

Computer Networks Project 2

Shirin Behnaminia 9919863

Narges Montazeri 9931053

Part 2:

2-1:

زمانی که ما سوکتمون رو ساختیم از دستور زیر استفاده کردیم

```
s = socket(AF_PACKET, SOCK_RAW)
```

این تایپ از سوکت SOCK_RAW به ما اجازه میدهد که مستقیماً در لایه ی ip یا ethernet قرار بگیریم و بسته ها را ارسال کنیم. در نتیجه حداقل طولی که لازمه یه بسته داشته باشه 14 بایته و کمتر از اون رو ارور میده. چون اگر یه مسیج رندوم به این سوکت بدیم تا منتقل کنه خودش میاد 14 بایت اول رو به عنوان ethernet header در نظر میگیره. (طول هدر اترنت 14 بایته)

2-2:

بسته مون باید هدر اترننش هدر ولیدی باشه. در نتیجه با توجه به نحوه ورودی گرفتن ما که یک رشته هگزادسیمال ورودی میگیره باید به تعداد زوج تا کاراکتر و حداقل 28 تا کاراکتر وارد کنیم که هر کاراکتر عددی بین 0 تا 9 یا حرفی بین a تا f است. اینکه قسمت type عدد ولیدی درنیاد مهم نیس چون وایرشارک میشناسه پکتو فقط type اش unknown در نظر گرفته میشه.

2-3:

```
narges@narges-vivobook:~/Documents/it/Lessons/4/Computer Networks/taklif/Project2/Network-Project2/Part1$ sudo python3 pkt_sender.py
What is your packet content? 1111111111111234567890120000
Which interface do you want to use?wlo1
narges@narges-vivobook:~/Documents/it/Lessons/4/Computer Networks/taklif/Project2/Network-Project2/Part1$
```

*wlo1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

eth.addr == 12:34:56:78:90:12

No.	Time	Source	Destination	Protocol	Length	Info
682	1.925959456	12:34:56:78:90:12	Private_11:11:11	0x0000	14	Ethernet II

Frame 682: 14 bytes on wire (112 bits), 14 bytes captured (112 bits) on interface wlo1, id 0

Ethernet II, Src: 12:34:56:78:90:12 (12:34:56:78:90:12), Dst: Private_11:11:11 (11:11:11:11:11:11)

- Destination: Private_11:11:11 (11:11:11:11:11:11)
- Source: 12:34:56:78:90:12 (12:34:56:78:90:12)
- Type: Unknown (0x0000)

0000 11 11 11 11 11 12 34 56 78 90 12 00 004 Vx....

wireshark_wlo1_20220627215931_CuzDSu.pcapng Packets: 2193 · Displayed: 1 (0.0%) Profile: Default

2-4:

```
What is your packet content? 5c80b67b1b9bfeaa813968640800450000289ffa000031060636a29f8aeaac140a0201bb943eec30fba93dfb1d6950100047f2b40000
Which interface do you want to use?wlo1
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 37950

No.	Time	Source	Destination	Protocol	Length	Info
6	0.030329047	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=1 Win=71 Len=0
56	0.224916840	172.20.10.2	162.159.138.234	TLSv1.2	141	Application Data
157	0.354218570	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=88 Win=71 Len=0
421	1.068218863	172.20.10.2	162.159.138.234	TLSv1.2	141	Application Data
433	1.191439448	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=175 Win=71 Len=0
513	1.384869174	172.20.10.2	162.159.138.234	TLSv1.2	141	Application Data
582	1.466324183	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=4294965188 Ack=4294934938 Win=71 Len=0
605	1.494940850	172.20.10.2	162.159.138.234	TLSv1.2	141	Application Data
612	1.507049179	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=262 Win=71 Len=0
663	1.810676899	172.20.10.2	162.159.138.234	TLSv1.2	141	Application Data
668	1.829465111	172.20.10.2	162.159.138.234	TCP	141	[TCP Retransmission] 37950 → 443 [PSH, ACK] Seq=349 Ack=1 Win=0
671	1.868559943	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=349 Win=71 Len=0
677	1.922094386	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=436 Win=71 Len=0
682	1.944974525	162.159.138.234	172.20.10.2	TCP	66	[TCP Dup ACK 677#1] 443 → 37950 [ACK] Seq=1 Ack=436 Win=71 Len=0
1238	3.294118126	172.20.10.2	162.159.138.234	TLSv1.2	115	Application Data
1254	3.411541424	162.159.138.234	172.20.10.2	TCP	54	443 → 37950 [ACK] Seq=1 Ack=497 Win=71 Len=0
1288	3.531025565	162.159.138.234	172.20.10.2	TLSv1.2	111	Application Data
1289	3.531035800	172.20.10.2	162.159.138.234	TCP	54	37950 → 443 [ACK] Seq=497 Ack=58 Win=501 Len=0
1411	3.900883730	172.20.10.2	162.159.138.234	TLSv1.2	141	Application Data

Frame 682: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlo1, id 0

Ethernet II, Src: fe:aa:81:39:68:64 (fe:aa:81:39:68:64), Dst: IntelCor_7b:1b:9b (5c:80:b6:7b:1b:9b)

Destination: IntelCor_7b:1b:9b (5c:80:b6:7b:1b:9b)

Source: fe:aa:81:39:68:64 (fe:aa:81:39:68:64)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 162.159.138.234, Dst: 172.20.10.2

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 52

Identification: 0xa1a5 (41381)

Flags: 0x0000

Fragment offset: 0

Time to live: 50

Protocol: TCP (6)

Header checksum: 0x037f [validation disabled]

[Header checksum status: Unverified]

Source: 162.159.138.234

Destination: 172.20.10.2

Transmission Control Protocol, Src Port: 443, Dst Port: 37950, Seq: 1, Ack: 436, Len: 0

