به نام خدا

گزارش کار گروه E

حامد نظریان
(976127033)

موضوع پروژه:

- ✓برنامه شنود و آنالیز شبکه
- ✓Packet Sniffer (mini wireshark) - Packet Capture

توجه برنامه به طور اولیه وِیژه محیط **لینوکس** طراحی شد، در انتها پشتیبانی از ویندوز در آن قرار داده شد ولی هیچ گونه تستی انجام نشده.

توجه در محیط لینوکس برنامه **باید** با **دسترسی** سطح **ادمین (sudo)** اجرا شود.

➢ انتخاب موضوع

- ایمیل اول

Thu, Jun 30, 2:28 PM (4 days ago)

شبتون بخیر استاد
استاد موضوع پکت اسنیفر رو تایید می کنید دیگه همین ایمیل رو بیارم تو اسناد تحویل پروژه
پکت اسنیفر کلاینت و سرور نداره، ولی شدید با قابلیت های کتابخونه سوکت و اینترفیسی که از سیستم عامل میگیره کار میکنه
مخصوصا وقتی از
raw_socket
،استفاده می کنم، و تو لایه های مختلف یه فریم (بعد پکت و سگمنت) رو اطلاعاتش میخونم
.خیلیی بیشتر از حالت کلاینت سرور معمولی از کدها و مفاهیم کتابخونه استفاده می کنه
و ترافیک رو به فانکشن های مختلف این کتابخونه پاس میده

لازم به ذکر تو حالت اولیه پروکسی من از کلاینت سرور استفاده میکردم
با تشکر

- ایمیل دوم از جانب شما

Nastooh Taheri Jun 30, 2022, 2:14 PM (4 days ago)
to me

سلام.
این توضیحات رو کامل توی داکیومنتتون بگین

==========
Yours truly Nastooh
==========
http://www.nastooh.com
==========

- ایمیل سوم

Hamed Nazarian <enzed.hm@gmai... Jun 30, 2022, 2:28 PM (4 days ago)
to Nastooh

شبتون بخیر استاد
استاد موضوع پکت اسنیفر رو تایید می کنید دیگه همین ایمیل رو بیارم تو اسناد تحویل پروژه
پکت اسنیفر کلاینت و سرور نداره، ولی شدید با قابلیت های کتابخونه سوکت و اینترفیسی که از سیستم عامل میگیره کار میکنه
مخصوصا وقتی از
raw_socket
،استفاده می کنم، و تو لایه های مختلف یه فریم (بعد پکت و سگمنت) رو اطلاعاتش میخونم
.خیلیی بیشتر از حالت کلاینت سرور معمولی از کدها و مفاهیم کتابخونه استفاده می کنه
و ترافیک رو به فانکشن های مختلف این کتابخونه پاس میده

لازم به ذکر تو حالت اولیه پروکسی من از کلاینت سرور استفاده میکردم
با تشکر

- من به طور اولیه یه پروکسی نوشتم که از یک کلاینت و یک سرور تشکیلی شده بود.
  - ✓ کلاینت برای دریافت پکت و باز کردنش، خوندن فیلد های مورد نیاز برای ارسال یا تحلیل پکت.
  - ✓ سرور برای ارسال مجدد پکت به مقصد اصلی

احتمالا تنها نکته برنامه اینه که برای بالا بردن سرعت برنامه باید از تکنولوژی رشته سازی در پردازش ها Threading استفاده کنیم، تا برنامه وقتی یه پکت گرفت، متوقف نشه و فرایند دریاف رو همچنان ادامه بده، درحالی که به منظور پردازش پکت دریافتی یه thread درست میکنه و طور موازی اجرا رو ادامه میده.

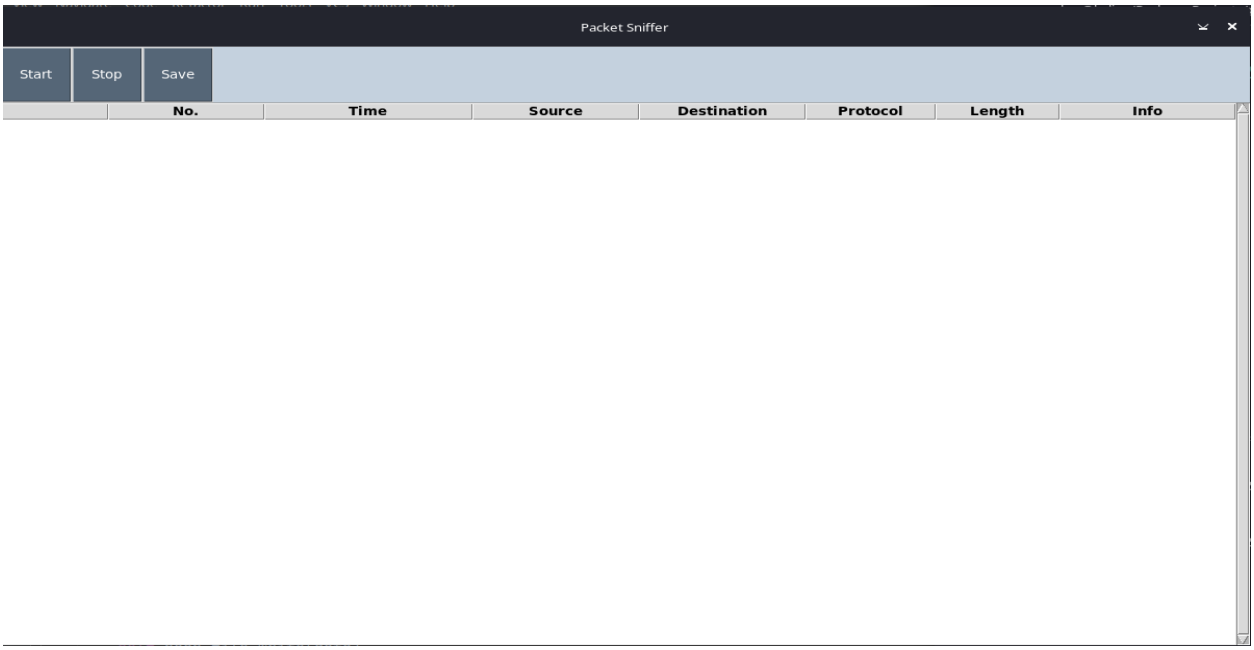
چالش های دیگه ای در مورد اندازه بافر و اینکه که چطوری پکت های دریافتی رو چون به صورت یه جریان ارسای میشن از هم واکاوی کنیم.

- اما وقتی من در مورد موضوعم از شما مشورت گرفتی به جنبه تحلیل اشاره کردید و گفتید که به برنامه فعلی پروتکل های دیگر لایه ها مختلف (پایین تر) شبکه رو اضافه کنم و به صورت زیبا با استفاده از گرافیک این رو نشون کاربر، بزارم و قابلیت ذخیر این پکت هارو هم اضافه کنم.
قابل توجه است برنامه اولیه من HTTP Proxy بود و فقط پکت های شنود شده (دریافتی و ارسالی) رو در سطح HTTP Header برسی میکرد.

