BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Pada tahap implementasi ini, penulis akan membahas tahapan-tahapan yang dilakukan dalam mengaktifkan inisialisasi Lua pada file. Setelah melakukan rancangan, selanjutnya membuat *scripting* lua, menjelaskan cara kerja fungsi dari Lua *dissector*, kemudian dievaluasi dengan kasus uji dalam bentuk kesimpulan.

Kasus uji menunjukkan proses kumpulan *dissector* berupa jumlah angka yang akan dieksekusi dari *IP* sumber (source) ke *IP* destinasi, kemudian ditampilkan ke dalam jendela TCP Sequence Number Analyzer. Selain itu, HTTP Response dan Request ditampilkan ke panel detail paket agar lebih ringkas dan mudah dibaca. Kasus uji akan diperkenalkan dengan akurasi sedemikian rupa sehingga hasilnya dapat diproses.

Sebelum menjalankan *scripting* lua, di Gambar 4.1 menjelaskan konfigurasi pertama berisi *scripting* untuk menjalankan perintah mengaktifkan dukungan lua, dan perintah untuk akses *super user*.

```
    15. disable_lua = false
    16.
    17. if not disable_lua then
    18. return
    19. end
```

Gambar 4.1 Konfigurasi pada init.lua

Jika file *scripting* berada di luar plug in (selain direktori "Wireshark/plugins/..") dapat dimasukkan direktori ke dalam *scripting* pada Gambar 4.2, baris 700 digunakan untuk memasukkan direktori berada di posisi baris terakhir pada *scripting*.

```
20. if not running_superuser or run_user_scripts_when_superuser then
21. dofile(DATA_DIR.."scripting.lua")
22. end
```

Gambar 4.2 Konfigurasi file direktori pada init.lua

Langkah berikutnya adalah Membuat protokol baru, yang berisi beberapa variabel utama dan fungsi – fungsi yang akan dijalankan pada Gambar 4.3.

```
23. local http_response_proto = Proto("http_resp", "Upstream response HTTP request")
24. local http_shortcut_detail = Proto("http_shortcut", "Shortcut HTTP data")
25.
26. local stream_map = {} -- table per stream : { req_cnt, resp_cnt, reqs }
27. local resp_map = {}
```

Gambar 4.3 Membuat protokol baru dalam scripting

Selanjutnya, membuat *dissector function* untuk TCP Sequence Analyzer (TSA) yang akan digunakan untuk membuat dan memanggil fungsi dijelaskan pada Gambar 4.4.

28.	<pre>function tsa_menu_func()</pre>
29.	tsa analyze function
30.	<pre>function tsa_do(ip1, ip2)</pre>
31.	local results = {
32.	Total
33.	["counter"] = 0,
34.	["dupack"] = 0,
35.	["retrans"] = 0,
36.	["fastretrans"] = 0,
37.	["zerowindow"] = 0,
38.	["windowfull"] = 0,
39.	
40.	dari ip1 ke ip2
41.	["counter1to2"] = 0,
42.	["dupack1to2"] = 0,
43.	["retrans1to2"] = 0,
44.	["fastretrans1to2"] = 0,
45.	["zerowindow1to2"] = 0,
46.	["windowfull1to2"] = 0,
47.	
48.	dari ip2 ke ip1
49.	["counter2to1"] = 0,
50.	["dupack2to1"] = 0,
51.	["retrans2to1"] = 0,
52.	["fastretrans2to1"] = 0,
53.	["zerowindow2to1"] = 0,
54.	["windowfull2to1"] = 0,
55.	}
56.	local result_win = TextWindow.new("Tcp Sequence Numbers Analyze")
57.	local http_port = 80

Gambar 4.4 Membuat fungsi dissector

Setelah membuat dan memanggil fungsi, akan membuat tap. Tap adalah hasil langkah dari paket yang akan memproses hasil *capture* pada Gambar 4.5.

