מטלה 5 – רשתות תקשורת

מגישות :

- **אופק כץ 315138693**
- **מוריה אסתר אוחיון 322953308**

תוכן עניינים:

TASK A

<u>מטרת המטלה:</u>

בחלק זה התבקשנו לכתוב sniffer משלנו ללכידת פקטות TCP.

כל פקטה שלכדנו נכתוב לתוך קובץ טקסט בשם " 315138693_322953308"

המידע הנוסף של הפקטה יהיה כהקסדצימאלי.

לבדיקה- נריץ את מטלה 2.

<u>הסבר על הרצת התוכנית:</u>

<u>נדרש:</u>

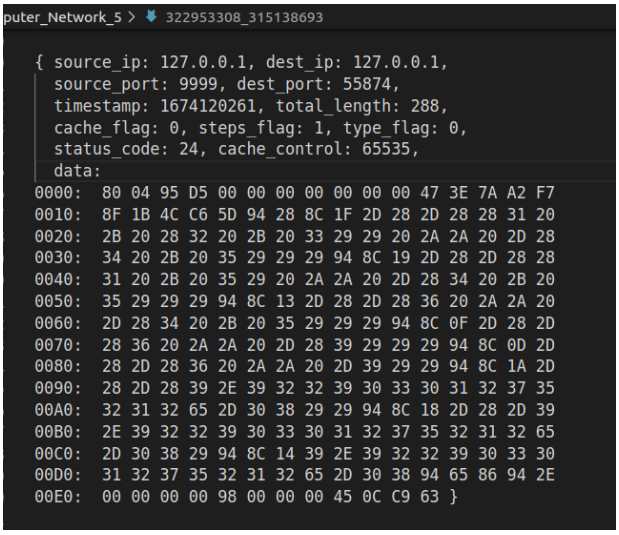
1. קובץ sniffing.c
2. מטלה 2:
   a. קובץ client.py
   b. קובץ proxy.py
   c. קובץ server.py

<u>הרצת התוכנית:</u>

1. פתיחת טרמינל (בתיקייה בה נמצא הסניפר)
2. קימפול הקוד: gcc -o snif ./sniffing.c -lpcap
3. הרצת הקוד: sudo ./snif
4. כעת נפעיל את מטלה 2:
   a. Python3 server.py
   b. Python3 proxy.py
   c. Python3 client.py או python3 client.py -p 9998
5. פקטות הTCP שיתקבלו ע"י הרצת הלקוח (לשרת או לפרוקסי) ירשמו לקובץ חדש שיפתח (או יידרס אם קיים כבר) בשם " 315138693_322953308".

<u>צילומי מסך:</u>

מהקובץ:



```
puter_Network_5 > 322953308_315138693

{ source_ip: 127.0.0.1, dest_ip: 127.0.0.1,
  source_port: 9999, dest_port: 55874,
  timestamp: 1674120261, total_length: 288,
  cache_flag: 0, steps_flag: 1, type_flag: 0,
  status_code: 24, cache_control: 65535,
  data:
0000:  80 04 95 D5 00 00 00 00 00 00 00 47 3E 7A A2 F7
0010:  8F 1B 4C C6 5D 94 28 8C 1F 2D 28 2D 28 28 31 20
0020:  2B 20 28 32 20 2B 20 33 29 29 20 2A 2A 20 2D 28
0030:  34 20 2B 20 35 29 29 29 94 8C 19 2D 28 2D 28 28
0040:  31 20 2B 20 35 29 20 2A 2A 20 2D 28 34 20 2B 20
0050:  35 29 29 29 94 8C 13 2D 28 2D 28 36 20 2A 2A 20
0060:  2D 28 34 20 2B 20 35 29 29 29 94 8C 0F 2D 28 2D
0070:  28 36 20 2A 2A 20 2D 28 39 29 29 29 94 8C 0D 2D
0080:  28 2D 28 36 20 2A 2A 20 2D 39 29 29 94 8C 1A 2D
0090:  28 2D 28 39 2E 39 32 32 39 30 33 30 31 32 37 35
00A0:  32 31 32 65 2D 30 38 29 29 94 8C 18 2D 28 2D 39
00B0:  2E 39 32 32 39 30 33 30 31 32 37 35 32 31 32 65
00C0:  2D 30 38 29 94 8C 14 39 2E 39 32 32 39 30 33 30
00D0:  31 32 37 35 32 31 32 65 2D 30 38 94 65 86 94 2E
00E0:  00 00 00 00 98 00 00 00 45 0C C9 63 }
```