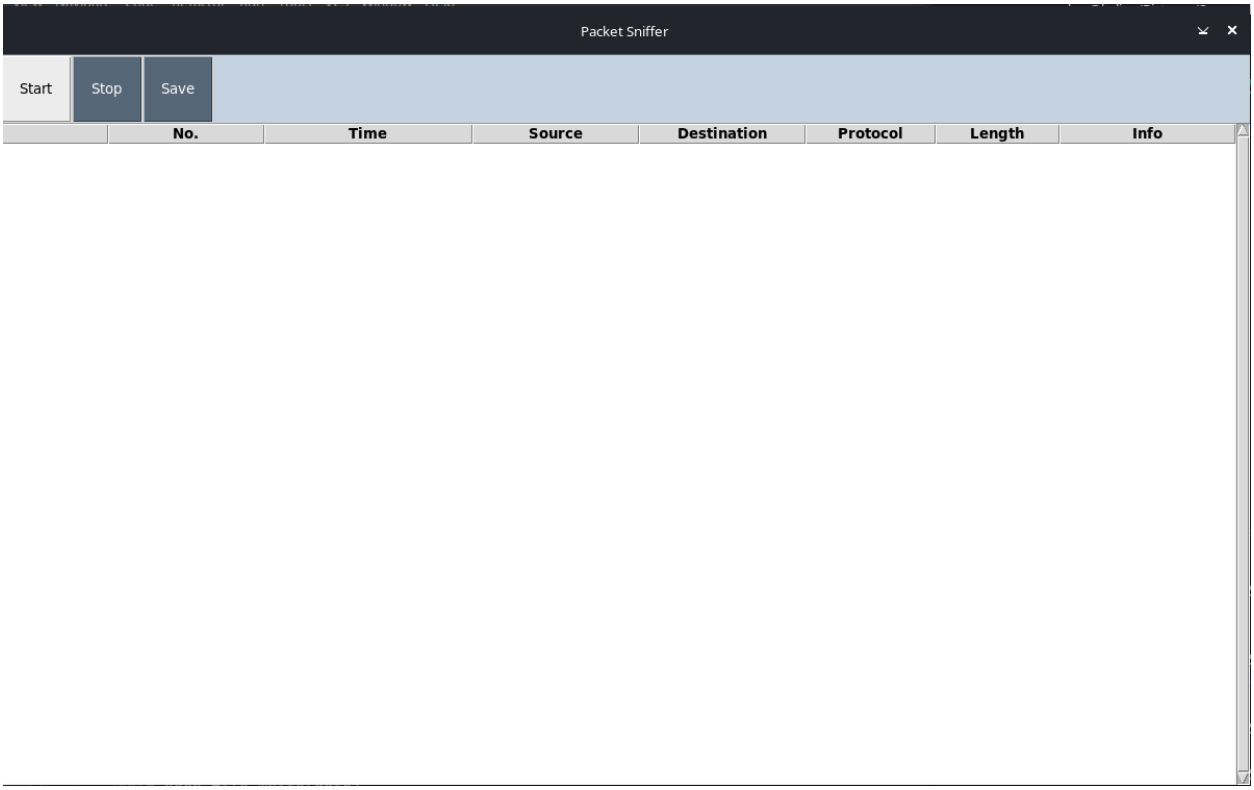
➢ شرح برنامه کنونی – Packet Capture

- برنامه کنونی دیگه از Interface کلاینت و سرور کتابخونه socket استفاده نمیکنه بلکه از قابلیت های بسیار، بیشتر، خاصر تر و پیشرفته تر مثل Raw Socket برای دریافت پکت تو لایه پایین و حرکت به لایه های بالایی و پردازششون انجام میده.
- از برنامه wireshark الهام گرفته شده، و Interface در مورادی شبیه اون طراحی شده.
- جزییات قابلیت ها:

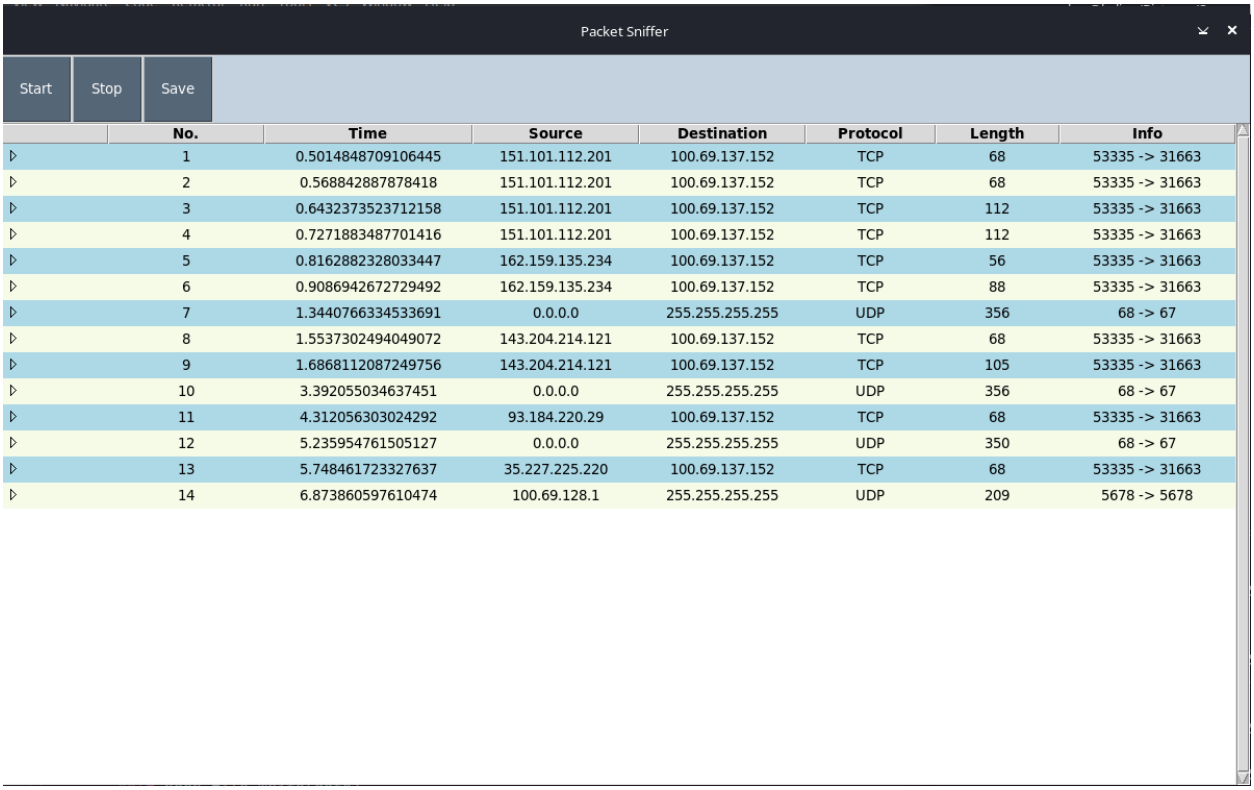
✓ برنامه یه محیط زیبا و ساده داره که سه گزینه Start, Stop, Save داره.



✓ با فشردن دکمه Start دریافت پکت ها به صورت real time شروع میشه.



- ✓ هر فریم دریافتی در قالب یک سطر ظاهر میشه که حاولی اطلاعات و حتی ظاهری بسیار شبیه چیزیه که wireshark استفاده می کنه.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.5014848709106445	151.101.112.201	100.69.137.152	TCP	68	53335 -> 31663
2	0.568842887878418	151.101.112.201	100.69.137.152	TCP	68	53335 -> 31663
3	0.6432373523712158	151.101.112.201	100.69.137.152	TCP	112	53335 -> 31663
4	0.7271883487701416	151.101.112.201	100.69.137.152	TCP	112	53335 -> 31663
5	0.8162882328033447	162.159.135.234	100.69.137.152	TCP	56	53335 -> 31663
6	0.9086942672729492	162.159.135.234	100.69.137.152	TCP	88	53335 -> 31663
7	1.3440766334533691	0.0.0.0	255.255.255.255	UDP	356	68 -> 67
8	1.5537302494049072	143.204.214.121	100.69.137.152	TCP	68	53335 -> 31663
9	1.6868112087249756	143.204.214.121	100.69.137.152	TCP	105	53335 -> 31663
10	3.392055034637451	0.0.0.0	255.255.255.255	UDP	356	68 -> 67
11	4.312056303024292	93.184.220.29	100.69.137.152	TCP	68	53335 -> 31663
12	5.235954761505127	0.0.0.0	255.255.255.255	UDP	350	68 -> 67
13	5.748461723327637	35.227.225.220	100.69.137.152	TCP	68	53335 -> 31663
14	6.873860597610474	100.69.128.1	255.255.255.255	UDP	209	5678 -> 5678

✓در نگاه اول اطلاعات زیر در هر سطر نمایش داده میشه.

No: شماره فریم از آغاز فرایند capture

Time: زمان دریافت فریم تا نانو ثانیه از آغاز فرایند capture

Source: آدرس IP فرستنده(آدرس مک فرستنده در مورد فریم هایی که لایه سه ندارند.)

Destination: آدرس IP گیرنده(آدرس مک گیرنده در مورد فریم هایی که لایه سه ندارند.)

