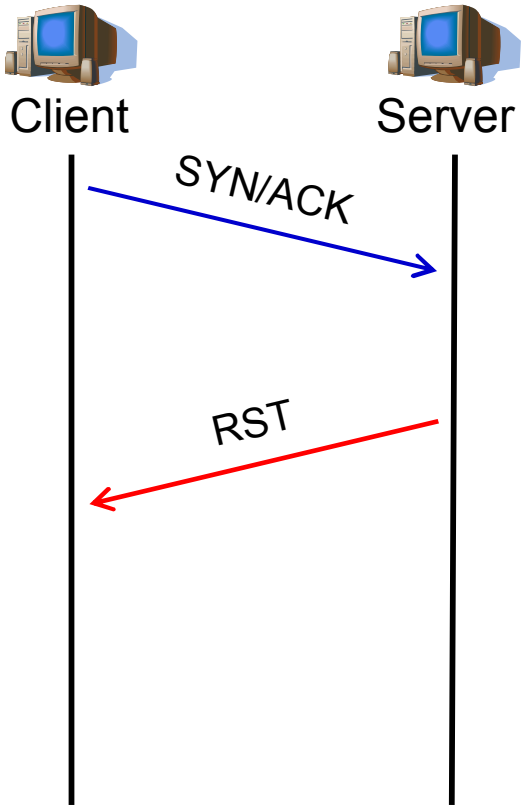
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

TCP Behavior

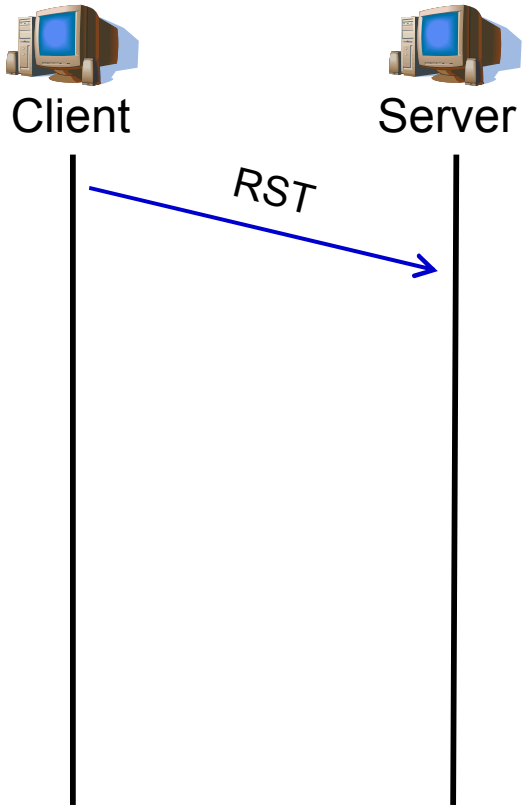
Unexpected packet that doesn't belong to any session



Unexpected packet with no associated SYN



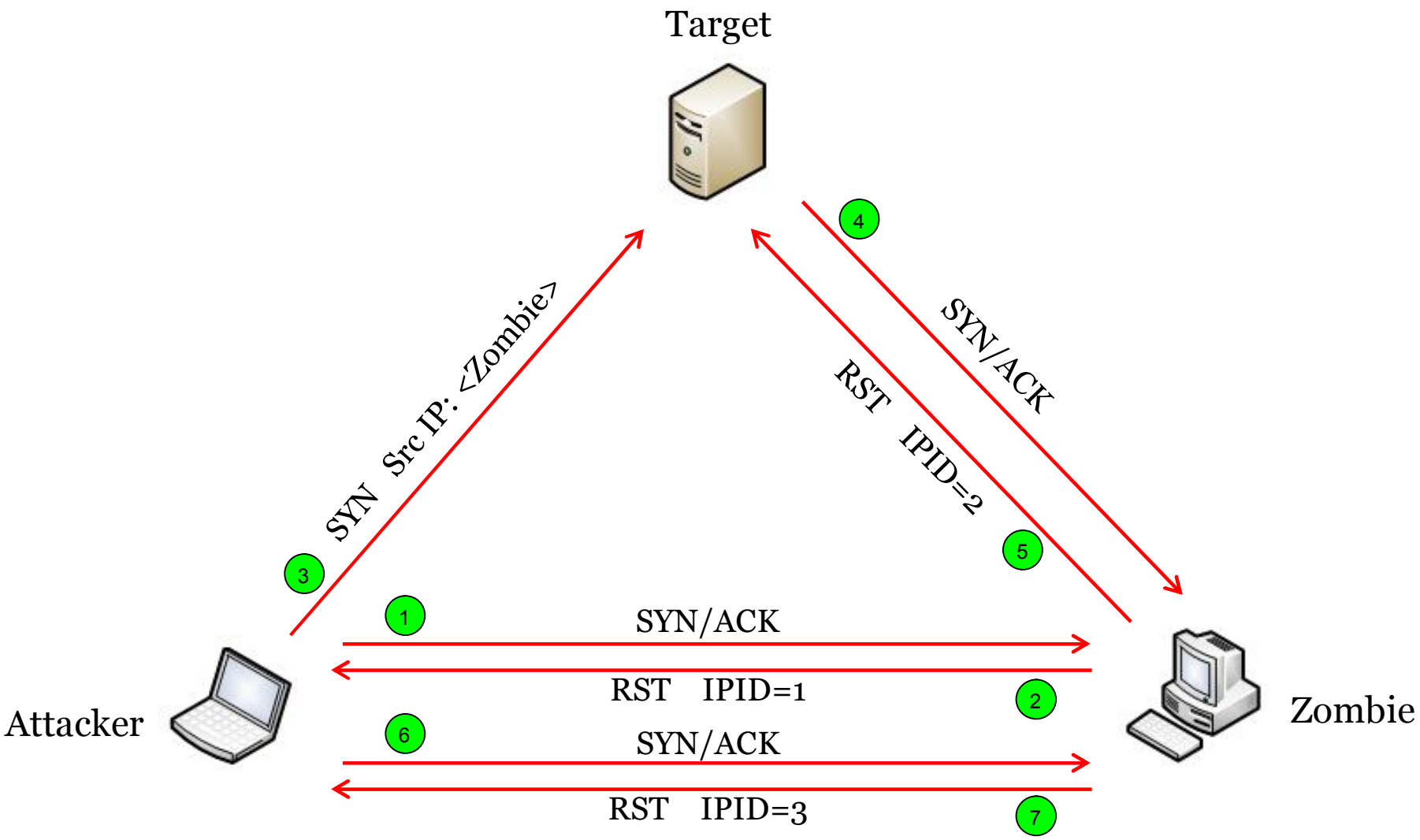
Packet that immediately terminates a connection



