دانشکده‌ی مهندسی کامپیوتر

شبکه‌های کامپیوتری

نیم‌سال دوم ۰۴-۱۴۰۳

استاد: امیرمهدی صادق‌زاده

تمرین سوم

پاسخ‌دهنده: معین آعلی - ۴۰۱۱۰۵۵۶۱

فهرست مسائل

پاسخ مسئله‌ی ۱.

آ

کل بسته دارای ۳۰۰۰ بایت است و هدر ما ۲۰ بایت، پس دیتا ۲۹۸۰ بایت است. از طرفی می‌دانیم که هر فرگمنت ۱۴۸۰ بایت داده منتقل می‌کند. پس حداقل به ۳ فرگمنت برای این کار نیاز داریم.

$$۳ > \frac{۲۹۸۰}{۱۴۸۰} > ۲$$

ب

از بین روترهای A و B و C روتر C بیشترین پیشوند مشترک را دارد و بسته به آن روتر فرستاده می‌شود. همچنین داخل شبکه روتر D قرار ندارد و اصلا مچ نمیشود.

ج

- **درست.** پروتوکل $IPv۶$ با داشتن فضای آدرس‌دهی بسیار بزرگ، نیاز به ترجمه آدرس شبکه که در $IPv۴$ برای جبران کمبود آدرس‌ها استفاده می‌شود را کاهش می‌دهد.
- **درست.** تونلینگ به عنوان روشی برای ارسال بسته‌های یک پروتکل داخل پروتکل دیگر استفاده می‌شود که در VPN ها و همچنین برای عبور $IPv۶$ روی زیرساخت $IPv۴$ کاربرد دارد.

د

- **ضعف.** چون NAT باعث می‌شود دستگاه‌ها در داخل شبکه محلی آدرس‌های خصوصی داشته باشند و همه آنها پشت یک آدرس عمومی مشترک مخفی شوند، این موضوع مدیریت و شناسایی دستگاه‌ها را پیچیده‌تر می‌کند.
- **ضعف.** زیرا NAT نیاز به پردازش اضافه برای ترجمه آدرس‌ها دارد که ممکن است تاخیر کمی ایجاد کند و همچنین پنهان کردن آدرس‌های واقعی باعث می‌شود عیب‌یابی شبکه دشوارتر شود.

پاسخ مسئله‌ی ۲.

آ

دو سابنت طرح شده را داخل جدول مسیریابی روتر خود قرار می‌دهیم به این صورت:

Destination	CIDR	Next Hop
۱۰۷.۲۱.۴۱.۰	/۲۴	eth3
۱۰۷.۲۱.۱۷.۰	/۲۴	eth2
۰.۰.۰.۰	/۰	eth1

ب

نیاز نیست کار خیلی سختی انجام دهیم، صرفا آیپی سرور را به جدول مسیریابی اضافه می‌کنیم:

Destination	CIDR	Next Hop
۱۰۷.۲۱.۴۱.۰	/۲۴	eth3
۱۰۷.۲۱.۱۷.۵۰	/۳۲	eth3
۱۰۷.۲۱.۱۷.۰	/۲۴	eth2
۰.۰.۰.۰	/۰	eth1

جدول فوق به درستی کار می‌کند، چرا که مسیربابی با توجه به Longest-Prefix انجام می‌شود.

پاسخ مسئله‌ی ۳.