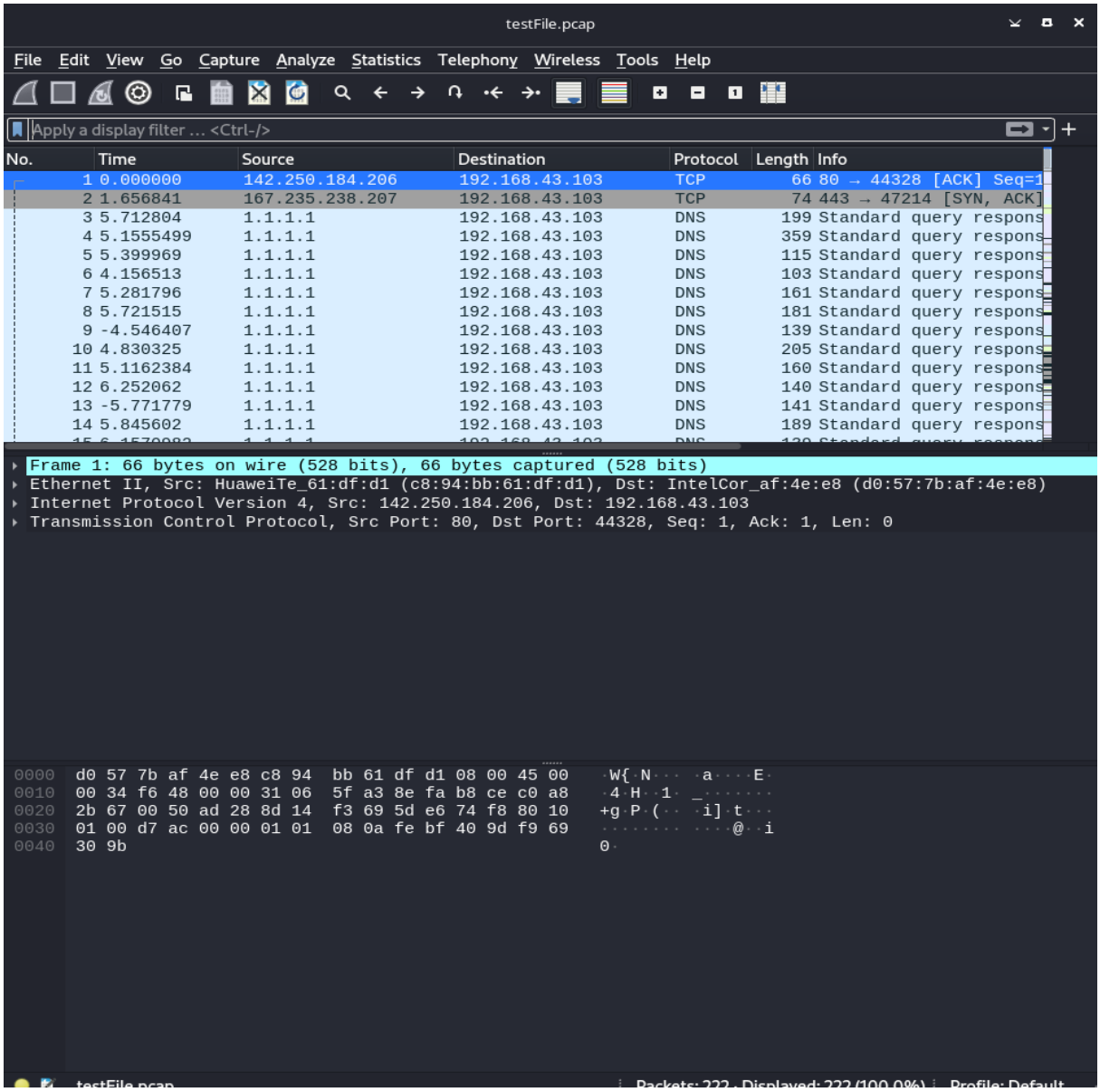
Protocol: پروتکل لایه چهار (لایه 3 اگه لایه چهار وجود نداره)

در حال حاضر فقط tcp, udp, icmp, icmpv6, arp ساپورت میشن.

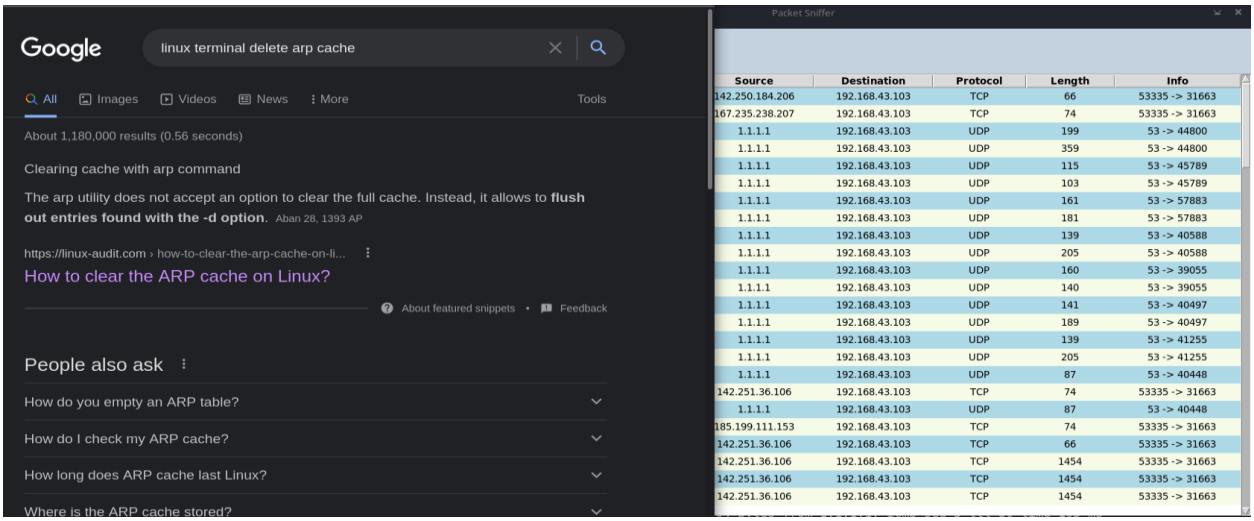
Length: طول خالص فریم دریافتی.

Info: اطلاعات لایه های بالاتر (تو برنامه من فقط پورت مبدا و مقصد)

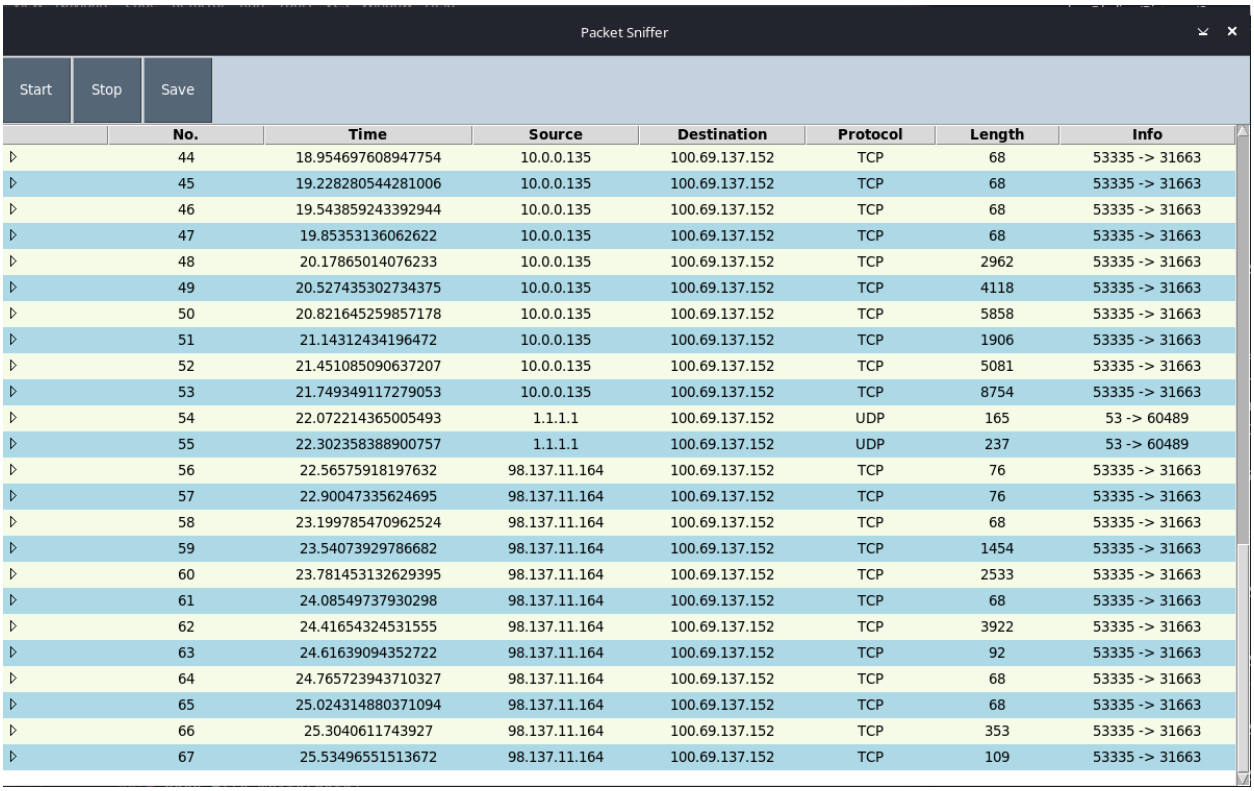
✓ Interface برنامه wireshark:



✓ یکم بیشتر ترافیک تولید می کنیم، با بازدید از چند صفحه وب و عملکرد برنامه رو مشاهده می کنیم.