TCP Behavior

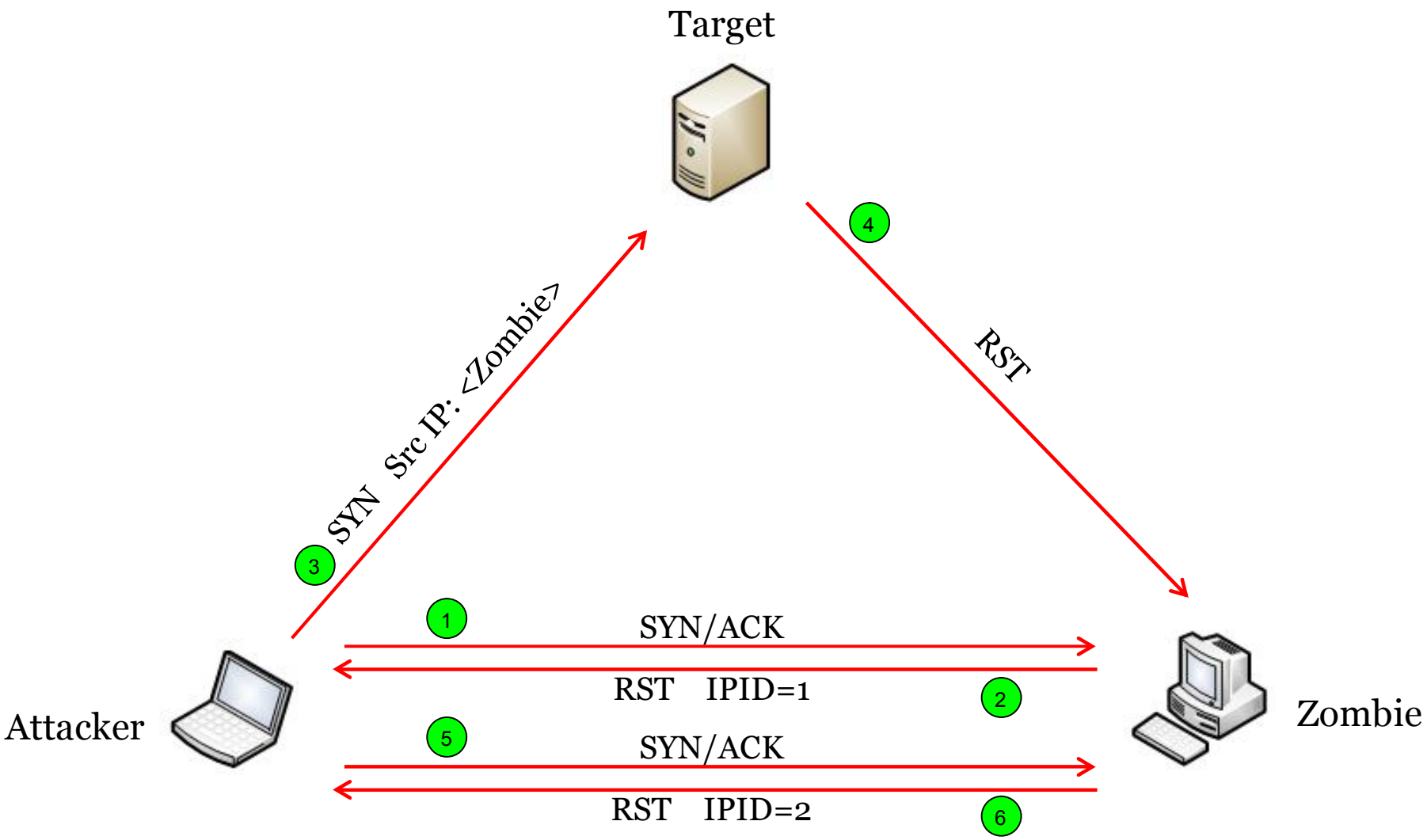
- The basic TCP rule illustrated above is:
  - Most TCP packets (e.g. SYN, FIN, ACK, SYN/ACK, etc) cause the receiving host to send a reply back to the sender (e.g. SYN/ACK, FIN/ACK, RST, etc)
  - RST packets don't cause a reply to be generated

TCP Idle Scan – Open Port

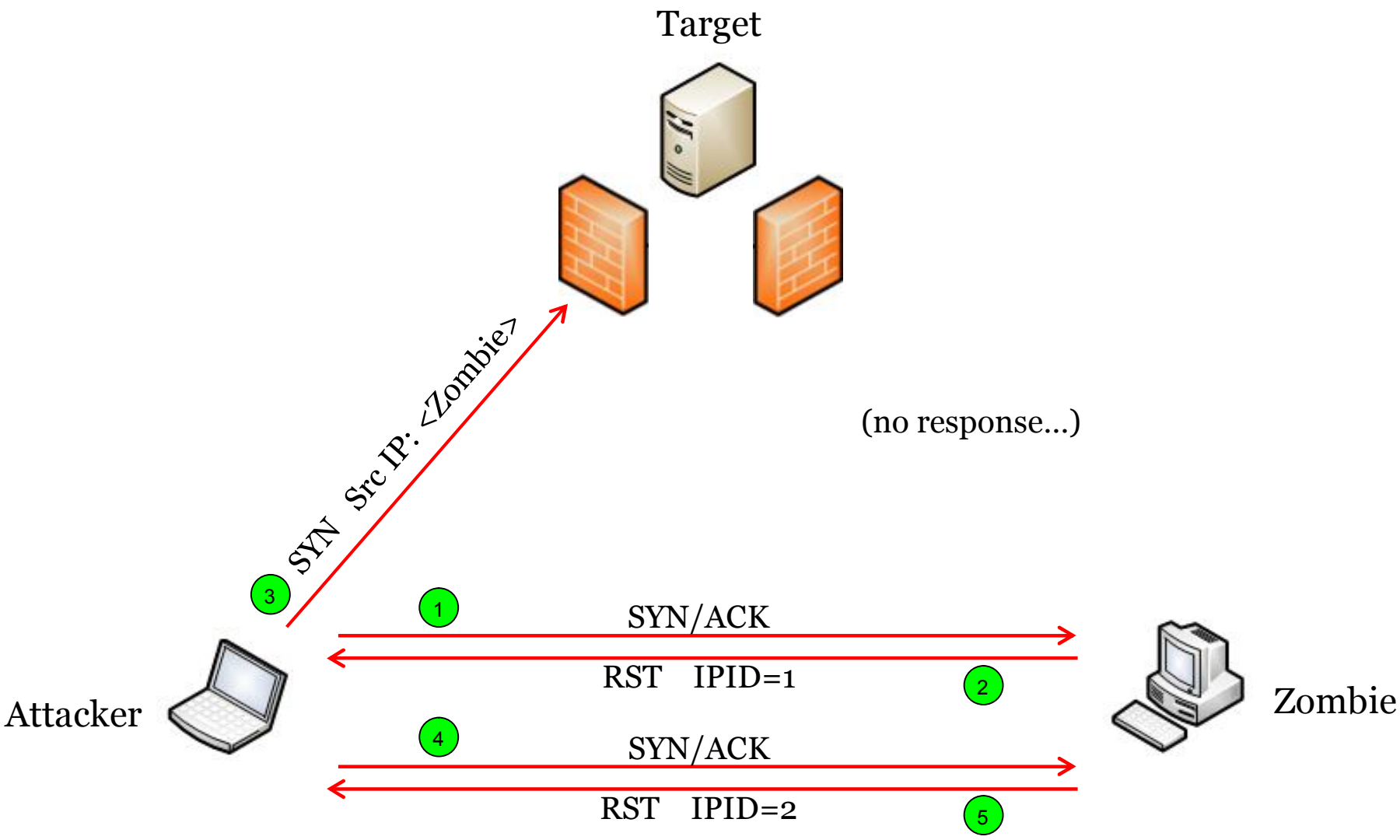


* IPID increments by 2 on newer OSes

TCP Idle Scan – Closed Port



TCP Idle Scan – Filtered Port



Possible Issues

- Certain things can render regular Idle Scans ineffective or hard to perform:
  - Some Operating Systems randomize or set the IP ID to zero until a connection has been established (completion of 3-Way-Handshake)
  - Busy/talkative zombie hosts
  - Stateful firewalls placed near or installed on zombie hosts
  - The Internet is moving from IPv4 to IPv6

ADVANCED IDLE SCANNING

Research Backgound

- Not a new scanning technique but an enhancement of the existing Idle scan
- Some of the ideas implemented have been previously suggested by people on the Bugtraq mailing list but no research work was ever published or tools created (besides *hping*)
- Our research work was performed in 2009 - 2010, hence the listed examples are old but still applicable to newer Operating Systems

Overcoming Obstacle 1

- Obstacle
  - Some Operating Systems randomize or set the IP ID to zero until a connection has been established (completion of 3-Way-Handshake)
- Observation
  - Popular tools send SYN/ACK packets to zombies. This is a clever trick which serves two purposes:
    - to check for incremental IP IDs
    - to check if there is a stateful firewall protecting the zombie

Overcoming Obstacle 1

- Solution 1
  - Why only use SYN/ACK packets? We should test different TCP Flags and check if any of them cause the zombie machine to reply with incremental IP IDs
- Solution 2
  - Since the IP ID field is part of the IP and not the TCP header, we can use other Layer 4 protocols (TCP, UDP, ICMP, etc) to check for incremental IP IDs

IP IDentifier Tool

```
Usage: identifier.py   [options]   target

Options:
  -h, --help            show this help message and exit
  -T TCP                use TCP scan mode with one or more of the following flags
                        S=Syn, A=Ack, F=Fin, X=Xmas, N=Null
  -U                    use UDP scan mode
  -I ICMP               use ICMP scan mode P=Ping (default) T=Timestamp
                        A=Address Mask
  -P                    use IP Protocol scan mode
  -t TCP_PORT           TCP port to use for scanning
  -u UDP_PORT           UDP port to use for scanning
  -p IP_PROTO           IP protocol number to use for scanning
  --ttl=TTL             how many hops the packet will traverse
  --tracert             detect IPID for every hop in the path
  --max_hops=MAX_HOPS       terminate traceroute at hop N. (use with –tracert only)
  --max_noresp=MAX_NORESP           terminate traceroute after N irresponsive hops.
                                    (use with --tracert only)
  -c PACKET_COUNT       number of packets to send
```