آ

زمان خروج	بسته خارج شده
$t = ۱$	۱
$t = ۲$	۴
$t = ۳$	۶
$t = ۴$	۵
$t = ۵$	۲
$t = ۶$	۳
$t = ۷$	۷
$t = ۸$	۸
$t = ۹$	۹
$t = ۱۰$	۱۰
$t = ۱۱$	۱۱
$t = ۱۲$	۱۲

ب

زمان خروج	بسته خارج شده
$t = ۱$	۱
$t = ۲$	۲
$t = ۳$	۴
$t = ۴$	۳
$t = ۵$	۶
$t = ۶$	۷
$t = ۷$	۵
$t = ۸$	۸
$t = ۹$	۱۱
$t = ۱۰$	۹
$t = ۱۱$	۱۰
$t = ۱۲$	۱۲

ج

زمان خروج	بسته خارج شده
$t = ۱$	۱
$t = ۲$	۲
$t = ۳$	۳
$t = ۴$	۴
$t = ۵$	۷
$t = ۶$	۸
$t = ۷$	۶
$t = ۸$	۹
$t = ۹$	۱۰
$t = ۱۰$	۵
$t = ۱۱$	۱۲
$t = ۱۲$	۱۱

پاسخ مسئله‌ی ۴.

آ

A: 223.1.1.0/24
B: 223.1.3.0/24
C: 223.1.2.0/24
D: 223.1.7.0/24
E: 223.1.8.0/24
F: 223.1.9.0/24

ب

توجه کنید که D و E و F زیرمجموعه سابنت B و C هستند زیرا آیپی بلااستفاده دارند:

A' : 214.97.254.0/24
B' : 214.97.255.0/25
C' : 214.97.255.128/25
D' : 214.97.255.60/31
E' : 214.97.255.62/31
F' : 214.97.255.126/31

ج

R2(Net C):
R1(NetA) ← 214.97.254.0/24
R3(NetB) ← 214.97.255.0/25
R3(NetB) ← 0.0.0.0/0

R1(Net A):
R3(NetB) ← 214.97.255.0/25
R2(NetC) ← 214.97.255.128/25
R2(NetC) ← 0.0.0.0/0

R3(Net B):
R1(NetA) ← 214.97.254.0/24
R2(NetC) ← 214.97.255.128/25
R2(NetC) ← 0.0.0.0/0

پاسخ مسئله‌ی ۵.

برای پیدا کردن آدرس شبکه، IP را با Subnet-Mask به صورت بیت به بیت AND می‌کنیم.

Subnet-Mask داده شده 255.255.240.0 است که معادل 20/ می‌باشد. این یعنی شبکه شامل آدرس‌های 172.17.16.0 تا 172.17.15.255 است. همچنین Default-Gateway ما 172.17.31.255 است که داخل این رنج شبکه قرار ندارد.

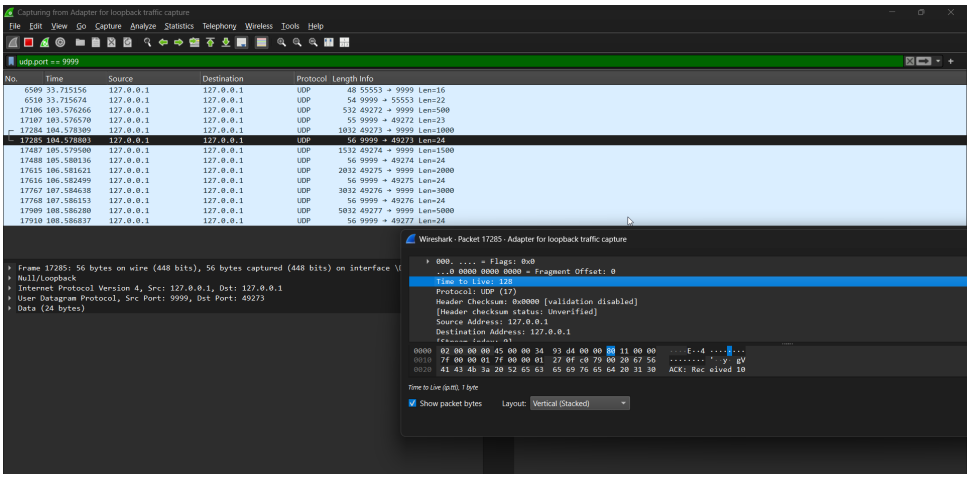
برای حل این مشکل یا باید آیپی را به محدوده دیگری تغییر دهیم، یا اینکه ماسک را تغییر دهیم تا بازه بزرگتر شود. همچنین میتوان Default-Gateway را تغییر داد تا به یک ادرس داخل شبکه تبدیل شود.