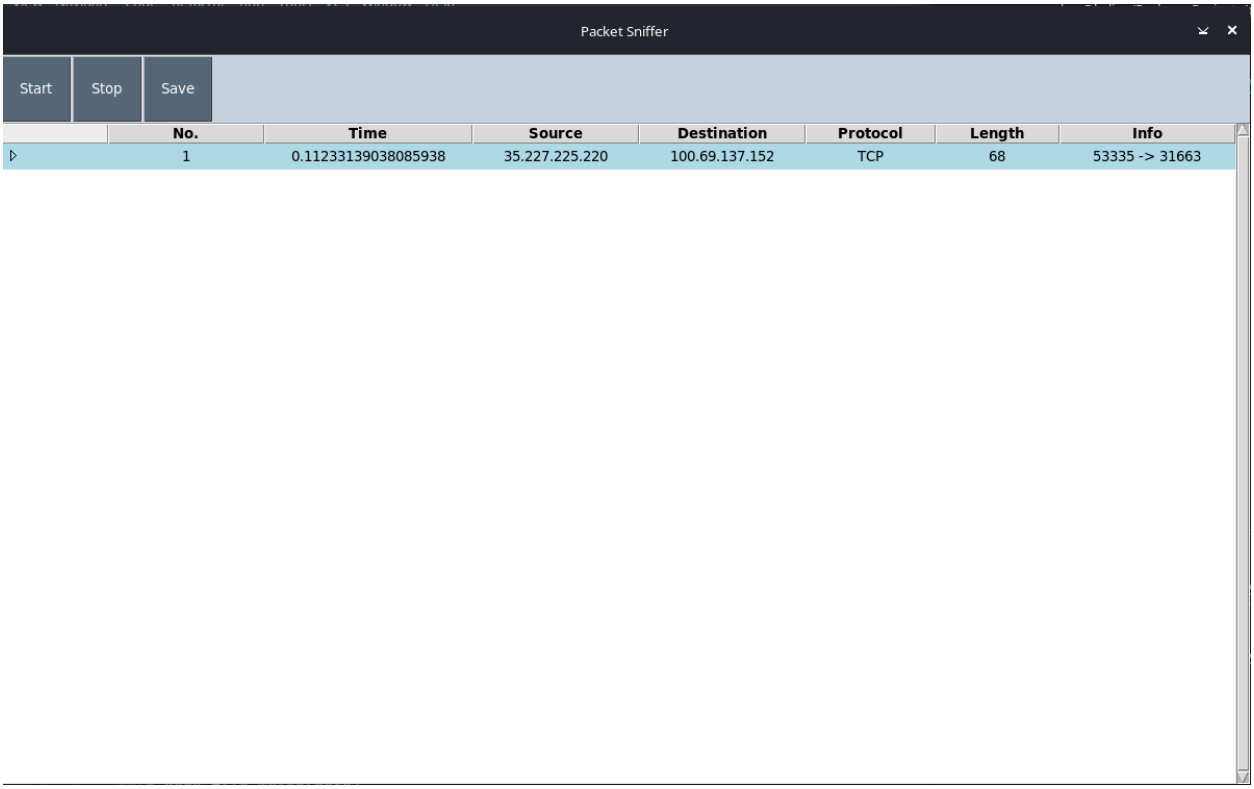


✓با فشردن دکمه Stop فرایند Capture متوقف می شود.

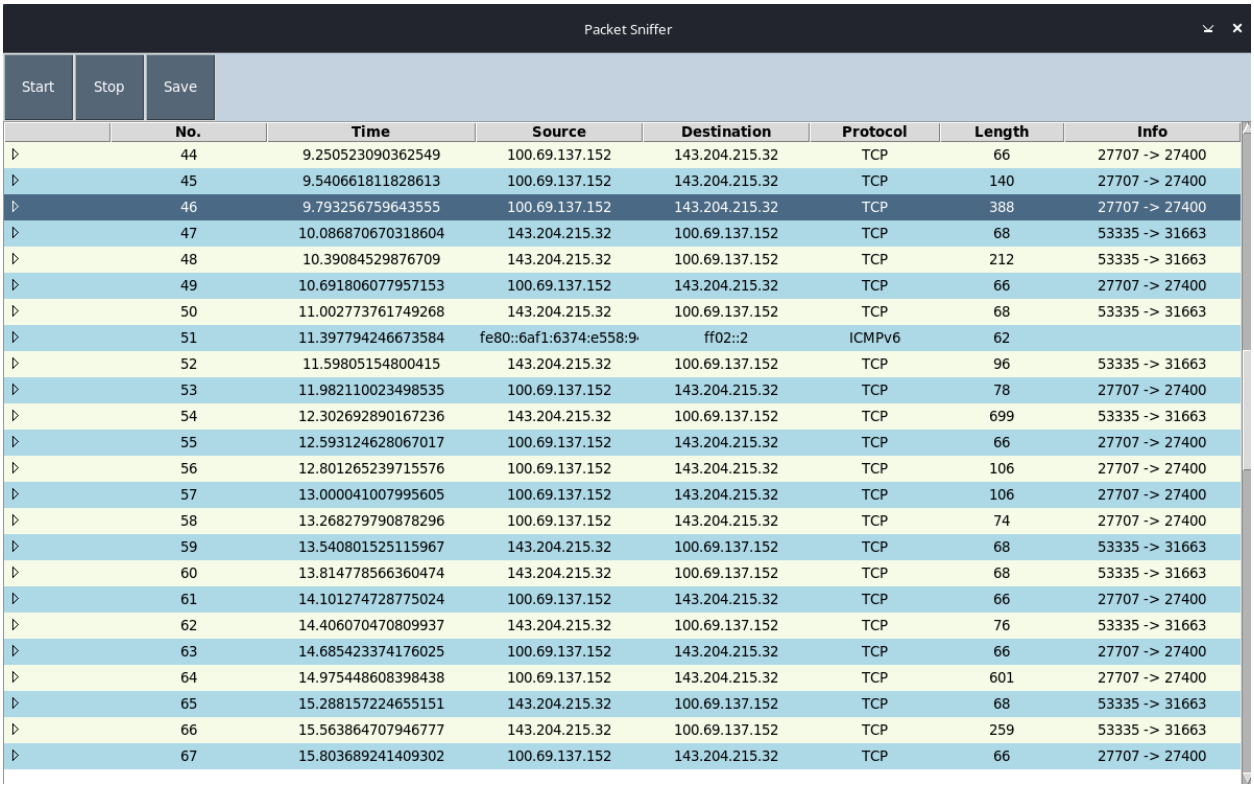


No.	Time	Source	Destination	Protocol	Length	Info
44	18.954697608947754	10.0.0.135	100.69.137.152	TCP	68	53335 -> 31663
45	19.228280544281006	10.0.0.135	100.69.137.152	TCP	68	53335 -> 31663
46	19.543859243392944	10.0.0.135	100.69.137.152	TCP	68	53335 -> 31663
47	19.85353136062622	10.0.0.135	100.69.137.152	TCP	68	53335 -> 31663
48	20.17865014076233	10.0.0.135	100.69.137.152	TCP	2962	53335 -> 31663
49	20.527435302734375	10.0.0.135	100.69.137.152	TCP	4118	53335 -> 31663
50	20.821645259857178	10.0.0.135	100.69.137.152	TCP	5858	53335 -> 31663
51	21.14312434196472	10.0.0.135	100.69.137.152	TCP	1906	53335 -> 31663
52	21.451085090637207	10.0.0.135	100.69.137.152	TCP	5081	53335 -> 31663
53	21.749349117279053	10.0.0.135	100.69.137.152	TCP	8754	53335 -> 31663
54	22.072214365005493	1.1.1.1	100.69.137.152	UDP	165	53 -> 60489
55	22.302358388900757	1.1.1.1	100.69.137.152	UDP	237	53 -> 60489
56	22.56575918197632	98.137.11.164	100.69.137.152	TCP	76	53335 -> 31663
57	22.90047335624695	98.137.11.164	100.69.137.152	TCP	76	53335 -> 31663
58	23.199785470962524	98.137.11.164	100.69.137.152	TCP	68	53335 -> 31663
59	23.54073929786682	98.137.11.164	100.69.137.152	TCP	1454	53335 -> 31663
60	23.781453132629395	98.137.11.164	100.69.137.152	TCP	2533	53335 -> 31663
61	24.08549737930298	98.137.11.164	100.69.137.152	TCP	68	53335 -> 31663
62	24.41654324531555	98.137.11.164	100.69.137.152	TCP	3922	53335 -> 31663
63	24.61639094352722	98.137.11.164	100.69.137.152	TCP	92	53335 -> 31663
64	24.765723943710327	98.137.11.164	100.69.137.152	TCP	68	53335 -> 31663
65	25.024314880371094	98.137.11.164	100.69.137.152	TCP	68	53335 -> 31663
66	25.3040611743927	98.137.11.164	100.69.137.152	TCP	353	53335 -> 31663
67	25.53496551513672	98.137.11.164	100.69.137.152	TCP	109	53335 -> 31663