Open Port 23/tcp

 # ./identifier.py -TSAFXN -t23 -c3 172.16.1.17

```
Scan process initiated...
TCP SYN:  ip=172.16.1.17     id=9899       inc=0
TCP SYN:  ip=172.16.1.17     id=59880      inc=49981
TCP SYN:  ip=172.16.1.17     id=64964      inc=5084
-------------------------------------------------------------------
TCP SYN/ACK: No Response
TCP SYN/ACK: No Response
TCP SYN/ACK: No Response
-------------------------------------------------------------------
TCP ACK:  ip=172.16.1.17     id=37149      inc=0
TCP ACK:  ip=172.16.1.17     id=40149      inc=3000
TCP ACK:  ip=172.16.1.17     id=50986      inc=10837
-------------------------------------------------------------------
TCP FIN: No Response
-------------------------------------------------------------------
TCP XMAS: No Response
-------------------------------------------------------------------
TCP NULL: No Response
-------------------------------------------------------------------
```

Closed Port 80/tcp

#./identifier.py -TSAFXN –t80 -c3 172.16.1.17

```
Scan process initiated...
TCP SYN:  ip=172.16.1.17     id=38072       inc=0
TCP SYN:  ip=172.16.1.17     id=38073       inc=1
TCP SYN:  ip=172.16.1.17     id=38074       inc=1
-------------------------------------------------------------------
TCP SYN/ACK:  ip=172.16.1.17  id=38075       inc=0
TCP SYN/ACK:  ip=172.16.1.17  id=38076       inc=1
TCP SYN/ACK:  ip=172.16.1.17  id=38079       inc=3
-------------------------------------------------------------------
TCP ACK:  ip=172.16.1.17     id=38080       inc=0
TCP ACK:  ip=172.16.1.17     id=38082       inc=2
TCP ACK:  ip=172.16.1.17     id=38084       inc=2
-------------------------------------------------------------------
TCP FIN: No Response
-------------------------------------------------------------------
TCP XMAS: No Response
-------------------------------------------------------------------
TCP NULL: No Response
-------------------------------------------------------------------
```

Open Port 137/udp

 # unicornscan -mU 192.168.10.9

UDP open netbios-ns[137] from 192.168.10.9 ttl 128

 # ./identifier.py -U –u137 –c5 192.168.10.9

Scan process initiated...
UDP Scan: No Response
UDP Scan: No Response
UDP Scan: No Response
UDP Scan: No Response
UDP Scan: No Response

Open Port 137/udp (w/ payload)

 # hping2 192.168.10.9 --udp -p 137 -c 5 -r -d 50 --file ../137.txt

HPING 192.168.10.9 (eth0 192.168.10.9): udp mode set, 28 headers + 50 data bytes
[main] memlockall(): Success
Warning: can't disable memory paging!
len=293 ip=192.168.10.9 ttl=128 id=925 seq=0 rtt=0.4 ms
len=293 ip=192.168.10.9 ttl=128 id=+1 seq=1 rtt=0.3 ms
len=293 ip=192.168.10.9 ttl=128 id=+1 seq=2 rtt=0.4 ms
len=293 ip=192.168.10.9 ttl=128 id=+1 seq=3 rtt=0.4 ms
len=293 ip=192.168.10.9 ttl=128 id=+1 seq=4 rtt=0.4 ms

12:20:19.933615 IP (tos 0x0, ttl 128, id 58360, offset 0, flags [none], proto UDP (17), length 78)
192.168.10.3.4115 > 192.168.10.9.137: **NBT** UDP PACKET(137): **QUERY; REQUEST; BROADCAST**

12:20:19.934388 IP (tos 0x0, ttl 128, id 925, offset 0, flags [DF], proto UDP (17), length 185)
192.168.10.9.137 > 192.168.10.3.4115: **NBT** UDP PACKET(137): **QUERY; POSITIVE; RESPONSE;**

12:20:22.001809 IP (tos 0x0, ttl 128, id 30695, offset 0, flags [none], proto UDP (17), length 78)
192.168.10.3.34471 > 192.168.10.9.137: **NBT** UDP PACKET(137): **QUERY; REQUEST; BROADCAST**

12:20:22.002602 IP (tos 0x0, ttl 128, id 926, offset 0, flags [DF], proto UDP (17), length 185)
192.168.10.9.137 > 192.168.10.3.34471: **NBT** UDP PACKET(137): **QUERY; POSITIVE; RESPONSE;**

Closed Port 4444/udp

 # nmap -n -sU –p4444 192.168.10.9 –reason

Starting Nmap 5.00 (http://nmap.org) at 2010-05-31 20:01 EEST
Interesting ports on 192.168.10.9:

PORT	STATE	SERVICE	REASON
4444/udp	closed	krb524	port-unreach

 # ./identifier.py -U –u4444 –c5 192.168.10.9

Scan process initiated...

UDP Scan:	ip=192.168.10.9	id=5272	inc=0	(ICMP_Unreachable)
UDP Scan:	ip=192.168.10.9	id=5273	inc=1	(ICMP_Unreachable)
UDP Scan:	ip=192.168.10.9	id=5274	inc=1	(ICMP_Unreachable)
UDP Scan:	ip=192.168.10.9	id=5275	inc=1	(ICMP_Unreachable)
UDP Scan:	ip=192.168.10.9	id=5276	inc=1	(ICMP_Unreachable)

ICMP Requests & Replies

 # ./identifier.py -IPAT -c3 10.10.10.200

```
Scan process initiated...
ICMP Ping:  ip=10.10.10.200    id=5432        inc=0
ICMP Ping:  ip=10.10.10.200    id=63756       inc=58324
ICMP Ping:  ip=10.10.10.200    id=50910       inc=-12846
---------------------------------------------------------------------
ICMP Timestamp: ip=10.10.10.200    id=62973        inc=0
ICMP Timestamp: ip=10.10.10.200    id=29249        inc=-33724
ICMP Timestamp: ip=10.10.10.200    id=23274        inc=-5975
---------------------------------------------------------------------
ICMP Address Mask:    No Response
ICMP Address Mask:    No Response
ICMP Address Mask:    No Response
---------------------------------------------------------------------
```

ICMP TimeX Messages

 # ./identifier.py -TSA -c3 10.20.20.147 –ttl=1

```
Scan process initiated...
TCP SYN:  ip=10.10.10.200     id=33093      inc=0       (ICMP_TTL_Exceeded)
TCP SYN:  ip=10.10.10.200     id=33106      inc=13      (ICMP_TTL_Exceeded)
TCP SYN:  ip=10.10.10.200     id=33113      inc=7       (ICMP_TTL_Exceeded)
-------------------------------------------------------------------
TCP SYN/ACK: ip=10.10.10.200  id=33116      inc=0       (ICMP_TTL_Exceeded)
TCP SYN/ACK: ip=10.10.10.200  id=33118      inc=2       (ICMP_TTL_Exceeded)
TCP SYN/ACK: ip=10.10.10.200  id=33124      inc=6       (ICMP_TTL_Exceeded)
-------------------------------------------------------------------
TCP ACK:  ip=10.10.10.200     id=33129      inc=0       (ICMP_TTL_Exceeded)
TCP ACK:  ip=10.10.10.200     id=33132      inc=3       (ICMP_TTL_Exceeded)
TCP ACK:  ip=10.10.10.200     id=33139      inc=7       (ICMP_TTL_Exceeded)
-------------------------------------------------------------------
```