پاسخ مسئله‌ی ۶.

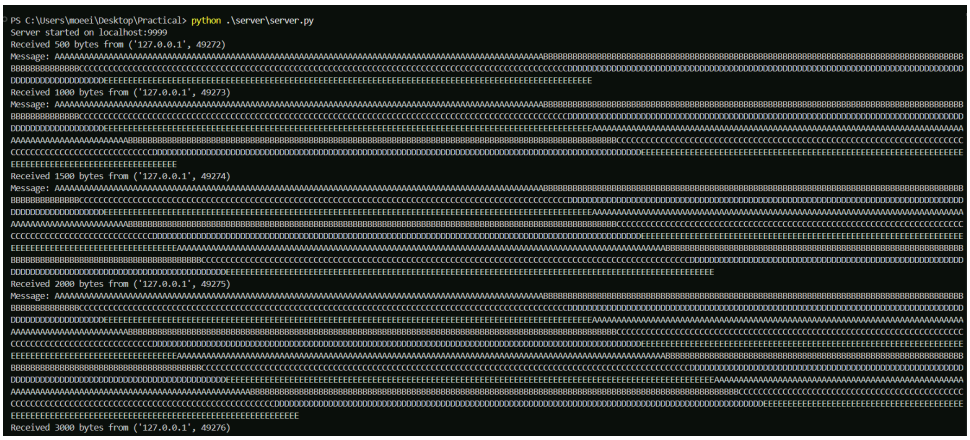
در این بخش صرفا خروجی کد و اسکرین‌شات‌های WireShark را قرار می‌دهیم. توضیحات کامل‌تر کد و نحوه پیاده‌سازی، داخل فایل REAMDE هر کد موجود است. در حل این سوال ناچارا از مدل‌های زبانی بزرگ استفاده شده است.

آ

ابتدا به صورت عادی پیام‌ها را ارسال می‌کنیم و می‌بینیم که به صورت Fragmented برای سرور ارسال شده است و دارای TTL درست است.



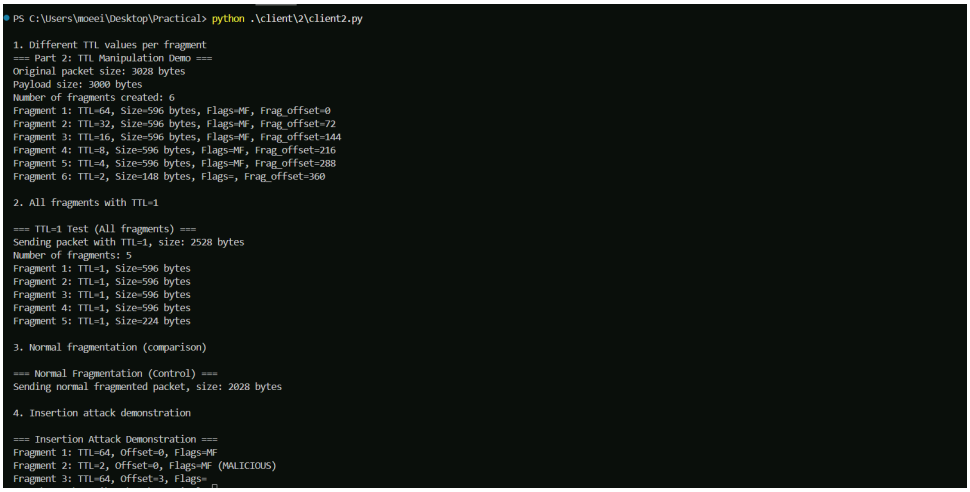
همچنین پیام‌ها در سرور به درستی تحویل داده شده‌اند:



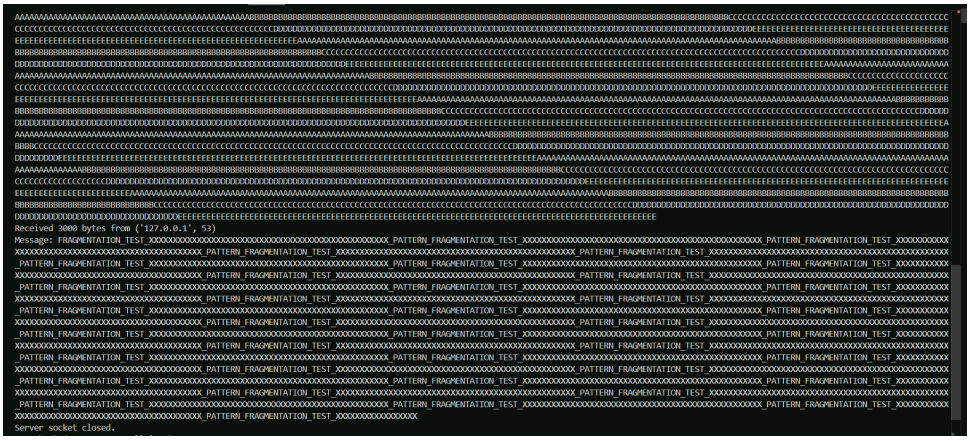
ب

در این بخش قرار است مقدار TTL را خودمان تنظیم کنیم و مقدار ثابت ۱ را برای هر فرگمنت قرار دهیم تا توسط فایروال سرور قابل ردیابی نباشد.

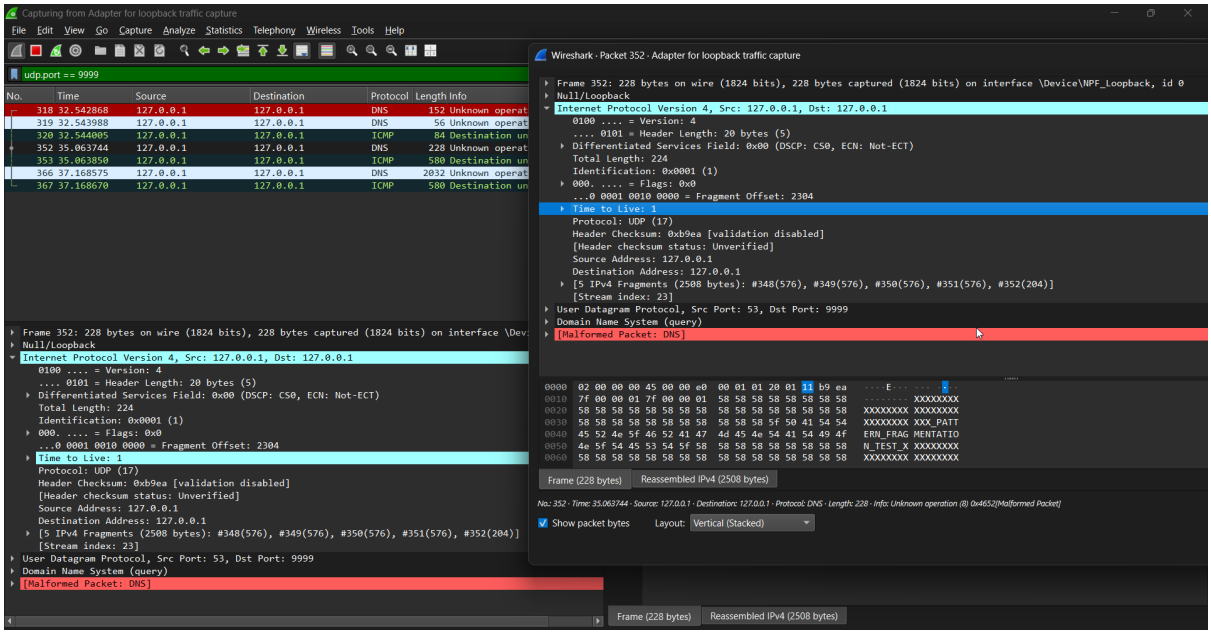
کد کلاینت را ران می‌کنیم. همانطور که در تصویر می‌بینید این کد دارای حالت‌های مختلف برای تست کردن است که توضیحات آن در فایل README موجود است.