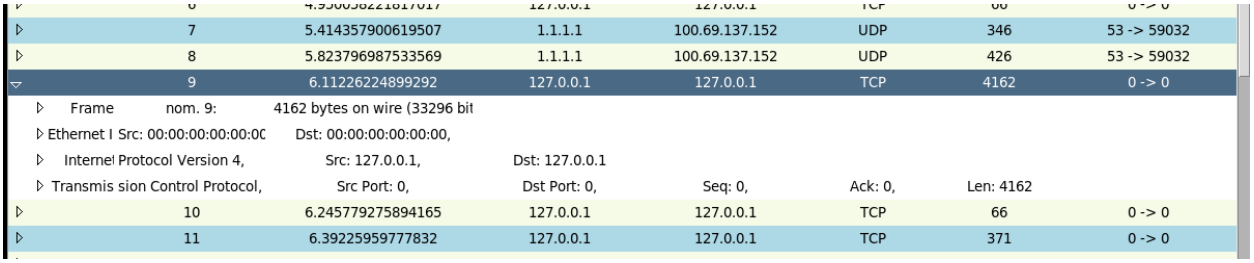
✓ حال با فشردن مجدد دکمه Start جدول Packet Capture کنونی را Flush می کند، همه سطر ها پاک شده و همه چی پاکسازی می شود، و Capture فریم ها از سرگیری می شود.



✓ هر سطر سمت چپ یه فلش کوچک داره که با کلیک اطلاعات لایه ها مختلفت رو خیلی عمیق تر و تو فرمتی خیلی شبیه وایرشارک نشون میده.
ابتدا پروتکل tcp رو مفصل و چند پروتکل دیگه رو خلاصه تر برسی می کنم.

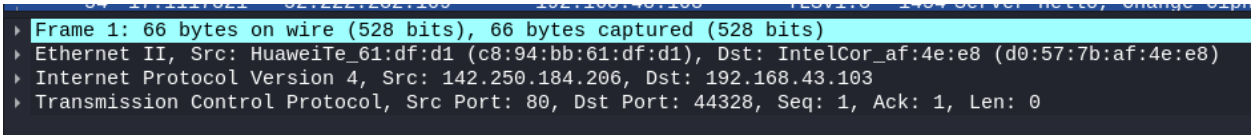


No.	Time	Source	Destination	Protocol	Length	Info
44	9.250523090362549	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
45	9.540661811828613	100.69.137.152	143.204.215.32	TCP	140	27707 -> 27400
46	9.793256759643555	100.69.137.152	143.204.215.32	TCP	388	27707 -> 27400
47	10.086870670318604	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
48	10.39084529876709	143.204.215.32	100.69.137.152	TCP	212	53335 -> 31663
49	10.691806077957153	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
50	11.002773761749268	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
51	11.397794246673584	fe80::6af1:6374:e558:9	ff02::2	ICMPv6	62	
52	11.59805154800415	143.204.215.32	100.69.137.152	TCP	96	53335 -> 31663
53	11.982110023498535	100.69.137.152	143.204.215.32	TCP	78	27707 -> 27400
54	12.302692890167236	143.204.215.32	100.69.137.152	TCP	699	53335 -> 31663
55	12.593124628067017	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
56	12.801265239715576	100.69.137.152	143.204.215.32	TCP	106	27707 -> 27400
57	13.000041007995605	100.69.137.152	143.204.215.32	TCP	106	27707 -> 27400
58	13.268279790878296	100.69.137.152	143.204.215.32	TCP	74	27707 -> 27400
59	13.540801525115967	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
60	13.814778566360474	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
61	14.101274728775024	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
62	14.406070470809937	143.204.215.32	100.69.137.152	TCP	76	53335 -> 31663
63	14.685423374176025	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
64	14.975448608398438	100.69.137.152	143.204.215.32	TCP	601	27707 -> 27400
65	15.288157224655151	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
66	15.563864707946777	143.204.215.32	100.69.137.152	TCP	259	53335 -> 31663
67	15.803689241409302	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400

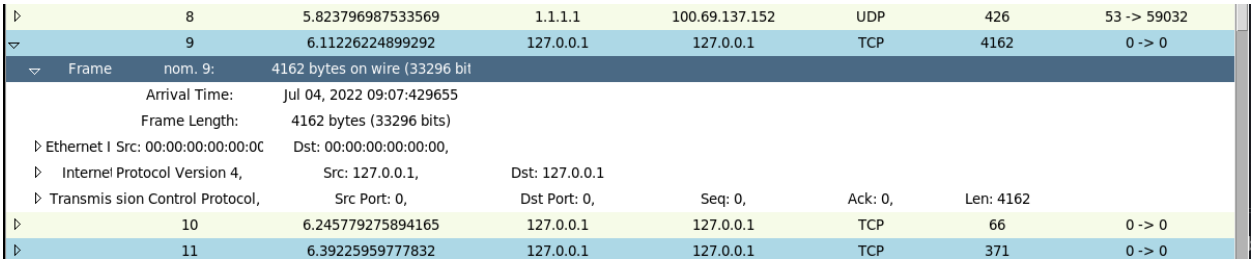


7	5.414357900619507	1.1.1.1	100.69.137.152	UDP	346	53 -> 59032
8	5.823796987533569	1.1.1.1	100.69.137.152	UDP	426	53 -> 59032
9	6.11226224899292	127.0.0.1	127.0.0.1	TCP	4162	0 -> 0
Frame nom. 9:	4162 bytes on wire (33296 bit					
Ethernet I Src: 00:00:00:00:00:0(	Dst: 00:00:00:00:00:00,					
Interne Protocol Version 4,	Src: 127.0.0.1,	Dst: 127.0.0.1				
Transmis sion Control Protocol,	Src Port: 0,	Dst Port: 0,	Seq: 0,	Ack: 0,	Len: 4162	
10	6.245779275894165	127.0.0.1	127.0.0.1	TCP	66	0 -> 0
11	6.39225959777832	127.0.0.1	127.0.0.1	TCP	371	0 -> 0

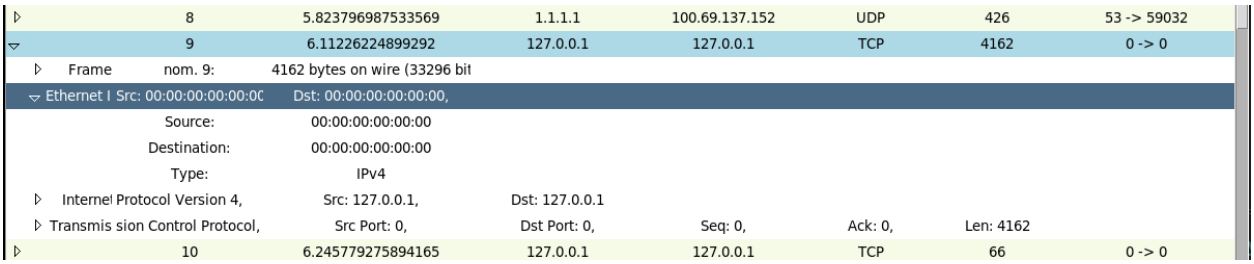
- ✓ اطلاعات نمایش داده شده برای هر فریم، در غالب هر لایه یک سطر رو اشکال می کنه و از لایه های دیگه جدا شده. (بسیار شبیه به wireshark)



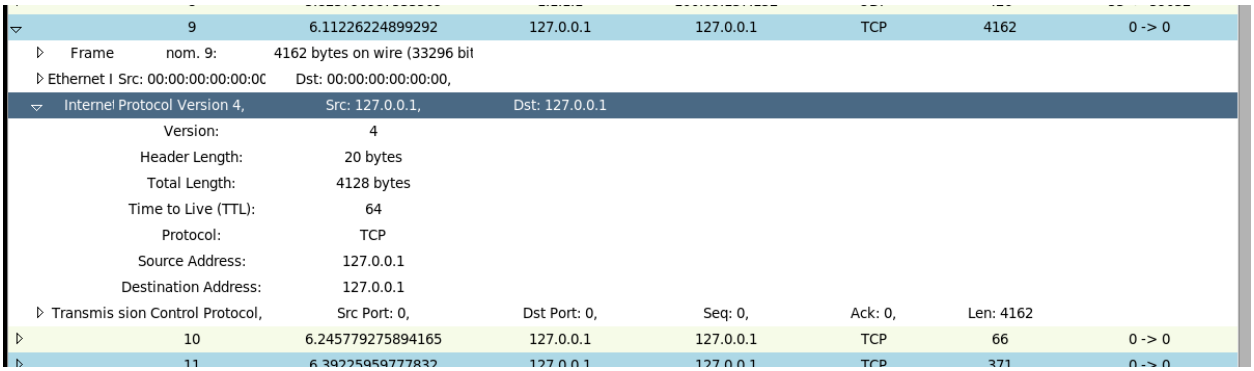
- ✓ همچنین با کلیک بر روی این سطر های جدی، دوباره اطلاعات بیشتر و مفصل تری نمایش داده می شود.
- ✓ ابتدا لایه فیزیکی قرار داره که اطلاعاتی از قبیل زمان رسیدن و طول کلی فریم (به بایت و بیت) رو نشون میده.