Tracing the whole path to a host

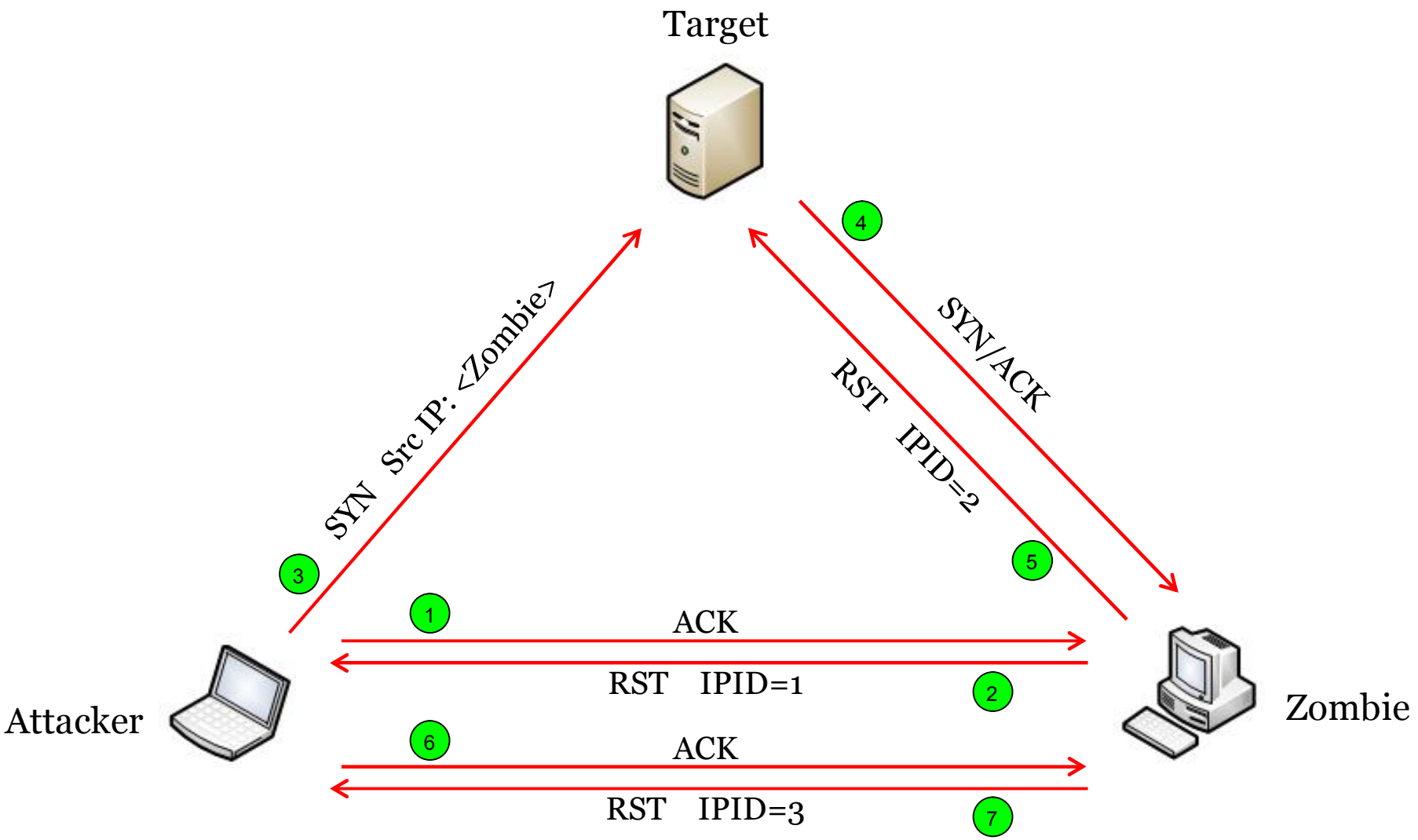
#./identifier.py -TS -t80 194.42.1.1 -c3 --tracert

```
Scan process initiated...
TCP SYN 1: No Response
TCP SYN 1: No Response
TCP SYN 1: No Response
-----------------------------------------------------------------
TCP SYN 2:  ip=91.184.192.49    id=55793       inc=0   (ICMP_TTL_Exceeded)
TCP SYN 2:  ip=91.184.192.49    id=55797       inc=4   (ICMP_TTL_Exceeded)
TCP SYN 2:  ip=91.184.192.49    id=55800       inc=3   (ICMP_TTL_Exceeded)
-----------------------------------------------------------------
TCP SYN 3:  ip=91.184.192.180   id=16742       inc=0   (ICMP_TTL_Exceeded)
TCP SYN 3:  ip=91.184.192.180   id=16747       inc=5   (ICMP_TTL_Exceeded)
TCP SYN 3:  ip=91.184.192.180   id=16752       inc=5   (ICMP_TTL_Exceeded)
-----------------------------------------------------------------
TCP SYN 4:  ip=91.184.192.163   id=39391       inc=0   (ICMP_TTL_Exceeded)
TCP SYN 4:  ip=91.184.192.163   id=39392       inc=1   (ICMP_TTL_Exceeded)
TCP SYN 4:  ip=91.184.192.163   id=39393       inc=1   (ICMP_TTL_Exceeded)
-----------------------------------------------------------------
…snip…
```

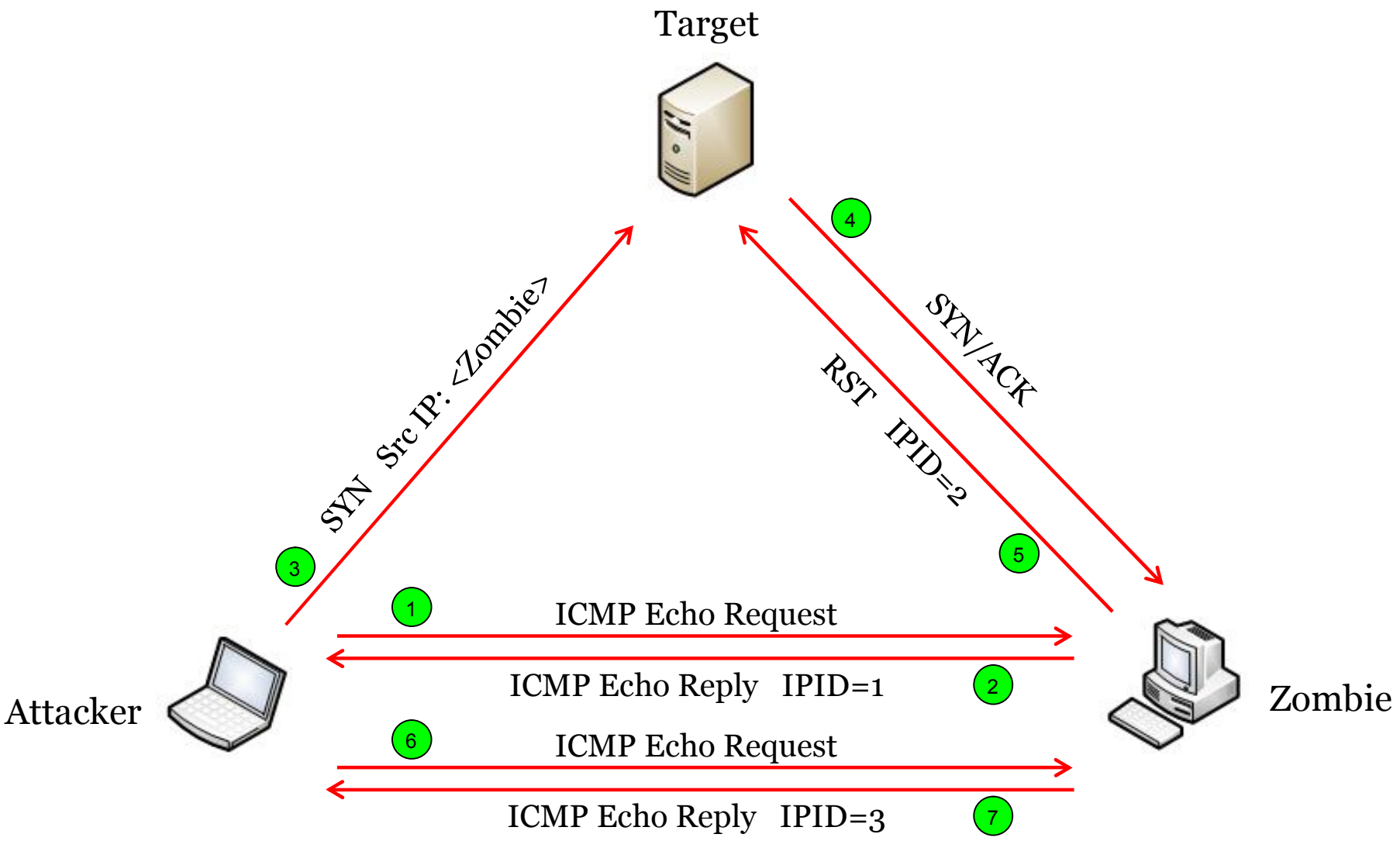
Tracing the whole path (Cont'd)

```
…snip…
------------------------------------------------------------------
TCP SYN 5:  ip=193.22.30.53    id=60650      inc=0   (ICMP_TTL_Exceeded)
TCP SYN 5:  ip=193.22.30.53    id=60651      inc=1   (ICMP_TTL_Exceeded)
TCP SYN 5:  ip=193.22.30.53    id=60652      inc=1   (ICMP_TTL_Exceeded)
------------------------------------------------------------------
TCP SYN 6:  ip=82.116.192.17   id=39113      inc=0   (ICMP_TTL_Exceeded)
TCP SYN 6:  ip=82.116.192.17   id=39114      inc=1   (ICMP_TTL_Exceeded)
TCP SYN 6:  ip=82.116.192.17   id=39116      inc=2   (ICMP_TTL_Exceeded)
------------------------------------------------------------------
TCP SYN 7:  ip=82.116.192.98   id=32316      inc=0   (ICMP_TTL_Exceeded)
TCP SYN 7:  ip=82.116.192.98   id=32317      inc=1   (ICMP_TTL_Exceeded)
TCP SYN 7:  ip=82.116.192.98   id=32318      inc=1   (ICMP_TTL_Exceeded)
------------------------------------------------------------------
TCP SYN 8:  ip=194.42.1.1      id=25476      inc=0
TCP SYN 8:  ip=194.42.1.1      id=25482      inc=6
TCP SYN 8:  ip=194.42.1.1      id=25487      inc=5
------------------------------------------------------------------
```