تاییدیه دریافت پیام در سرور:



در Wireshark به دلیل اینکه TTL برابر با ۱ ست شده، این بسته را قرمز نشان داده است. اما در عکس ترمینال سرور مشخص است که بست به درستی به دست سرور رسیده است:


Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.port == 9999
No. Time Source Destination Protocol Length Info
318 32.542868 127.0.0.1 127.0.0.1 DNS 152 Unknown operat
319 32.543988 127.0.0.1 127.0.0.1 DNS 56 Unknown operat
320 32.544005 127.0.0.1 127.0.0.1 ICMP 84 Destination un
352 35.063744 127.0.0.1 127.0.0.1 DNS 228 Unknown operat
353 35.063850 127.0.0.1 127.0.0.1 ICMP 580 Destination un
366 37.168575 127.0.0.1 127.0.0.1 DNS 2032 Unknown operat
367 37.168670 127.0.0.1 127.0.0.1 ICMP 580 Destination un
Frame 352: 228 bytes on wire (1824 bits), 228 bytes captured (1824 bits) on interface \Dev
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 224
Identification: 0x0001 (1)
000. .... = Flags: 0x0
...0 0001 0010 0000 = Fragment Offset: 2304
Time to Live: 1
Protocol: UDP (17)
Header Checksum: 0xb9ea [validation disabled]
[Header checksum status: Unverified]
Source Address: 127.0.0.1
Destination Address: 127.0.0.1
[5 IPv4 Fragments (2508 bytes): #348(576), #349(576), #350(576), #351(576), #352(204)]
[Stream index: 23]
User Datagram Protocol, Src Port: 53, Dst Port: 9999
Domain Name System (query)
[Malformed Packet: DNS]
Frame (228 bytes)
Reassembled IPv4 (2508 bytes)
Wireshark · Packet 352 · Adapter for loopback traffic capture
Frame 352: 228 bytes on wire (1824 bits), 228 bytes captured (1824 bits) on interface \Device\NPF_Loopback, id 0
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 224
Identification: 0x0001 (1)
000. .... = Flags: 0x0
...0 0001 0010 0000 = Fragment Offset: 2304
Time to Live: 1
Protocol: UDP (17)
Header Checksum: 0xb9ea [validation disabled]
[Header checksum status: Unverified]
Source Address: 127.0.0.1
Destination Address: 127.0.0.1
[5 IPv4 Fragments (2508 bytes): #348(576), #349(576), #350(576), #351(576), #352(204)]
[Stream index: 23]
User Datagram Protocol, Src Port: 53, Dst Port: 9999
Domain Name System (query)
[Malformed Packet: DNS]
0000 02 00 00 00 45 00 00 e0 00 01 01 20 01 11 b9 ea
0010 7f 00 00 01 7f 00 00 01 58 58 58 58 58 58 58 58
0020 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58
0030 58 58 58 58 58 58 58 58 58 58 58 5f 50 41 54 54
0040 45 52 4e 5f 46 52 41 47 4d 45 4e 54 41 54 49 4f
0050 4e 5f 54 45 53 54 5f 58 58 58 58 58 58 58 58 58
0060 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58
Frame (228 bytes)
Reassembled IPv4 (2508 bytes)
No.: 352 · Time: 35.063744 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: DNS · Length: 228 · Info: Unknown operation (8) 0x4652[Malformed Packet]
Show packet bytes
Layout: Vertical (Stacked)
Frame (228 bytes)
Reassembled IPv4 (2508 bytes)