```

0000  5c 80 b6 7b 1b 9b fe aa 81 39 68 64 08 00 45 00  \...\...9hd..E.
0010  00 34 a1 a5 00 00 32 06 03 7f a2 9f 8a ea ac 14  -4...2- .....
0020  0a 02 01 bb 94 3e ec 31 03 e6 3d fb 9d 83 80 10  >...1- .....
0030  00 47 7d 9f 00 00 01 01 05 0a 3d fb 9d 2c 3d fb  -G).....-...=.
0040  9d 83

```

پکتی که به عنوان duplicated ack شناخته شده پکتیه که ما مجدداً با همون seq num و ack num فرستادیم به همین دلیل dup ack در نظر گرفته شده.

2-5:

در این حمله، فرد هکر با شیوه های مختلف ip spoofing و sniff کردن پکت ها به محتوای ریکوستی که فرضاً A برای web server میفرستد دست پیدا میکند و بدون اینکه نیاز باشد محتوای آن را بخواند یا رمزگشایی کند آن را برای سرور retransmit(replay) میکند. این موضوع باعث میشود سرور به اشتباه بیفتد و تصور کند پیام از سمت شخص A است که مثلاً قصد داره دوباره لاگین کند. در نتیجه دیتا را برای فرد هکر میفرستد.

کاری که ما در سوال قبل انجام دادیم به نوعی یجور replay attack بود.

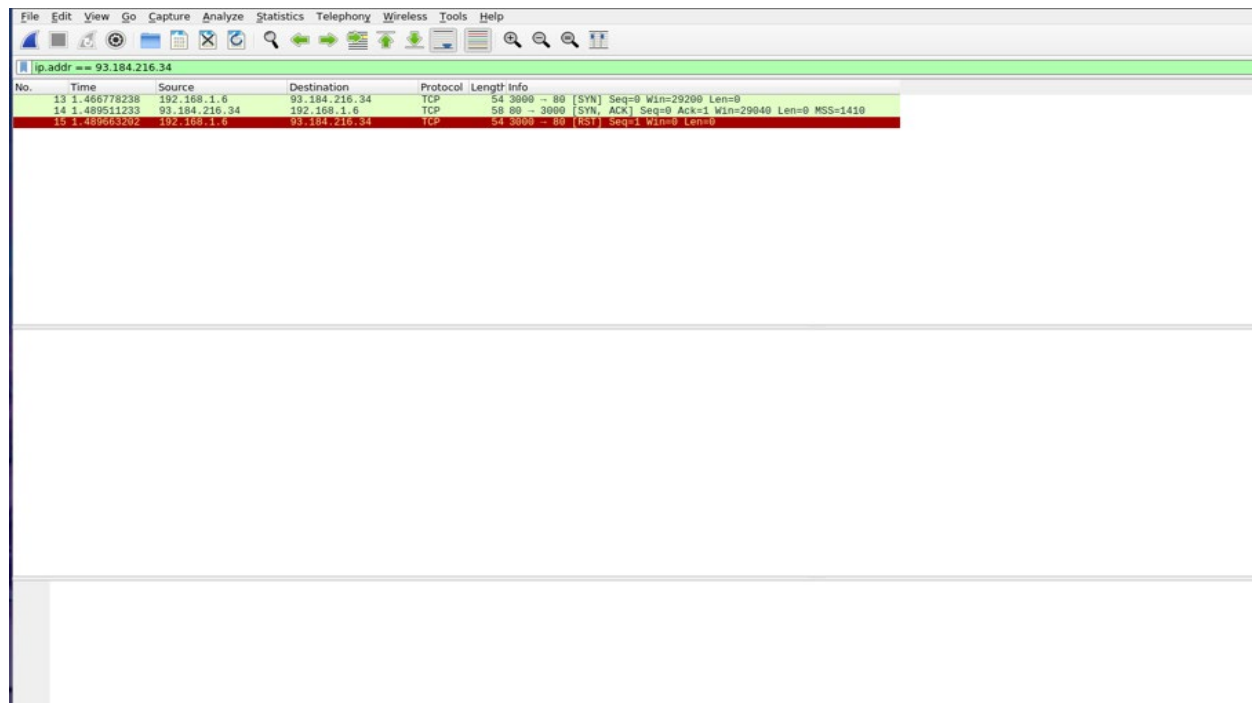
Part 3:

3-1:

مقادیری که نمیتونیم دلخواه مقدار دهی کنیم یکی مقادیر مربوط به مبدا هست. مقادیری که به tcp_syn بودن بسته مربوط است و مقادیری مثل چک سام که وابسته به بقیه فیلد ها مقدار میگیرند. لذا مقادیر قابل تغییر توی شکل عبارتند از:

,Dest_mac, proto3, ver, diff , t_len, id, ttl , dest_ip , dest_port, seq_num, w_size, up

3-2:



The image shows a Wireshark packet capture window. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. The main display area shows a list of captured packets. The first three packets are highlighted in red, indicating they are selected. The first packet is a TCP SYN packet from 192.168.1.6 to 93.184.216.34. The second packet is a TCP SYN-ACK packet from 93.184.216.34 to 192.168.1.6. The third packet is a TCP ACK packet from 192.168.1.6 to 93.184.216.34. The packet details pane on the right shows the selected packet's details, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol.

No.	Time	Source	Destination	Protocol	Length	Info
13	1.466778238	192.168.1.6	93.184.216.34	TCP	54	3000 → 80 [SYN] Seq=0 Win=29200 Len=0
14	1.489511233	93.184.216.34	192.168.1.6	TCP	58	80 → 3000 [SYN, ACK] Seq=0 Ack=1 Win=29040 Len=0 MSS=1460
15	1.502233022	192.168.1.6	93.184.216.34	TCP	54	3000 → 80 [ACK] Seq=1460 Win=0 Len=0

3-3:

این بسته ها قسمتی از فرایند triple handshaking را نشان می دهند. بسته ی اول همون بسته ایه که ما ارسال کرده ایم. بسته دوم بسته syn-ack است که از سمت سرور ارسال شده است. بسته ی سوم قاعدتاً باید بسته ack باشد که ما برای سرور ارسال کرده ایم ولی فلگ rst دارد. دلیلش احتمالاً این است که ما در سمت خودمون دیگه اک رو دریافت نمیکنیم چون قصد برقراری ارتباط tcp را نداریم. لذا کانکشن ایجاد شده داون میشود و به همین دلیل فلگ rst برابر 1 مقدار گرفته است.

3-4:

فایل ifinfo.sh ضمیمه شده است.

Part 4:

4-1:

14 Bytes

4-2:

14 Bytes 😊

اگر منظور هدر tcp باشه 20 بایته.

4-3:

با چک کردن فلگ های ACK , SYN در هدر tcp بسته ی موردنظر.