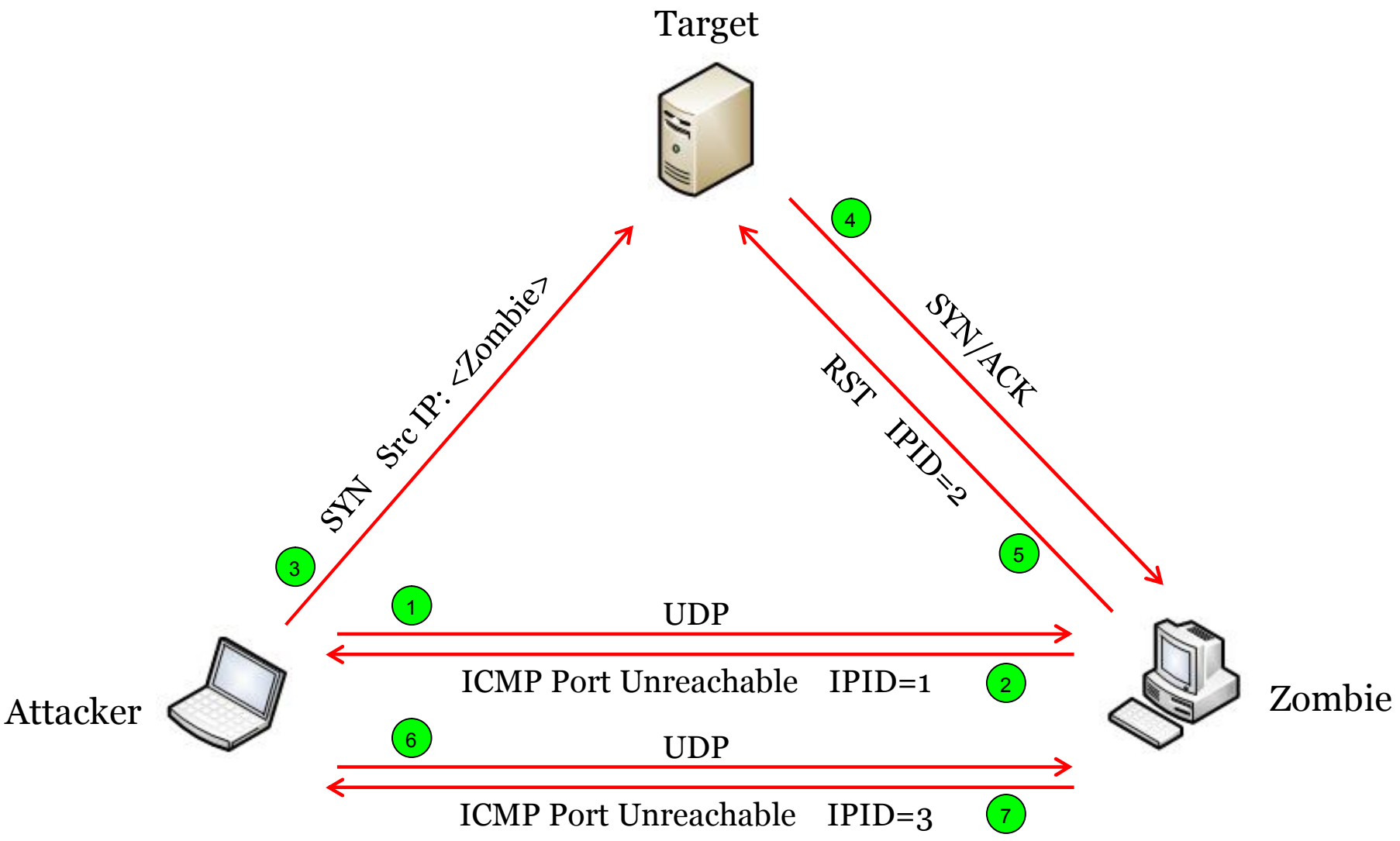
TCP (ACK) Idle Scan



ICMP Idle Scan



UDP Idle Scan



Overcoming Obstacle 2

- Obstacle
  - Busy zombie machines cannot be utilized because extraneous traffic will increment the IP ID and mess-up our scan results
- Solution
  - Send a burst of packets and observe if there is a comparative increase in the IP ID sequence

Regular Idle Scan - NMAP

 # nmap -vv -n -PN -sI 10.10.10.253:80 10.10.10.200 -p23 --packet-trace

Starting Nmap 4.60 (http://nmap.org) at 2014-09-23 14:47 GMT
Initiating ARP Ping Scan at 14:47
Scanning 10.10.10.200 [1 port]
Completed ARP Ping Scan at 14:47, 0.01s elapsed (1 total hosts)
Initiating idle scan against 10.10.10.200 at 14:47
Interesting ports on 10.10.10.200:

PORT	STATE	SERVICE
23/tcp	open	telnet

…snip…
SENT (0.7200s) TCP 10.50.12.44:63957 > 10.10.10.253:80 SA ttl=42 id=37634 iplen=44 seq=1663218960
RCVD (0.7200s) TCP 10.10.10.253:80 > 10.50.12.44:63957 R ttl=64 id=33865 iplen=40 seq=4101936255
SENT (0.7200s) TCP 10.10.10.253:80 > 10.10.10.200:23 S ttl=43 id=13887 iplen=44 seq=3990003447
SENT (0.7760s) TCP 10.50.12.44:63918 > 10.10.10.253:80 SA ttl=43 id=30888 iplen=44 seq=1663219460
RCVD (0.7760s) TCP 10.10.10.253:80 > 10.50.12.44:63918 R ttl=64 id=33867 iplen=40 seq=4101936255
…snip…

Puppeteer Idle Scanner

```
Usage: puppeteer.py   [options]   target:port

Options:
  -h, --help            show this help message and exit
  -z ZOMBIE             IP address of zombie host (mandatory)
  -T TCP                use TCP to communicate with the zombie. Available flags
                        S/A/F/U/P/R
  -U                    use UDP to communicate with the zombie
  -I ICMP               use ICMP to communicate with the zombie P=Ping (default)
                        T=Timestamp A=Address Mask
  -P                    use IP Protocol to communicate with the zombie
  -t TCP_PORT           TCP port to use when communicating with the zombie
  -u UDP_PORT           UDP port to use when communicating with the zombie
  -p IP_PROTO           IP protocol number to use when communicating with the zombie
  --ttl=TTL             how many hops the packet will traverse
  -c PACKET_COUNT               number of packets to send to zombie host
  -b BURST              number of packets to send to the target as a quick burst
  --fast                send zombie probes faster than 1pps
```

Regular Idle Scan - Puppeteer

 # ./puppeteer.py –z 10.10.10.253 -TS –t80 –c10 –b1 10.10.10.200:23

```
Scan process initiated...
TCP:  ip=10.10.10.253   id=30231  inc=0
TCP:  ip=10.10.10.253   id=30232  inc=1
TCP:  ip=10.10.10.253   id=30233  inc=1
TCP:  ip=10.10.10.253   id=30234  inc=1
TCP:  ip=10.10.10.253   id=30235  inc=1
TCP: sending 1 spoofed packet(s)...
TCP:  ip=10.10.10.253   id=30237  inc=2
TCP:  ip=10.10.10.253   id=30238  inc=1
TCP:  ip=10.10.10.253   id=30239  inc=1
TCP:  ip=10.10.10.253   id=30240  inc=1
TCP:  ip=10.10.10.253   id=30241  inc=1
------------------------------------------------------------------------
```

Regular Idle Scan - Busy Zombie

 # ./puppeteer.py –z 10.10.10.253 -TS –t80 –c10 –b1 10.10.10.200:23

```
Scan process initiated...
TCP:  ip=10.10.10.253   id=36532       inc=0
TCP:  ip=10.10.10.253   id=36539       inc=7
TCP:  ip=10.10.10.253   id=36545       inc=6
TCP:  ip=10.10.10.253   id=36551       inc=6
TCP:  ip=10.10.10.253   id=36558       inc=7
TCP: sending 1 spoofed packet(s)...
TCP:  ip=10.10.10.253   id=36565       inc=7
TCP:  ip=10.10.10.253   id=36571       inc=6
TCP:  ip=10.10.10.253   id=36577       inc=6
TCP:  ip=10.10.10.253   id=36584       inc=7
TCP:  ip=10.10.10.253   id=36590       inc=6
------------------------------------------------------------------------
```