מהווירשארק (הקלטה TASKA.pcapng): (לקובץ נכנסו רק הפקטות עם המידע- [PSH])

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	55874 → 9999 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=2583966255 TSecr=0 WS=128
2	0.000020615	127.0.0.1	127.0.0.1	TCP	76	9999 → 55874 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2583966256 TSecr=2583966255 WS=128
3	0.000032477	127.0.0.1	127.0.0.1	TCP	68	55874 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2583966256 TSecr=2583966256
4	0.000469207	127.0.0.1	127.0.0.1	TCP	574	55874 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=506 TSval=2583966256 TSecr=2583966256 [TCP segment of a reassembled
5	0.000476478	127.0.0.1	127.0.0.1	TCP	68	9999 → 55874 [ACK] Seq=1 Ack=507 Win=65024 Len=0 TSval=2583966256 TSecr=2583966256
6	0.007756534	127.0.0.1	127.0.0.1	TCP	304	9999 → 55874 [PSH, ACK] Seq=1 Ack=507 Win=65536 Len=236 TSval=2583966263 TSecr=2583966256 [TCP segment of a reassembl
7	0.007784240	127.0.0.1	127.0.0.1	TCP	68	55874 → 9999 [ACK] Seq=507 Ack=237 Win=65408 Len=0 TSval=2583966263 TSecr=2583966263
8	0.007807050	127.0.0.1	127.0.0.1	TCP	68	9999 → 55874 [FIN, ACK] Seq=237 Ack=507 Win=65536 Len=0 TSval=2583966263 TSecr=2583966263
9	0.009085220	127.0.0.1	127.0.0.1	TCP	68	55874 → 9999 [FIN, ACK] Seq=507 Ack=238 Win=65536 Len=0 TSval=2583966265 TSecr=2583966263
10	0.009105021	127.0.0.1	127.0.0.1	TCP	68	9999 → 55874 [ACK] Seq=238 Ack=508 Win=65536 Len=0 TSval=2583966265 TSecr=2583966265
11	1.911904721	127.0.0.1	127.0.0.1	TCP	76	55890 → 9999 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=2583968167 TSecr=0 WS=128
12	1.911916918	127.0.0.1	127.0.0.1	TCP	76	9999 → 55890 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2583968167 TSecr=2583968167 WS=128
13	1.911926193	127.0.0.1	127.0.0.1	TCP	68	55890 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2583968167 TSecr=2583968167
14	1.912361177	127.0.0.1	127.0.0.1	TCP	574	55890 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=506 TSval=2583968168 TSecr=2583968167 [TCP segment of a reassembled

תשובות לשאלות מחקר:

בשביל להסניף פקטות, נצטרך להיכנס לתוך תעבורת הרשת, דבר שמצריך אישור של מנהל המערכת. כשהרצנו את תכנית הסניפר, הרצנו עם sudo , מה שאומר למערכת שאנחנו בעלי הרשאת root וכך נוכל להיכנס לממשק הרשת ולהסניף תעבורת רשת.

ללא הרשאת מנהל מערכת, התוכנית תיפול בניסיון שלה להסניף פקטות מתוך תעבורת הרשת.

היכולת של הסניפר היא רק לקרוא פקטות שמגיעות לכרטיס הרשת שלו. אין באפשרותו לערוך או/וגם לשנות את הפקטות אותן הוא קרא.