58.	pemanggilan tap.draw
59.	<pre>function refresh_result()</pre>
60.	result_win:clear()
61.	result_win:set("Total\n")
62.	result_win:append("\tPackets counter: " results["counter"] "\n")
63.	<pre>result_win:append("\tDuplicate ACK: " results["dupack"] "\n")</pre>
64.	result_win:append("\tRetransmission: " results["retrans"] "\n")
65.	result_win:append("\tFast Retransmission: " results["fastretrans"] "\n")
66.	result_win:append("\tZero Window: " results["zerowindow"] "\n")
67.	result_win:append("\tWindow Full: " results["windowfull"] "\n")
68.	<pre>result_win:append("\n")</pre>
69.	result_win:append(ip1 " -> " ip2 "\n")
70.	result_win:append("\tPackets counter: " results["counter1to2"] "\n")
71.	<pre>result_win:append("\tDuplicate ACK: " results["dupack1to2"] "\n")</pre>
72.	result_win:append("\tRetransmission: " results["retrans1to2"] "\n")
73.	result_win:append("\tFast Retransmission: " results["fastretrans1to2"] "\n")
74.	result_win:append("\tZero Window: " results["zerowindow1to2"] "\n")
75.	result_win:append("\tWindow Full: " results["windowfull1to2"] "\n")
76.	result_win:append("\n")
77.	result_win:append(ip2 " -> " ip1 "\n")
78.	result_win:append("\tPackets counter: " results["counter2to1"] "\n")
79.	<pre>result_win:append("\tDuplicate ACK: " results["dupack2to1"] "\n")</pre>
80.	result_win:append("\tRetransmission: " results["retrans2to1"] "\n")
81.	result_win:append("\tFast Retransmission: " results["fastretrans2to1"] "\n")
82.	result_win:append("\tZero Window: " results["zerowindow2to1"] "\n")
83.	result_win:append("\tWindow Full: " results["windowfull2to1"] "\n")
84.	end

Gambar 4.5 Proses output pada scripting yang akan ditampilkan

Gambar 4.6 menjelaskan jumlah TSA paket akan ditampilkan setelah proses tap selesai, setiap baris mempunyai script yang sama, tetapi perbedaan hanya pemanggilan fungsi di setiap barisan, berikut adalah salah satu dari 18 fungsi *scripting* yang akan ditampilkan di window.

```
85. -- packets counter, total
           local counter_tap = Listener.new("frame", "ip.addr == " .. ip1 .. " && ip.addr == " .. ip2
86.
    .. " && tcp.port == " .. http_port)
           function counter_tap.reset()
87.
              results["counter"] = 0
88.
89.
           end
90.
           function counter_tap.packet(pinfo, tvb, ip)
              results["counter"] = results["counter"] + 1
91.
92.
           end
93.
           function counter_tap.draw()
```

Gambar 4.6 Scripting untuk menampilkan jumlah seluruh paket

Untuk menghapus hasil jumlah paket tap pada baris 296 sampai baris 316 dijelaskan pada Gambar 4.7 menggunakan remove_alltap, fungsi ini untuk menutup aplikasi *plugin*

96.	<pre>function remove_alltap()</pre>
97.	<pre>counter_tap:remove()</pre>
98.	<pre>dupack_tap:remove()</pre>
99.	retrans_tap:remove()
100.	<pre>fastretrans_tap:remove()</pre>
101.	<pre>zerowindow_tap:remove()</pre>
102.	<pre>windowfull_tap:remove()</pre>
103.	
104.	<pre>counter1to2_tap:remove()</pre>
105.	<pre>dupack1to2_tap:remove()</pre>
106.	<pre>retrans1to2_tap:remove()</pre>
107.	<pre>fastretrans1to2_tap:remove()</pre>
108.	<pre>zerowindow1to2_tap:remove()</pre>
109.	<pre>windowfull1to2_tap:remove()</pre>
110.	
111.	<pre>counter2to1_tap:remove()</pre>
112.	<pre>dupack2to1_tap:remove()</pre>
113.	<pre>retrans2to1_tap:remove()</pre>
114.	<pre>fastretrans2to1_tap:remove()</pre>
115.	<pre>zerowindow2to1_tap:remove()</pre>
116.	<pre>windowfull2to1_tap:remove()</pre>
117.	End
118.	
119.	<pre>result_win:set_atclose(remove_alltap)</pre>
120.	
121.	retap_packets()
122.	end

Gambar 4.7 Menghilangkan dan menutup hasil TSA

Prompt alamat IP diperlukan untuk mengisi form berupa alamat IP. Pada Gambar 4.8, *scripting* ini juga untuk mengakhiri program TSA pada baris 327. Di baris 332, tree item adalah fungsi yang akan ditampilkan di detail paket Wireshark, *scripting* yang akan dibuat adalah respons dan jalan pintas untuk protokol HTTP.

125.					
126.					
127.					
128.http_response_proto.fields.re_req_method	<pre>= ProtoField.string("http_resp.request.method", "Request</pre>				
Method")					
129.http_response_proto.fields.re_req_uri	<pre>= ProtoField.string("http_resp.request.uri", "Request</pre>				
URI")					
130.http_response_proto.fields.re_req_ver	<pre>= ProtoField.string("http_resp.request.version",</pre>				
"Request Version")					
131.http_response_proto.fields.re_host	<pre>= ProtoField.string("http_resp.host", "Host")</pre>				
Gambar 4.8 Scripting untuk memasukkan Tree item					