Busy Zombie vs Burst of Packet

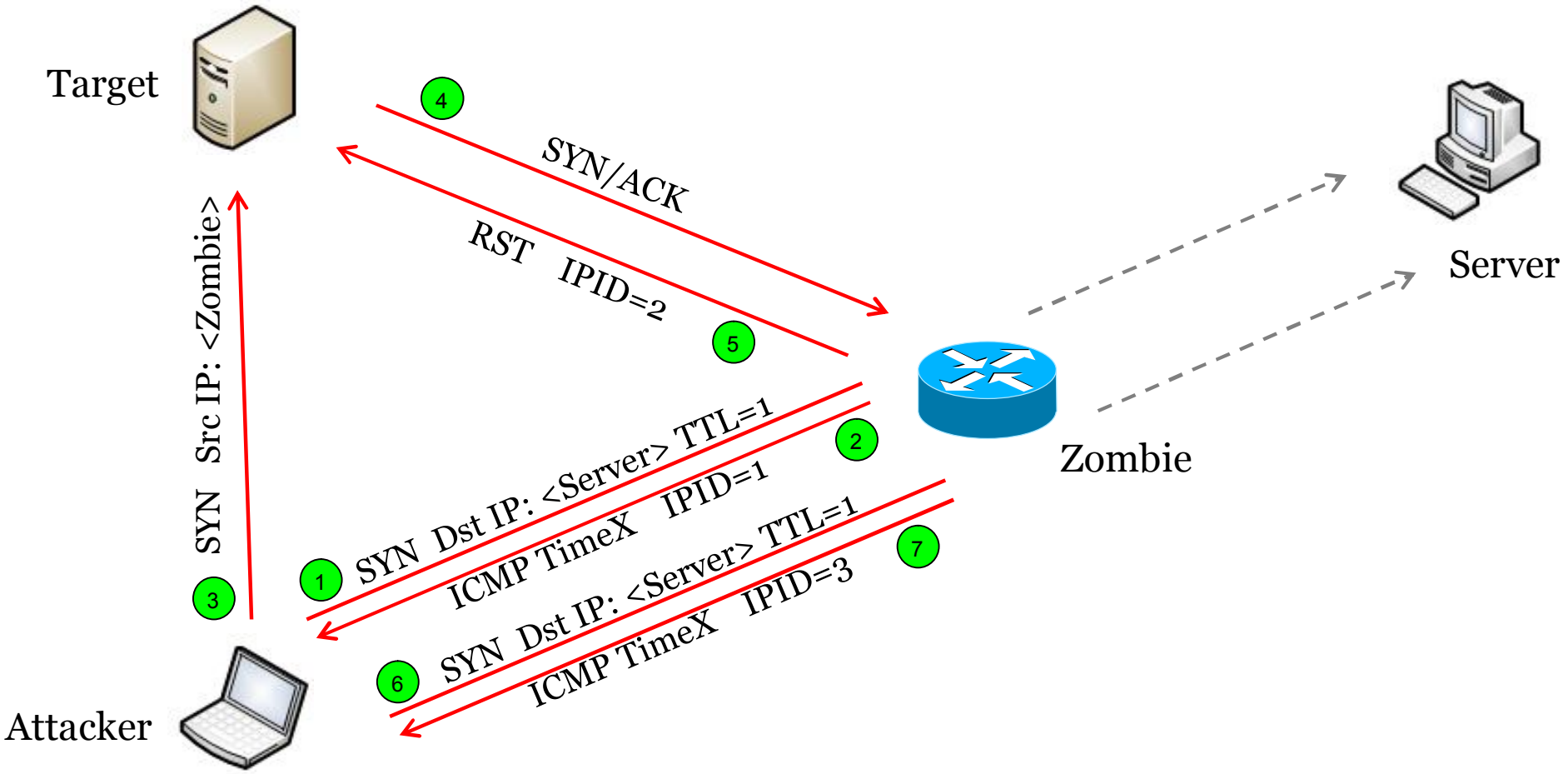
 # ./puppeteer.py –z 10.10.10.253 -TS –t80 –c10 –b5 10.10.10.200:23

```
Scan process initiated...
TCP:  ip=10.10.10.253   id=33493       inc=0
TCP:  ip=10.10.10.253   id=33499       inc=6
TCP:  ip=10.10.10.253   id=33505       inc=6
TCP:  ip=10.10.10.253   id=33512       inc=7
TCP:  ip=10.10.10.253   id=33518       inc=6
TCP: sending 5 spoofed packet(s)...
TCP:  ip=10.10.10.253   id=33529       inc=11
TCP:  ip=10.10.10.253   id=33535       inc=6
TCP:  ip=10.10.10.253   id=33542       inc=7
TCP:  ip=10.10.10.253   id=33548       inc=6
TCP:  ip=10.10.10.253   id=33554       inc=6
------------------------------------------------------------------------
```

Overcoming Obstacle 3

- Obstacle
  - Stateful firewalls are placed on the perimeter of a network or on the OSes, and as a results they render good zombie candidates useless to us
- Solution
  - To solve this problem we can utilize idle or moderately busy routers as our zombies, which are less likely to be protected by stateful firewalls
  - And with routers we can use ICMP Time Exceeded messages that are more likely to carry incremental IP IDs

ICMP TimeX Idle Scan



ICMP TimeX Idle Scan – Open Port

 # ./puppeteer.py -c16 -TS -b10 --ttl=5 -z 194.42.1.1 78.158.146.182:80

```
Scan process initiated...
TCP SYN:  ip=193.22.30.53     id=56987     inc=0     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=56989     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=56990     inc=1     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=56992     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=56995     inc=3     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=56997     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57003     inc=6     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57005     inc=2     (ICMP_TTL_Exceeded)
TCP SYN: sending 10 spoofed packet(s)...
TCP SYN:  ip=193.22.30.53     id=57017     inc=12    (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57021     inc=4     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57026     inc=5     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57028     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57031     inc=3     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57033     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57035     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57037     inc=2     (ICMP_TTL_Exceeded)
------------------------------------------------------------------
```

ICMP TimeX Idle Scan – Closed Port

 # ./puppeteer.py -c16 -TS -b10 --ttl=5 -z 194.42.1.1 78.158.146.182:4444

```
Scan process initiated...
TCP SYN:  ip=193.22.30.53     id=57424     inc=0     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57425     inc=1     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57427     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57429     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57430     inc=1     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57433     inc=3     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57435     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57438     inc=3     (ICMP_TTL_Exceeded)
TCP SYN: sending 10 spoofed packet(s)...
TCP SYN:  ip=193.22.30.53     id=57439     inc=1     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57441     inc=2     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57446     inc=5     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57447     inc=1     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57450     inc=3     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57451     inc=1     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57454     inc=3     (ICMP_TTL_Exceeded)
TCP SYN:  ip=193.22.30.53     id=57489     inc=35    (ICMP_TTL_Exceeded)
------------------------------------------------------------------
```

Overcoming Obstacle 4

- Obstacle
  - The Internet is moving from IPv4 to IPv6 and IPv6 is not vulnerable to IPv4 Idle Scans
- Solution
  - IPv6 Idle Scanning is still possible (see Mathias Morbitzer's presentation at HACK.LU 2013)
  - The best thing is that many of the principles discussed so far are applicable in IPv6 Idle Scanning