TASK B

מטרת המטלה:

בחלק זה של המטלה נתבקשנו לכתוב קוד spoofer שמסוגל לזייף פקטות מסוג ICMP. זאת אומרת, ישלח פינג לכתובת 10.0.2.15 (המחשב שלי) בשם כתובת אחרת לדוגמא (8.8.8.8).

בנוסף ניתן לשנות את הקוד שלנו לזיוף פקטות TCP או UDP בשינויים קלים.

הסבר על הרצת התוכנית:

נדרש:

1. קובץ spoofing.c

הרצת התוכנית:

1. פתיחת טרמינל (בתיקייה בה נמצא הספופר)
2. קימפול הקוד: gcc -o spof ./spoofing.c
3. הרצת הקוד: sudo ./spof
4. ישלח פינג מ"8.8.8.8" למחשב שלנו "10.0.2.15" והמחשב לבד יחזיר פונג. – כלומר זייפנו פקטת פינג.
5. לשינוי שליחת פקטה בפרוטוקול UDP/TCP: נפעיל את הפונקציה הרצוייה. (כעת מופעל ICMP בלבד) צילום מסך מהקוד:

```
//icmp
icmp(buffer,ip);

//udp
//udp(buffer,ip);

//tcp
//tcp(buffer, ip);
```

צילומי מסך:

ווירשארק (הקלטה TASKB.pcapng): שורה 1/ 6 – הפקטה אותה זייפנו

icmp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	8.8.8.8	10.0.2.15	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=20 (reply in 2)
2	0.000030915	10.0.2.15	8.8.8.8	ICMP	44	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 1)
6	5.886336588	8.8.8.8	10.0.2.15	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=20 (reply in 7)
7	5.886348024	10.0.2.15	8.8.8.8	ICMP	44	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 6)

כאשר כלל הפונקציות מופעלות (ווירשארק הקלטה TASK_B-ICMP_UDP_TCP.pcapng):

שורה 1- זיוף פינג, שורה 3 – זיוף UDP , שורה 5- זיוף TCP.

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	8.8.8.8	10.0.2.15	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=20 (reply in 2)
2	0.000037592	10.0.2.15	8.8.8.8	ICMP	44	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 1)
3	0.006812809	8.8.8.8	10.0.2.15	UDP	58	12345 → 9090 Len=14
4	0.006838678	10.0.2.15	8.8.8.8	ICMP	86	Destination unreachable (Port unreachable)
5	0.006929717	8.8.8.8	10.0.2.15	TCP	74	12345 → 9090 [URG, NS, Reserved] Seq=1 Win=21349 Urg=25970 Len=14

<u>תשובות לשאלות מחקר:</u>

1. שדה אורך החבילה משמש לציון האורך הכולל של הפקטה בבתים, כולל הכותרת (headers) והנתונים (data).
   אם נגדיר את השדה לערך שרירותי, זה עלול לגרום לבעיות ברשת:
   אם השדה מוגדר לערך הגדול מהגודל האמיתי של החבילה, הדבר עלול לגרום להפלת החבילה, ואם השדה מוגדר לערך הקטן מהגודל האמיתי של החבילה, הדבר עלול לגרום לשליחת רק חלק מהחבילה, וכתוצאה מכך לשליחת נתונים חסרים או פגומים.
2. חישוב הchecksum נעשה בפרוטוקולים אמינים כדי לוודא שהמידע עבר כמו שצריך.
   raw socket אינו פרוטוקול אמין ולכן אין צורך בחישוב הchecksum.

היכולת של הספופר היא רק לזייף פקטות ולשלוח אותן, אין ביכולותו לקרוא פקטות שמגיעות לכרטיס הרשת שלו.