```
if(tcp_shark[7] == 1 and tcp_shark[10] == 1): # SYN_ACK
```

4-4:

Tcp_syn_sender sends a packet:

```
t2/Part2$ sudo python3 tcp_syn_sender.py  
[sudo] password for narges:  
Sent 54-byte packet on wlo1
```

Mini-wireshark sniffes it:

```
narges@narges-vivobook:~/Documents/lu/Lessons/4/Computer Networks/Laklil/Project2/Network-Project2/Part  
3$ sudo python3 miniwreshark.py  
port 80 is open on 172.217.16.206
```

Part 5:

5-1:

Minimap :

```
ect2/Network-Project2/Part4$ sudo python3 mininmap_sender.py  
[sudo] password for narges:  
Sent TCP SYN packet to port 0  
Sent TCP SYN packet to port 1  
Sent TCP SYN packet to port 2  
Sent TCP SYN packet to port 3  
Sent TCP SYN packet to port 4  
Sent TCP SYN packet to port 5  
Sent TCP SYN packet to port 6  
Sent TCP SYN packet to port 7  
Sent TCP SYN packet to port 8  
Sent TCP SYN packet to port 9  
Sent TCP SYN packet to port 10
```

```
Sent TCP SYN packet to port 1992  
Sent TCP SYN packet to port 1993  
Sent TCP SYN packet to port 1994  
Sent TCP SYN packet to port 1995  
Sent TCP SYN packet to port 1996  
Sent TCP SYN packet to port 1997  
Sent TCP SYN packet to port 1998  
Sent TCP SYN packet to port 1999
```

miniwireshark:

```
Part3$ sudo python miniwireshark.py
port 25 is open on 176.101.52.70
port 80 is open on 176.101.52.70
port 110 is open on 176.101.52.70
port 143 is open on 176.101.52.70
port 993 is open on 176.101.52.70
port 993 is open on 176.101.52.70
port 993 is open on 176.101.52.70
port 443 is open on 176.101.52.70
port 465 is open on 176.101.52.70
port 587 is open on 176.101.52.70
port 993 is open on 176.101.52.70
port 995 is open on 176.101.52.70
```

5-2:

باتوجه به نتیجه قسمت قبل پورت‌های ۲۵، ۸۰، ۱۱۰، ۱۴۳، ۴۴۳، ۴۶۵، ۵۸۷، ۹۹۳، ۹۹۵ روی آدرس ۱۷۲.۱۰۱.۵۲.۷۰ متعلق به دانشگاه صنعتی اصفهان باز هستند که متعلق به سرویس‌های زیر هستند.

Simple Mail Transfer Protocol (SMTP) : ۲۵

Hypertext Transfer Protocol (HTTP) : ۸۰

Post Office Protocol, version 3 (POP3) : ۱۱۰

Internet Message Access Protocol (IMAP) : ۱۴۳

HTTP Secure (HTTPS) : ۴۴۳

for email client to email server communication : ۴۶۵

email message submission : ۵۸۷

Internet Message Access Protocol (IMAPS) : ۹۹۳

Post Office Protocol 3 (POP3S) : ۹۹۵

5-3:

```
Part4$ sudo python3 mini_nmap_sender_tcpsocket.py -min 70 -max 90
Sent TCP SYN packet to port 89
Sent TCP SYN packet to port 82
Sent TCP SYN packet to port 88
Sent TCP SYN packet to port 87
Sent TCP SYN packet to port 73
Sent TCP SYN packet to port 78
Sent TCP SYN packet to port 83
Sent TCP SYN packet to port 79
Sent TCP SYN packet to port 74
Sent TCP SYN packet to port 77
Sent TCP SYN packet to port 71
Sent TCP SYN packet to port 80
Sent TCP SYN packet to port 81
Sent TCP SYN packet to port 72
Sent TCP SYN packet to port 70
Sent TCP SYN packet to port 84
Sent TCP SYN packet to port 85
Sent TCP SYN packet to port 86
Sent TCP SYN packet to port 76
Sent TCP SYN packet to port 75
```

این بار برای هر کانکشن یک سوکت جدا ساختیم که به شکل پارالل در چند ترد جدا دیتا ارسال میکنند. به همین خاطر همونطور که در شکل بالا پیداست بسته ها لزوماً به ترتیب پورت ارسال نشده اند. از طرف دیگه با توجه به تصویر وایرشارک که در صفحه بعد قرار داده شده است، اختلاف زمان بین بسته های متوالی بسیار کم است و این به همان همزمان ارسال شدن بسته ها برمیگردد.

این مثال نشون میده که از پورت های رنج 70 – 90 فقط پورت 80 دانشگاه باز است که منطقیه.

به طور کلی هرچند در زمان خیلی صرفه جویی شد ولی هزینه ساختن هر ترد برای هر سوکت ممکنه یکم محدودیت نرم افزاری ایجاد کنه.

*wlo1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 93.184.216.34

No.	Time	Source	Destination	Protocol	Length	Info
7088	12.915103007	172.20.10.2	93.184.216.34	TCP	54	3000 → 88 [SYN] Seq=0 Win=29200 Len=0
7089	12.915460566	172.20.10.2	93.184.216.34	TCP	54	3000 → 85 [SYN] Seq=0 Win=29200 Len=0
7090	12.915642527	172.20.10.2	93.184.216.34	TCP	54	3000 → 82 [SYN] Seq=0 Win=29200 Len=0
7091	12.915852309	172.20.10.2	93.184.216.34	TCP	54	3000 → 79 [SYN] Seq=0 Win=29200 Len=0
7092	12.916088431	172.20.10.2	93.184.216.34	TCP	54	3000 → 87 [SYN] Seq=0 Win=29200 Len=0
7093	12.916229500	172.20.10.2	93.184.216.34	TCP	54	3000 → 73 [SYN] Seq=0 Win=29200 Len=0
7094	12.916476327	172.20.10.2	93.184.216.34	TCP	54	3000 → 74 [SYN] Seq=0 Win=29200 Len=0
7095	12.916623343	172.20.10.2	93.184.216.34	TCP	54	3000 → 72 [SYN] Seq=0 Win=29200 Len=0
7096	12.916805178	172.20.10.2	93.184.216.34	TCP	54	3000 → 83 [SYN] Seq=0 Win=29200 Len=0
7097	12.916927628	172.20.10.2	93.184.216.34	TCP	54	3000 → 89 [SYN] Seq=0 Win=29200 Len=0
7098	12.917188217	172.20.10.2	93.184.216.34	TCP	54	3000 → 84 [SYN] Seq=0 Win=29200 Len=0
7099	12.917425912	172.20.10.2	93.184.216.34	TCP	54	3000 → 76 [SYN] Seq=0 Win=29200 Len=0
7100	12.917569301	172.20.10.2	93.184.216.34	TCP	54	3000 → 75 [SYN] Seq=0 Win=29200 Len=0
7101	12.917750356	172.20.10.2	93.184.216.34	TCP	54	3000 → 86 [SYN] Seq=0 Win=29200 Len=0
7102	12.918005797	172.20.10.2	93.184.216.34	TCP	54	3000 → 77 [SYN] Seq=0 Win=29200 Len=0
7103	12.918161710	172.20.10.2	93.184.216.34	TCP	54	3000 → 70 [SYN] Seq=0 Win=29200 Len=0
7104	12.918303134	172.20.10.2	93.184.216.34	TCP	54	3000 → 80 [SYN] Seq=0 Win=29200 Len=0
7105	12.918498878	172.20.10.2	93.184.216.34	TCP	54	3000 → 78 [SYN] Seq=0 Win=29200 Len=0
7106	12.918994237	172.20.10.2	93.184.216.34	TCP	54	3000 → 71 [SYN] Seq=0 Win=29200 Len=0
7107	12.919406058	172.20.10.2	93.184.216.34	TCP	54	3000 → 81 [SYN] Seq=0 Win=29200 Len=0
7112	13.176398489	93.184.216.34	172.20.10.2	TCP	58	80 → 3000 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
7113	13.176574744	172.20.10.2	93.184.216.34	TCP	54	3000 → 80 [RST] Seq=1 Win=0 Len=0

Frame 7112: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface wlo1, id 0

Ethernet II, Src: fe:aa:81:39:68:64 (fe:aa:81:39:68:64), Dst: IntelCor_7b:1b:9b (5c:80:b6:7b:1b:9b)

Internet Protocol Version 4, Src: 93.184.216.34, Dst: 172.20.10.2

Transmission Control Protocol, Src Port: 80, Dst Port: 3000, Seq: 0, Ack: 1, Len: 0

```

0000  5c 80 b6 7b 1b 9b fe aa 81 39 68 64 08 00 45 00  \...{....9hd..E
0010  00 2c 3b 8c 00 00 32 06 61 4f 5d b8 d8 22 ac 14  ,;...2 a0]..".
0020  0a 02 00 50 0b b8 08 7f 8c 32 17 49 39 d2 60 12  .P....2 IO..
0030  ff ff c3 8c 00 00 02 04 05 78                .....x

```