REVEALING TRUST RELATIONSHIPS

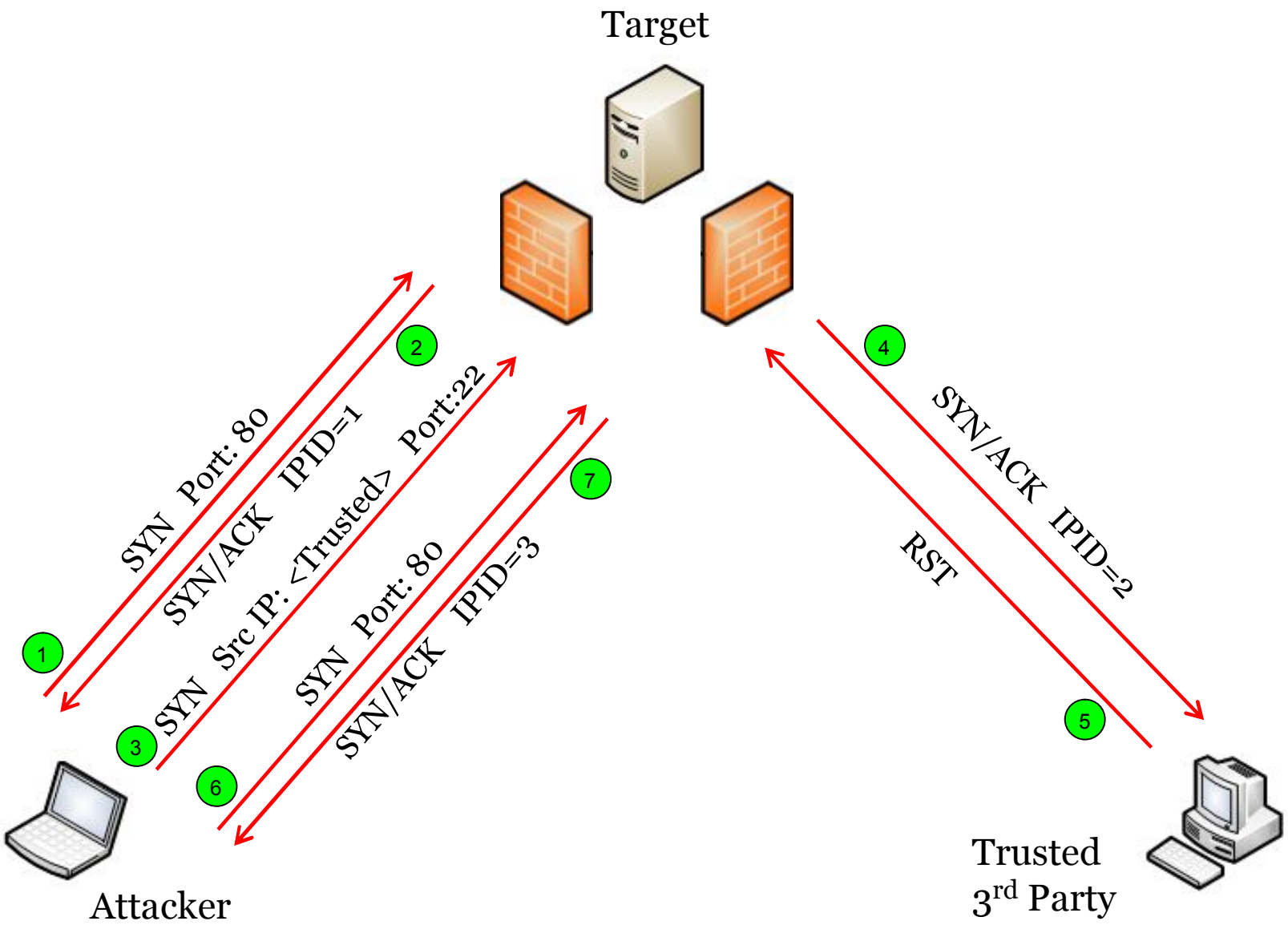
Introduction

- Regular Idle scans expose trust relationships between the zombie and the target, but…
- An improved technique can be used to reveal permissive firewall rules for trusted 3rd party networks / IP addresses
- The technique follows the same principles as Idle Scanning but with a few changes
  - We don't "ask" the zombie to reveal open/closed ports on the target,
  - …but instead we "ask" the target to reveal trusted IP addresses

What's Needed

- Prerequisites
  - The target host must generate global incremental IP IDs
  - At least one open/closed TCP or UDP port on the target or an ICMP responsive target
  - The attacker must be able to spoof his IP address (no NATing)

Basic Concept



Target Information

```
Starting Nmap 5.00 ( http://nmap.org ) at 2010-03-11 23:58 EET
Interesting ports on 192.168.1.13:
PORT      STATE     SERVICE
4444/tcp  open      krb524
4445/tcp  closed    unknown
4446/tcp  filtered  unknown
```

#./identifier.py -TS -t 4445 -c5 192.168.1.13

```
Scan process initiated...
TCP SYN:  ip=192.168.1.13      id=506  inc=0
TCP SYN:  ip=192.168.1.13      id=507  inc=1
TCP SYN:  ip=192.168.1.13      id=508  inc=1
TCP SYN:  ip=192.168.1.13      id=509  inc=1
TCP SYN:  ip=192.168.1.13      id=510  inc=1
---------------------------------------------------------------------
```

Trust Revealer Tool

Usage: revealer.py [options] target:port:protocol

Options:

-h, --help	show this help message and exit
-a SPOOFED	spoofed IP address and source port to be used to bypass ACL (mandatory) Acceptable format <IP_Address:Port>
-T TCP	use TCP to gather IPIDs from the target. Available flags S/A/F/U/P/R
-U	use UDP to gather IPIDs from the target
-I ICMP	use ICMP to gather IPIDs from the target P=Ping (default) T=Timestamp A=Address Mask
-P	use IP Protocol to gather IPIDs from the target
-t TCP_PORT	TCP port to use when gathering IPIDs from the target
-u UDP_PORT	UDP port to use when gathering IPIDs from the target
-p IP_PROTO	IP protocol number to use when gathering IPIDs from the target
--ttl=TTL	how many hops the packet will traverse
-c PACKET_COUNT	number of IPID probe packets to send to the target
-b BURST	number of spoofed packets to send to the target as a quick burst
--fast	send IPID probes to target faster than 1pps

Trust Revealer – Open Port

 # ./revealer.py -TS -t4445 -a 192.168.1.3 192.168.1.13:4444:T -c10 -b5

```
Scan process initiated...
TCP:  ip=192.168.1.13   id=541  inc=0
TCP:  ip=192.168.1.13   id=542  inc=1
TCP:  ip=192.168.1.13   id=543  inc=1
TCP:  ip=192.168.1.13   id=544  inc=1
TCP:  ip=192.168.1.13   id=545  inc=1
TCP: sending 5 spoofed packet(s)... to port 4444
TCP:  ip=192.168.1.13   id=551  inc=6
TCP:  ip=192.168.1.13   id=552  inc=1
TCP:  ip=192.168.1.13   id=553  inc=1
TCP:  ip=192.168.1.13   id=554  inc=1
TCP:  ip=192.168.1.13   id=555  inc=1
------------------------------------------------------------------
```

Trust Revealer – Closed Port

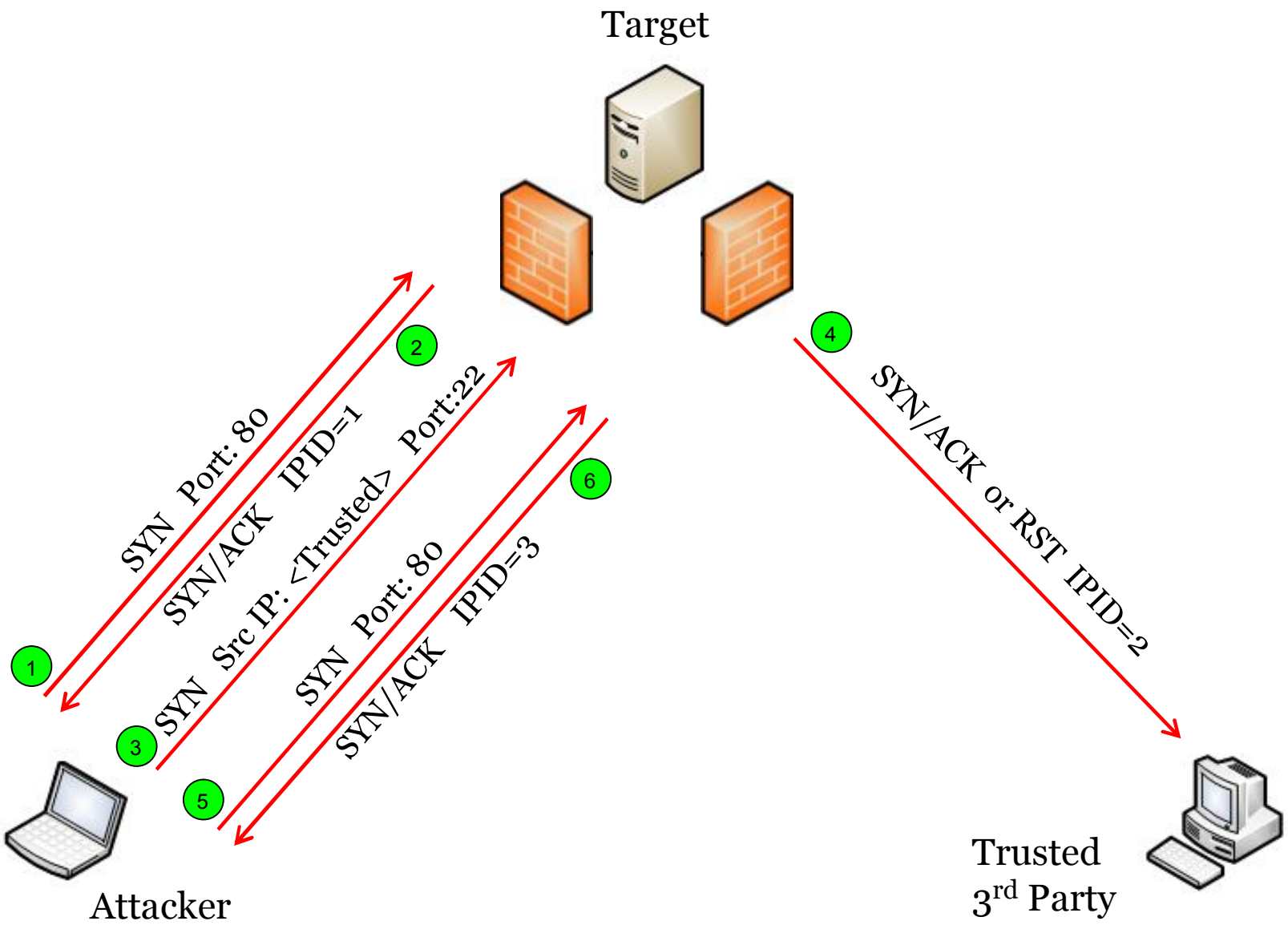
 # ./revealer.py -TS -t4444 -a 192.168.1.3 192.168.1.13:4445:T -c10 -b5

```
Scan process initiated...
TCP:  ip=192.168.1.13   id=556  inc=0
TCP:  ip=192.168.1.13   id=557  inc=1
TCP:  ip=192.168.1.13   id=558  inc=1
TCP:  ip=192.168.1.13   id=559  inc=1
TCP:  ip=192.168.1.13   id=560  inc=1
TCP: sending 5 spoofed packet(s)... to port 4445
TCP:  ip=192.168.1.13   id=566  inc=6
TCP:  ip=192.168.1.13   id=567  inc=1
TCP:  ip=192.168.1.13   id=568  inc=1
TCP:  ip=192.168.1.13   id=569  inc=1
TCP:  ip=192.168.1.13   id=570  inc=1
------------------------------------------------------------------
```