TASK C

<u>מטרת המטלה:</u>

בחלק זה של המטלה התבקשנו להסניף פקטת ICMP request (שלא הייתה מיועדת אלינו) ולשלוח ICMP reply לכתובת שאליה היה מיועד הICMP reply המקורי. לשם כך הרמנו שלושה דוקרים: Host A, Host B וattacker.

שאלה זו התחלקה לשלוש: א. Host A שולח פינג לHost B

ב. Host A שולח פינג לip שקיים ברשת (8.8.8.8)

ג. Host A שולח פינג לip שאינו מחובר (1.2.3.4)

<u>הסבר על הרצת התוכנית</u>:

<u>נדרש</u>:

1. קובץ sniff_and_spof.c
2. קובץ ping.c (מטלה 4)

<u>הרצת התוכנית</u>:

1. הרמת דוקרים
2. בattacker :
   a. נקמפל את הקוד(אנחנו כבר בroot):
   gcc -o spoof_new sniff_and_spof.c -lpcap
   b. נריץ: ./spoof_new.
3. בhost A :
   a. הרצת מטלה 4: gcc -o ping ping.c
   b. הרצת הקוד(אנחנו כבר בroot):
   א. ./ping "HOST B IP"(10.9.0.6).

   Host A שולח פינג לHost B, הattacker יסניף את פקטת הפינג. הattacker יצור בעצמו פקטת פונג, כאשר נחליף בין הip source לבין הip dest ונשמור על הseq_num וid של הicmp . כך שישלח לHost A פונג בשם Host B שבעצם אנחנו שלחנו.

   ב. ./ping 8.8.8.8.

   Host A שולח פינג לכתובת ip קיימת ברשת (8.8.8.8) הattacker יסניף את פקטת הפינג. הattacker יצור בעצמו פקטת פונג, כאשר נחליף בין הip source לבין הip dest ונשמור על הseq_num וid של הicmp . כך שישלח לHost A פונג בשם 8.8.8.8 שבעצם אנחנו שלחנו.

   ג. ./ping 1.2.3.4.

   Host A שולח פינג לכתובת ip שלא קיימת ברשת (1.2.3.4) הattacker יסניף את פקטת הפינג. הattacker יצור בעצמו פקטת פונג, כאשר נחליף בין הip source לבין הip dest ונשמור על

המseq_num וid של הicmp . כך שישלח לA Host פונג בשם 1.2.3.4
שבעצם אנחנו שלחנו. (על אף שהip אינו ולידי).

צילומי מסך:

ויירשארק (הקלטה TASKC.pcapng): תמונה כללית (- הסבר מפורט מטה)