Kemudian di Gambar 4.9, baris 338 – 349 memberikan panggilan permintaan dan respons dalam protocol field untuk memasukkan proses fungsi kemudian.

```
= ProtoField.string("http.request.URL", "Request URL")
132.http_shortcut_detail.fields.URL
133.
134.local f_req_meth = Field.new("http.request.method")
135.local f_req_uri = Field.new("http.request.uri")
136.local f_req_ver
                      = Field.new("http.request.version")
137.local f_req_host = Field.new("http.host")
138.local f_resp_code = Field.new("http.response.code")
139.local f_tcp_stream = Field.new("tcp.stream")
140.local f_tcp_dstport = Field.new("tcp.dstport")
141.local f_tcp_srcport = Field.new("tcp.srcport")
142.local f_ip_dsthost = Field.new("ip.dst_host")
143.local f_ip_srchost = Field.new("ip.src_host")
144.
145.function http_response_proto.init()
146.
       stream_map = {}
147.
        resp_map = {}
148.end
```

Gambar 4.9 Memasukkan untuk beberapa permintaan dan respons

Untuk fungsi TCP *port* pada Gambar 4.10 terbagi menjadi dua, yaitu baris 356 – 361 adalah optional port dan baris 363 – 368 adalah scheme by port. Masing – masing TCP *port* adalah port 443 dan port 80 di baris 357

```
149.local function optional_port(tcp_port)
150. if tcp_port ~= 443 and tcp_port ~= 80 then
151. return ":" .. tcp_port
152. end
153. return ""
154. end
```

```
155.
156. local function scheme_by_port(tcp_port)
157. if tcp_port == 443 then
158. return "https://"
159. else
160. return "http://"
161. end
162. end
```

Gambar 4.10 Fungsi optional port dan scheme by port untuk TCP port.

Membuat fungsi protokol dissector dimulai dari baris ke-371 pada Gambar 4.11, yang berfungsi untuk memproses respons dan permintaan dalam satu fungsi. Permintaan dan respons memiliki fungsi yang berbeda, permintaan berfungsi ketika paket stream datang dan melakukan dissector paket. Sedangkan respons berfungsi jika ada paket stream yang datang mendapat respons, lalu mengarahkan secara arus balik dan mengarah ke permintaan yang sesuai.

```
163.function http_response_proto.dissector(tvbuffer, pinfo, treeitem)
164.
           if not f tcp stream() then return end
165.
           local stream n
166.
           local URL
167.
           stream_n = f_tcp_stream().value
           URL = nil
168.
169.
170.
           if f_req_meth() then
171.
               tcp_port = f_tcp_dstport().value
172.
               local host
               if f_req_host() then
173.
                   host = f req host().value
174.
175.
               else
176.
                   host = f_ip_dsthost().value
177.
               end
178.
               URL = scheme_by_port (tcp_port) .. host .. optional_port(tcp_port) ..
    f_req_uri().value
179.
               if not pinfo.visited then
180.
                   if stream_map[stream_n] == nil then
181.
                       stream_map[stream_n] = {0,0,{}}
182.
                   end
```

Gambar 4.11 Memasukkan fungsi permintaan

Sedangkan di baris 392 sampai 394, perhitungan respons berfungsi jika ada paket stream yang datang mendapat respons, lalu baris 399 sampai dengan 412 mengarahkan arus balik, dan kembali ke permintaan yang sesuai Gambar 4.12.

183.	<pre>request_n = stream_map[stream_n][1] + 1</pre>
184.	<pre>stream_map[stream_n][1] = request_n</pre>
185.	stream_map[stream_n][3][request_n] = {f_req_meth().value, f_req_uri().value,
f	_req_ver().value, f_req_host().value}
186.	end
187.	End
188.	
189.	
190.	if f_resp_code() then
191.	local response_n
192.	if not pinfo.visited then
193.	if not stream_map[stream_n] then
194.	warn("HTTP response not in stream (" pinfo.number ")")
195.	response_n = 0
196.	else
197.	<pre>response_n = stream_map[stream_n][2] + 1</pre>
198.	<pre>stream_map[stream_n][2] = response_n</pre>
199.	<pre>resp_map[pinfo.number] = response_n</pre>
200.	end
201.	else
202.	<pre>response_n = resp_map[pinfo.number]</pre>
203.	End

Gambar 4.12 Memasukkan fungsi arus balik respons ke permintaan.

Untuk fungsi TCP port dilihat pada Gambar 4.13 menggunakan fungsi percabangan umtuk menentukan hasil data lost di baris 431.