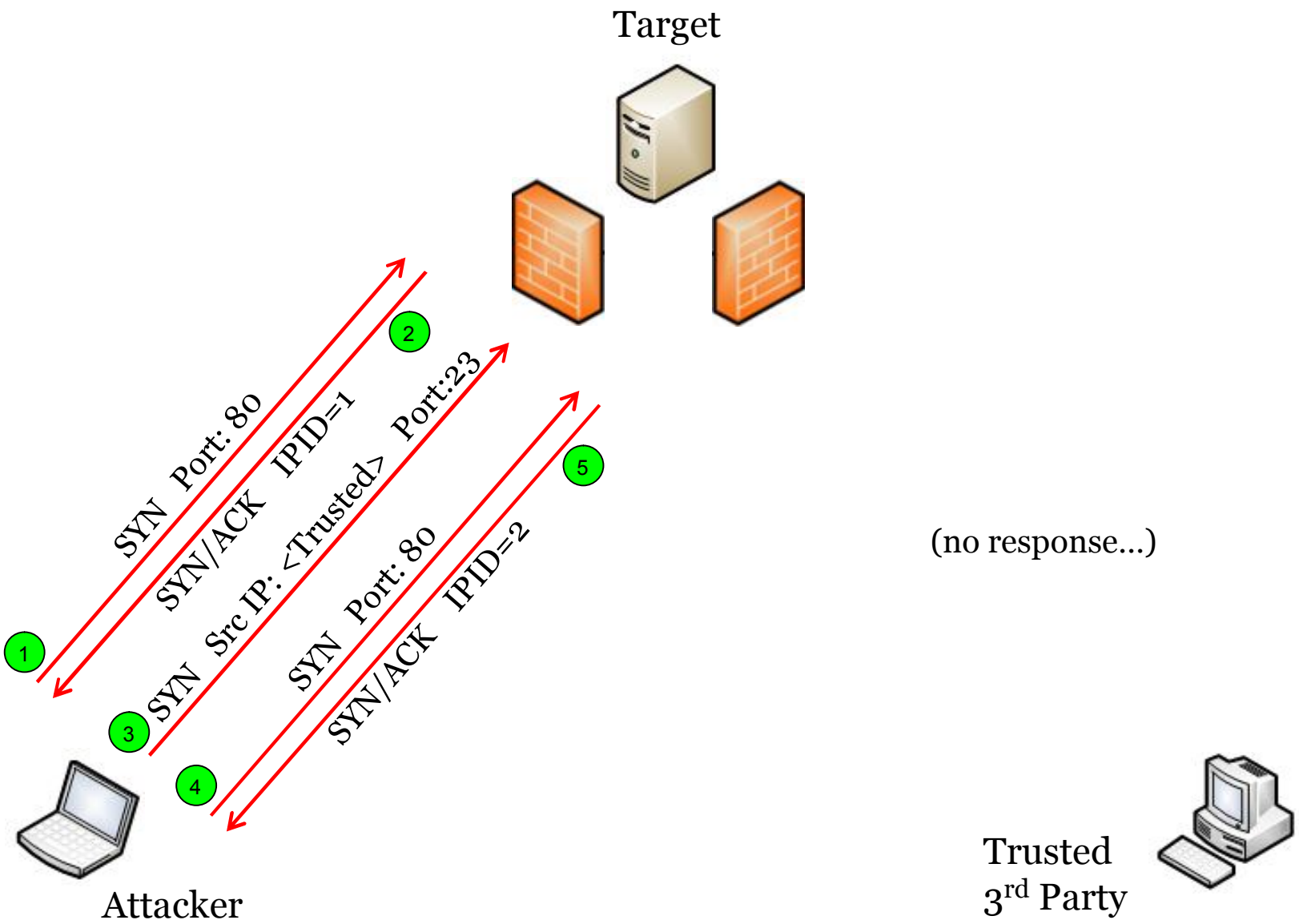
Observation

- We can only detect if a port is filtered or unfiltered, not if it's open or closed

Unfiltered Open|Closed Port



Filtered Port



Trust Revealer – Filtered Port

 # ./revealer.py -TS -t4445 -a 192.168.1.3 192.168.1.13:4446:T -c10 -b5

```
Scan process initiated...
TCP:  ip=192.168.1.13   id=1861        inc=0
TCP:  ip=192.168.1.13   id=1862        inc=1
TCP:  ip=192.168.1.13   id=1863        inc=1
TCP:  ip=192.168.1.13   id=1864        inc=1
TCP:  ip=192.168.1.13   id=1865        inc=1
TCP: sending 5 spoofed packet(s)... to port 4446
TCP:  ip=192.168.1.13   id=1866        inc=1
TCP:  ip=192.168.1.13   id=1867        inc=1
TCP:  ip=192.168.1.13   id=1868        inc=1
TCP:  ip=192.168.1.13   id=1869        inc=1
TCP:  ip=192.168.1.13   id=1870        inc=1
------------------------------------------------------------------
```

Trust Revealer – Filtered Port - Spoofed

 # ./revealer.py -TS -t4445 -a 192.168.1.2 192.168.1.13:4446:T -c10 -b5

```
Scan process initiated...
TCP:  ip=192.168.1.13   id=571  inc=0
TCP:  ip=192.168.1.13   id=572  inc=1
TCP:  ip=192.168.1.13   id=573  inc=1
TCP:  ip=192.168.1.13   id=574  inc=1
TCP:  ip=192.168.1.13   id=575  inc=1
TCP: sending 5 spoofed packet(s)... to port 4446
TCP:  ip=192.168.1.13   id=581  inc=6
TCP:  ip=192.168.1.13   id=582  inc=1
TCP:  ip=192.168.1.13   id=583  inc=1
TCP:  ip=192.168.1.13   id=584  inc=1
TCP:  ip=192.168.1.13   id=585  inc=1
------------------------------------------------------------------
```

Summary

- Enhancements to current Idle Scan techniques
  - Use of different protocols (e.g. IP, TCP, UDP, ICMP)
  - Use of different TCP Flags (e.g. SYN, ACK, FIN)
  - Use of ICMP Time Exceeded messages from routers
  - Use of packets bursts
- Improved technique for identifying firewall rules that allow connections from trusted 3rd parties based on source IP addresses.

Do You Have Any Questions?

We would be happy to help.

Presented

By: *Demetris Papapetrou*

For: *ISACA & (ISC)² Cyprus Chapters*

Date: 25 September 2014

Download Location:

http://www.qsecure.com.cy/whitepapers.html