icmp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.9.0.5	10.9.0.6	ICMP	61	Echo (ping) request id=0x1200, seq=0/0, ttl=64 (reply in 2)
2	0.000076622	10.9.0.6	10.9.0.5	ICMP	61	Echo (ping) reply id=0x1200, seq=0/0, ttl=64 (request in 1)
3	0.590240317	10.9.0.6	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
4	1.000791653	10.9.0.5	10.9.0.6	ICMP	61	Echo (ping) request id=0x1200, seq=256/1, ttl=64 (reply in 5)
5	1.000833605	10.9.0.6	10.9.0.5	ICMP	61	Echo (ping) reply id=0x1200, seq=256/1, ttl=64 (request in 4)
6	1.594611479	10.9.0.6	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=256/1, ttl=20
7	2.001486968	10.9.0.5	10.9.0.6	ICMP	61	Echo (ping) request id=0x1200, seq=512/2, ttl=64 (reply in 8)
8	2.001573325	10.9.0.6	10.9.0.5	ICMP	61	Echo (ping) reply id=0x1200, seq=512/2, ttl=64 (request in 7)
9	2.621709612	10.9.0.6	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=512/2, ttl=20
16	6.017139639	10.9.0.5	8.8.8.8	ICMP	61	Echo (ping) request id=0x1200, seq=0/0, ttl=64 (reply in 17)
17	6.043716895	8.8.8.8	10.9.0.5	ICMP	61	Echo (ping) reply id=0x1200, seq=0/0, ttl=117 (request in 16)
18	6.714275499	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
19	7.051630732	10.9.0.5	8.8.8.8	ICMP	61	Echo (ping) request id=0x1200, seq=256/1, ttl=64 (reply in 20)
20	7.070899265	8.8.8.8	10.9.0.5	ICMP	61	Echo (ping) reply id=0x1200, seq=256/1, ttl=117 (request in 19)
21	7.738048321	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=256/1, ttl=20
22	8.051928710	10.9.0.5	8.8.8.8	ICMP	61	Echo (ping) request id=0x1200, seq=512/2, ttl=64 (reply in 23)
23	8.078875901	8.8.8.8	10.9.0.5	ICMP	61	Echo (ping) reply id=0x1200, seq=512/2, ttl=117 (request in 22)
24	8.761896475	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=512/2, ttl=20
27	13.221206671	10.9.0.5	1.2.3.4	ICMP	61	Echo (ping) request id=0x1200, seq=0/0, ttl=64 (no response found!)
28	13.890435255	1.2.3.4	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
29	14.894735860	10.9.0.5	1.2.3.4	ICMP	61	Echo (ping) request id=0x1200, seq=256/1, ttl=64 (no response found!)
30	14.906134130	1.2.3.4	10.9.0.5	ICMP	42	Echo (ping) reply id=0x1200, seq=256/1, ttl=20

א. ניתן לראות בשורה 1 שליחת פינג מ host A לhost B
מיד לאחר מכן בשורה 2 יש את הפונג המקורי שB החזיר לA
בשורה 3 – רואים את הפונג שהattacker החזיר (כביכול מB לA)

1	0.000000000	10.9.0.5	10.9.0.6	ICMP	61 Echo (ping) request
2	0.000076622	10.9.0.6	10.9.0.5	ICMP	61 Echo (ping) reply
3	0.590240317	10.9.0.6	10.9.0.5	ICMP	42 Echo (ping) reply

ב. שורה 16- שליחת פינג מA ל8.8.8.8
שורה 17- תשובת פונג מקורית מ8.8.8.8
שורה 18- הפונג שהattacker החזיר (כביכול מ8.8.8.8 לA)

16	6.017139639	10.9.0.5	8.8.8.8	ICMP	61 Echo (ping) request
17	6.043716895	8.8.8.8	10.9.0.5	ICMP	61 Echo (ping) reply
18	6.714275499	8.8.8.8	10.9.0.5	ICMP	42 Echo (ping) reply

ג. שורה 27- שליחת פינג מA ל1.2.3.4 (אין תגובה מקורית כי 1.2.3.4 לא מחובר)
שורה 28- הפונג שהattacker החזיר (כביכול מ1.2.3.4 לA)

27	13.221206671	10.9.0.5	1.2.3.4	ICMP	61 Echo (ping) request
28	13.890435255	1.2.3.4	10.9.0.5	ICMP	42 Echo (ping) reply

TASK D

מטרת המטלה:

בחלק זה של המטלה נתבקשנו לכתוב קוד שישמש כGateway המבקש מהמשתמש ip אליו ירצה לשלוח פקטה.

הGateway יהיה מסוגל לקבל פקטות UDP דרך פורט P שבחרנו (10000).

הGateway יגריל מספר בין 0-1. במידה והמספר קטן מ0.5 הפקטה תלך לאיבוד ואם זה גדול מ0.5 הGateway ישלח הלאה את הפקטה לip שהמשתמש הכניס, דרך פורט P+1 (10001)

כך למעשה, נאבד כ 50% מהפקטות שהגיעו אלינו (לא נשלח אותן).