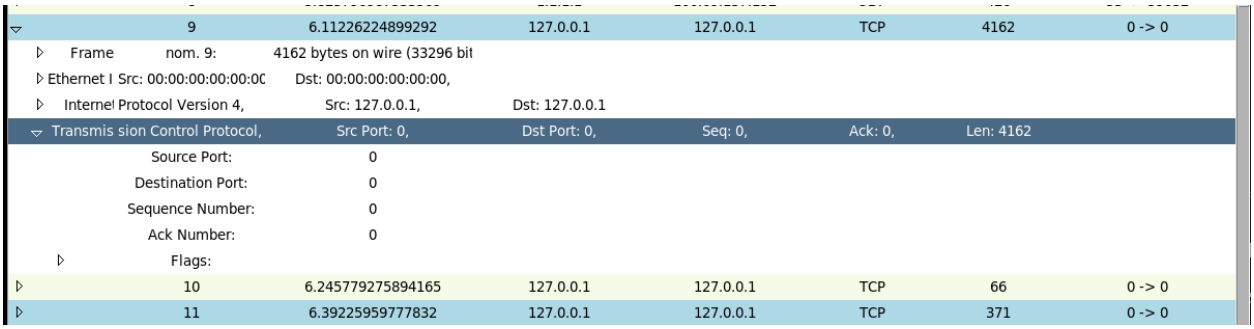
- ✓ سپس لایه (Ethernet) قرار داره که اطلاعاتی از قبیل آدرس سخت افزاری (مک) مبدا و مقصد فریم و پروتکل لایه بالایی رو نشون میده. (درحال حاضر فقط IPv4, IPv6 و ICMPv4, ICMPv6 پشتیبانی میشن)



- ✓ درلایه سه معمولا پروتکل IPv4 – Internet Protocol Version 4 قرار داره که ورژن پروتکل، طول سرایند و طول کل پکت، – TTL Time to Live، پروتکل لایه بالاتر (tcp-udp) و البته آدرس ip مبدا و مقصد پکت مشخص شده است:



- ✓ آخرین لایه اطلاعات که برنامه من از پکت ها capture شده به کاربر نشون میده، لایه چهار یا اینجا TCP که اطلاعات فراوانی از جمله شماره پورت مبدا و مقصد، شماره Sequence و Acknowledge و البته مجموعه ای از Flag ها را به کاربر نشان می دهد.



- ✓ متاسفانه با وجود زحمت فراوان و انرژی و زمان زیادی که بر روی توسعه این بخش گذاشته شده هنوز در تحلیل این بخش از بیت های Raw_packet (پکت اولیه) مشکل وجود داره و flag ها نشون داده شده غیر قابل اطمینان (unreliable) می باشد و نتایج اشتباهی نشان می دهد. لازم به ذکر است طی فرایند توسعه من چندید بار این مشکل رو حل کردم، اما با دستکاری کد برای سازگاری با بخش های دیگر مشکل در نسخه نهایی مشاهده گردید و به دلیل زمان فرصت نکردم برطرف کنم.

▷	8	5.823796987533569	1.1.1.1	100.69.137.152	UDP	426	53 -> 59032
▽	9	6.11226224899292	127.0.0.1	127.0.0.1	TCP	4162	0 -> 0
▷ Frame	nom. 9:	4162 bytes on wire (33296 bit					
▷ Ethernet I	Src: 00:00:00:00:00:0(	Dst: 00:00:00:00:00:00,					
▷ Internet	Protocol Version 4,	Src: 127.0.0.1,	Dst: 127.0.0.1				
▽ Transmis	sion Control Protocol,	Src Port: 0,	Dst Port: 0,	Seq: 0,	Ack: 0,	Len: 4162	
	Source Port:	0					
	Destination Port:	0					
	Sequence Number:	0					
	Ack Number:	0					
▽	Flags:						
	Urgent:	Set					
	Acknowledgment:	Set					
	Push:	Set					
	Reset:	Set					
	Syn:	Set					
	Fin:	Set					
▷	10	6.245779275894165	127.0.0.1	127.0.0.1	TCP	66	0 -> 0

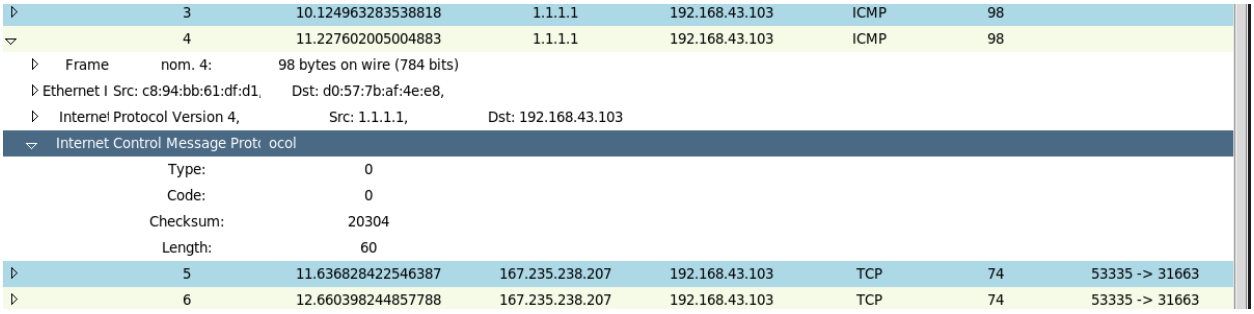
✓ Capture یک پکت ICMPv6: همانطور که مشاهده می شود به علت عدم وجود لایه IP از آدرس مک در ستون src, dst استفاده شده.