הסבר על הרצת התוכנית:

נדרש:

1. קובץ gateway.c

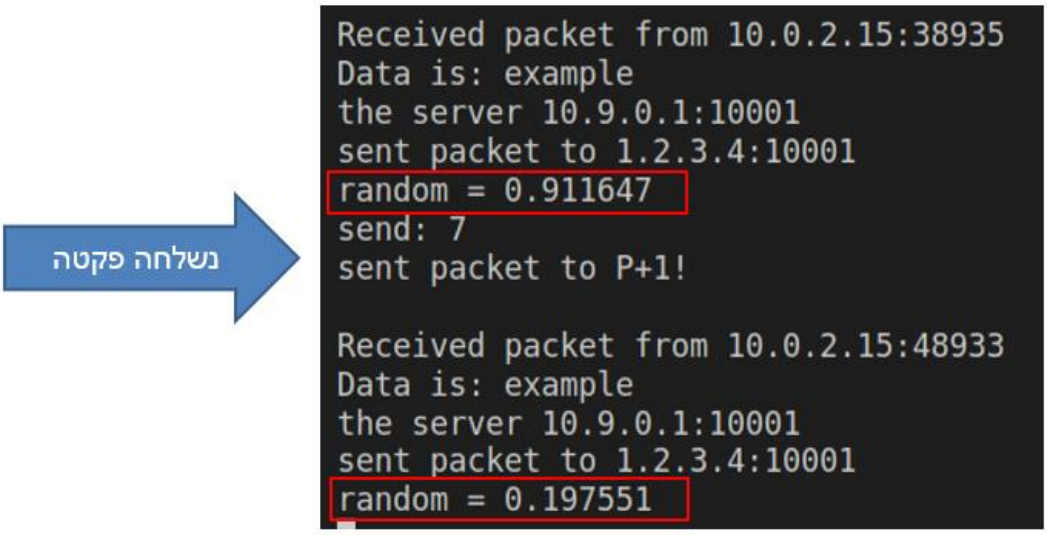
הרצת התוכנית:

1. נקמפל את הקוד: gcc -o gateway gateway.c
2. הרצת הקוד: ./ gateway <ip>
3. ה ip שהוכנס יהיה הIP אליו נשלח.
4. ברגע שתשלח פקטת UDP לשרת בפורט P (10000) השרת יגריל מספר ואם יצא מספר קטן מ0.5 תשלח אותה פקטה שהגיעה אל הIP שהוכנס (בהרצת הקוד) דרך פורט P +1 (10001)

צילומי מסך:

צילומי הדפסות: השרת שלנו- 10.9.0.1 ,השרת ששלח אלינו UDP- 10.0.2.15 , הIP שהכנסנו בהרצת התוכנית- 1.2.3.4

ניתן לראות שכאשר המספר הרנדומלי גדול מ0.5 שלחנו פקטה לעומת כאשר המספר הרנדומלי היה קטן מ0.5 לא שלחנו כלום:



ויירשארק (הקלטה TASKD.pcapng):

ניתן לראות שבשורה 1 נשלח UDP אל השרת שלנו בפורט 10000 (P) והשרת לא העביר את הפקטה שוב.

לעומת שורה 2 ששם נשלח UDP אל השרת שלנו בפורט 10000 (P) ובשורה 3 השרת העביר את אותה פקטה (forward) ל ip שהכנסנו אליו (1.2.3.4) לפורט 10001:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	10.9.0.1	UDP	51	52139 → 10000 Len=7
2	1.869888420	10.0.2.15	10.9.0.1	UDP	51	33047 → 10000 Len=7
3	1.869952523	10.0.2.15	1.2.3.4	UDP	51	10001 → 10001 Len=7
4	4.093201256	10.0.2.15	10.9.0.1	UDP	51	46509 → 10000 Len=7
5	6.096384606	10.0.2.15	10.9.0.1	UDP	51	35417 → 10000 Len=7
6	6.096442174	10.0.2.15	1.2.3.4	UDP	51	10001 → 10001 Len=7
9	8.840015193	10.0.2.15	10.9.0.1	UDP	51	50140 → 10000 Len=7