Packet Sniffer

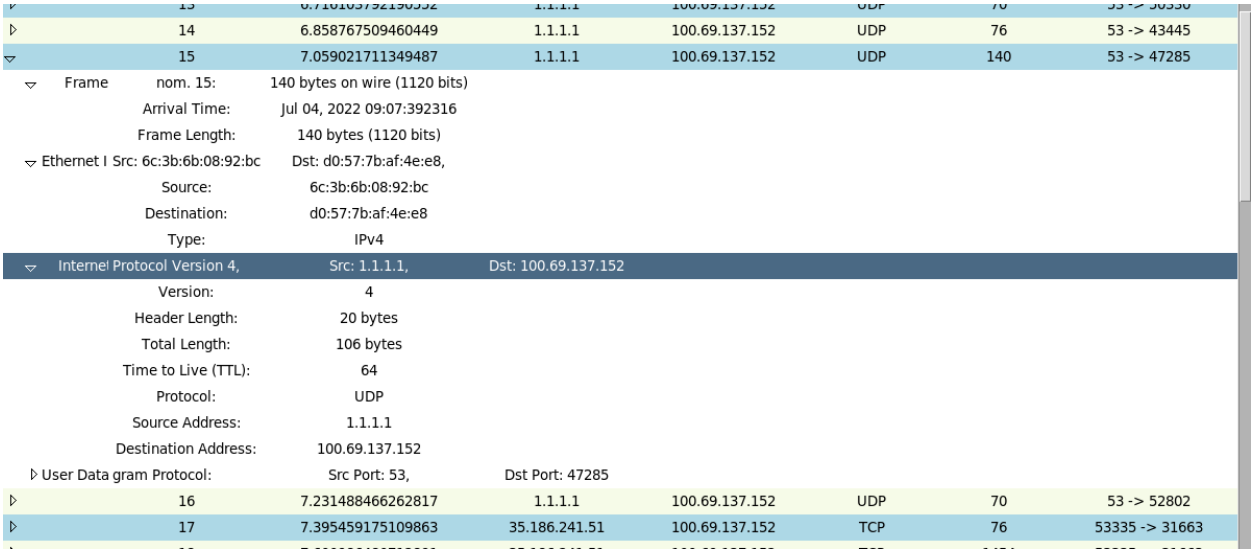
Start Stop Save

	No.	Time	Source	Destination	Protocol	Length	Info
▷	44	9.250523090362549	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
▷	45	9.540661811828613	100.69.137.152	143.204.215.32	TCP	140	27707 -> 27400
▷	46	9.793256759643555	100.69.137.152	143.204.215.32	TCP	388	27707 -> 27400
▷	47	10.086870670318604	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
▷	48	10.39084529876709	143.204.215.32	100.69.137.152	TCP	212	53335 -> 31663
▷	49	10.691806077957153	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
▷	50	11.002773761749268	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
▷	51	11.397794246673584	fe80::6af1:6374:e558:9	ff02::2	ICMPv6	62	
▷	52	11.59805154800415	143.204.215.32	100.69.137.152	TCP	96	53335 -> 31663
▷	53	11.982110023498535	100.69.137.152	143.204.215.32	TCP	78	27707 -> 27400
▷	54	12.302692890167236	143.204.215.32	100.69.137.152	TCP	699	53335 -> 31663
▷	55	12.593124628067017	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
▷	56	12.801265239715576	100.69.137.152	143.204.215.32	TCP	106	27707 -> 27400
▷	57	13.000041007995605	100.69.137.152	143.204.215.32	TCP	106	27707 -> 27400
▷	58	13.268279790878296	100.69.137.152	143.204.215.32	TCP	74	27707 -> 27400
▷	59	13.540801525115967	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
▷	60	13.814778566360474	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
▷	61	14.101274728775024	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
▷	62	14.406070470809937	143.204.215.32	100.69.137.152	TCP	76	53335 -> 31663
▷	63	14.685423374176025	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400
▷	64	14.975448608398438	100.69.137.152	143.204.215.32	TCP	601	27707 -> 27400
▷	65	15.288157224655151	143.204.215.32	100.69.137.152	TCP	68	53335 -> 31663
▷	66	15.563864707946777	143.204.215.32	100.69.137.152	TCP	259	53335 -> 31663
▷	67	15.803689241409302	100.69.137.152	143.204.215.32	TCP	66	27707 -> 27400

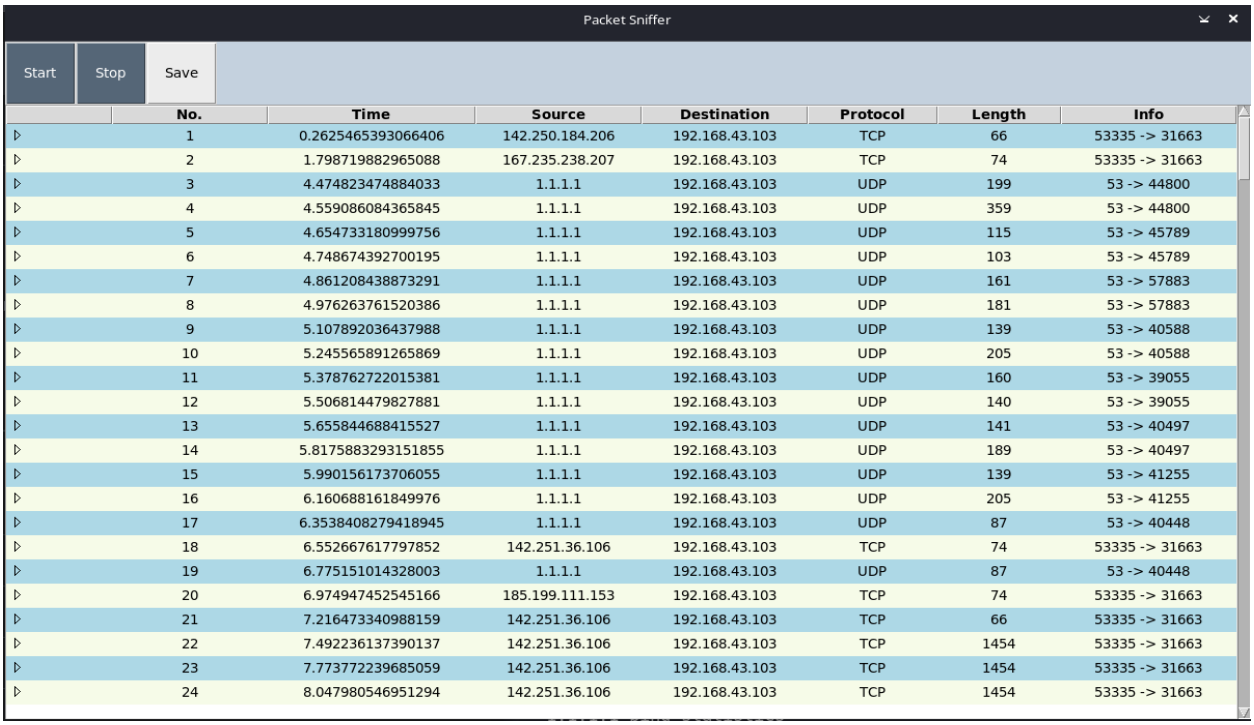
✓ مشاهده پکت ICMP کپچر شده:



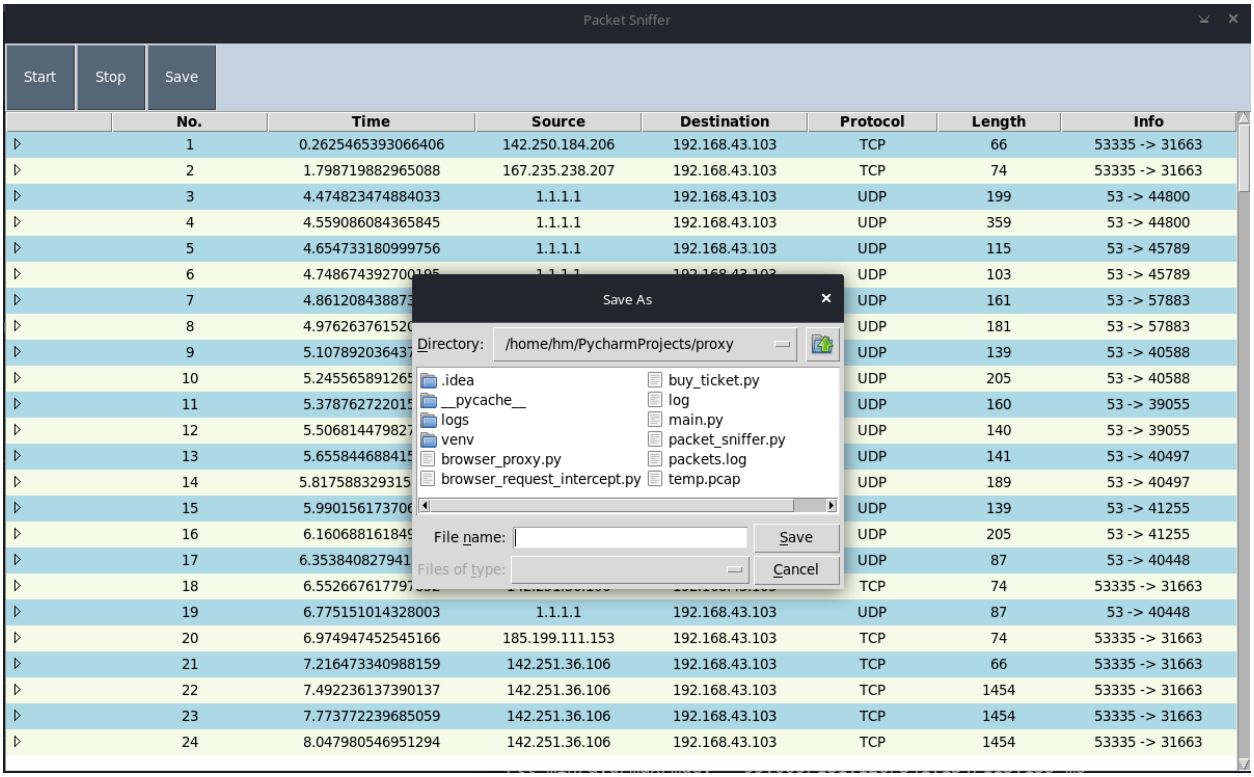
- ✓ پکت UDP:



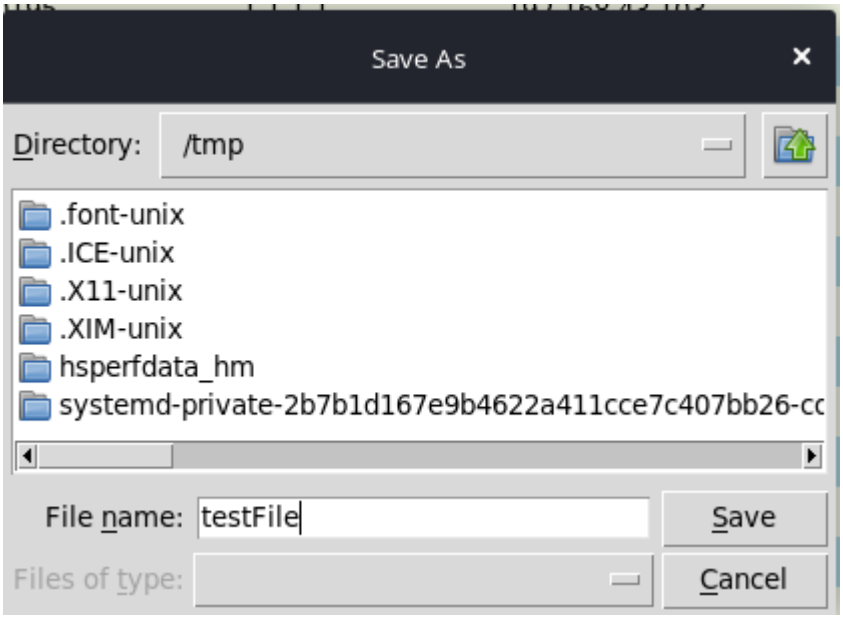
- ✓ برای تبدیل این نرم افزار ساده Packet Capture به یک برنامه واقعی کارامد و ایجاد حداکثر Integration با wireshark و دیگر برنامه های آنالیز و پردازش ترافیک شبکه،امکان ذخیره کل ترافیک و فریم های Capture شده (از جمله پروتکل هایی که توسط برنامه ساپورت و درنتیجه نمایش داده نمی شوند)، در فرمت ”pcap.“ فراهم شده است.



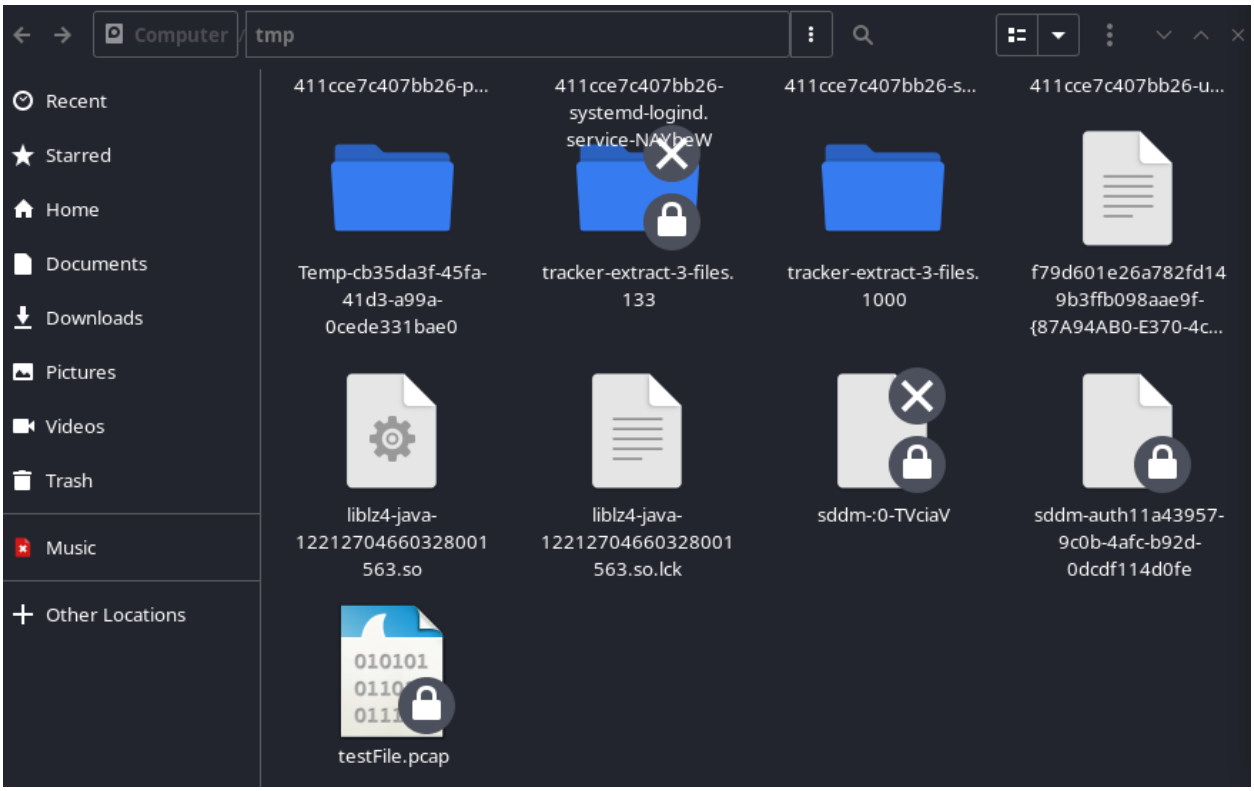
✓ با کلیک بر دکمه Save باکس انتخاب محل و اسم فایل خروجی باز می شود.



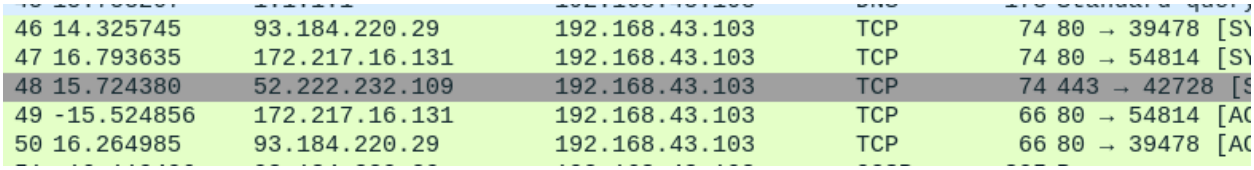
✓ انتخاب مسیر "/tmp" و نام "testFile"



✓ همانطور که مشاهده می شود، آخرین آیکون نمایانگر فایل Packet capture ما با نام "testFile.pcap" ذخیره شده است.



✓ راست-کلیک کرده و فایل را با استفاده از wireshark باز می کنیم.



46	14.325745	93.184.220.29	192.168.43.103	TCP	74 80 → 39478 [SY
47	16.793635	172.217.16.131	192.168.43.103	TCP	74 80 → 54814 [SY
48	15.724380	52.222.232.109	192.168.43.103	TCP	74 443 → 42728 [S
49	-15.524856	172.217.16.131	192.168.43.103	TCP	66 80 → 54814 [AC
50	16.264985	93.184.220.29	192.168.43.103	TCP	66 80 → 39478 [AC

➢ نکاتی در مورد کد برنامه

- برای توسعه از تکنولوژی برنامه نویسی پایتون استفاده شده
- برای رابط گرافیکی از ماژول tkinter استفاده شده
- برنامه به طور اولیه وِیژه محیط لینوکس طراحی شد، در انتها پشتیبانی از ویندوز در آن قرار داده شد ولی هیچ گونه تستی انجام نشده.
- در محیط لینوکس برنامه باید با دسترسی های سطح ادمین (sudo) اجرا شود
- برای ساختن برنامه از interface سطح پایین (low level) سیستم عامل استفاده شده. (raw_socket)

```
# Create Socket
if os.name == "nt":
    conn = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_IP)
    conn.bind((input("[+] YOUR_INTERFACE : "), 0))
    conn.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
    conn.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)
else:
    # conn = socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.ntohs(3))
    conn = socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0800))
```

- سپس با استفاده از یک حلقه while حداکثر بافر برای دریافت یک پکت را در نظرگرفته.

```
while capture:
    # Capture a packet up to specified buffer size (which is maximum size possible)
    raw_data = conn.recvfrom(65535)

    # Save captured packets into pcap file
    pcap_obj.write(raw_data[0])
```

- با استفاده از custom کلاس pcap که global header های یه فایل pcap. رو داراست روش ساخت این فایل رو پیاده سازی می کنیم. (در نهایت فایل رو ذخیره یا حذف میکنیم)
- و پکت دریافتی رو به لایه های دیگه میفرستیم
- هر لایه در غالب یه function تعریف شده که و اینجوری از یه تابع به تابع دیگه برای انجام پردازش های لایه بالا تر انجام میدیم.
- تنهای چالش دونستن فرمت ذخیره اطلاعات به بایت و بیت که برای استخراج از کتابخونه struct استفاده کرده و فرمت رو از داکیومنت های اینترنت نگاه می کنیم.