204.	<pre>tcp_port = f_tcp_srcport().value</pre>
205.	if response_n > 0 then
206.	<pre>local subtree = treeitem:add(http_response_proto, nil)</pre>
207.	local data = stream_map[stream_n][3][response_n]
208.	if data then
209.	<pre>subtree:add(http_response_proto.fields.re_req_method, data[1]):set_generated()</pre>
210.	<pre>subtree:add(http_response_proto.fields.re_req_uri, data[2]):set_generated()</pre>
211.	<pre>subtree:add(http_response_proto.fields.re_req_ver, data[3]):set_generated()</pre>
212.	if (data[4]) then
213.	<pre>subtree:add(http_response_proto.fields.re_host, data[4]):set_generated()</pre>
214.	end
215.	host = data[4]
216.	if not host then
217.	<pre>host = f_ip_srchost().value</pre>
218.	end
219.	<pre>URL = scheme_by_port(tcp_port) host optional_port(tcp_port) data[2]</pre>

220.	Else
221.	warn("HTTP request data lost (" stream_n "," response_n ")")
222.	end
223.	end
224.	end

Gambar 4.13 Fungsi respons untuk packet stream.

Post-dissector digunakan untuk menampilkan eksekusi pada aplikasi perangkat wireshark, posisi *post-dissector* sebagai akhir dari program *scripting* Lua dilihat pada Gambar 4.14.

225.-- Register post-dissector
226.register_postdissector(http_response_proto)
227.register_menu("Tcp Sequence Numbers Analyze", tsa_menu_func, MENU_TOOLS_UNSORTED)
Gambar 4.14 Fungsi register post-dissector.

4.2 Tahapan Menjalankan Dissector

4.2.1 Pengambilan Data

Pengambilan dilaksanakan pada tanggal 27 Agustus 2018, di lokasi Jalan Affandi, Soropadan, Condongcatur, Kecamatan Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta. Tabel 4.1 menjelaskan detail laporan pengambilan paket data yang dilakukan.

Nama Tempat	Warunk Upnormal
Lokasi	Jalan Affandi, Soropadan, Condongcatur, Kecamatan
	Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta
Tanggal Pengambilan Data	27 Agustus 2018
Durasi Pengambilan Data	20:10 – 23:16 (3 jam)
Ukuran (Megabyte)	783 MB
Format	Wireshark Capture File (.pcap)
Nama File	Capture.pcap

Tabel 4.1 Detail pengambilan paket data

4.2.2 Pengujian Pertama

Pengujian *dissector* menunjukkan proses pembuatan *dissector* untuk protokol khusus yang beroperasi di protokol TCP dan HTTP, sebelum memiliki *dissector* protokol khusus,

Wireshark tidak mampu menafsirkan dan mengumpulkan beberapa paket protokol. Hasil *dissector* paket protokol khusus di Wireshark tanpa *add on* yang tepat ditunjukkan pada Gambar 4.15, dapat dilihat seluruh sub tree hanya bawaan aplikasi Wireshark saja.

	Wireshark • Packet 522940 • capture.pcap _ 0 😣
 Frame 522940: 863 bytes on wire (6904 bits), 863 bytes of • Ethernet II, Src: Routerbo_13:c5:1e (oc:2d:e0:13:c5:1e), Internet Protocol Version 4, Src: 172.217.26.68, Dst: 1e • Transmission Control Protocol, Src Port: 80, Dst Port: 3 [36 Reassembled TCP Segments (34829 bytes): #517188(1418 • Hypertext Transfer Protocol 	captured (6904 bits) , Dst: Tp-LinkT_0d:50:1d (18:a6:f7:0d:50:1d) 0.1.41.214 36100, Seq: 34033, Ack: 403, Len: 797 8), #517252(1418), #517334(1418), #517312(1418), #517381(1418), #517440(1418), #517465(1418
> JPEG File Interchange Format	
4	•
0000 18 a6 f7 0d 50 1d cc 2d e0 13 c5 1e 08 00 45 00 00 10 13 c5 1e 08 00 45 00 00 10 33 51 c1 97 00 00 39 06 c2 1b ac d9 1a 44 0a 01 32 cb d8 18 32 cb 44 30 55 ad d4 dd ad b2 cb d4 39 cb d8 18	•
0030 00 f0 5a c7 00 00 1 01 08 0a 76 25 37 86 01 30 0040 c1 dd af ee ba ca e5 1d 37 81 63 15 30 74 73 bb	. Z
0050 8f 99 f5 a4 92 01 72 66 37 1a b9 1a 49 20 b1 49 0060 c0 5e 8a cc 07 49 24 d0 99 66 97 f7 a5 e4 3a 3a 0070 3a 82 a7 cc da 2 83 88 6c 24 5a ac fa 55 7 a 49	······································
0080 24 c1 13 b4 b3 f4 f8 76 b6 87 77 35 9a f6 62 9 9 0090 18 87 64 da 2 f6 bd d2 49 22 90 25 4b 24 59	· · · · · · · · · · · · · · · · · · ·
0000 b4 a5 94 a3 2b 10 65 77 76 76 e6 d4 d6 bf ad 58 00b0 85 e9 02 8e 51 d2 10 cb 9d 98 c4 82 e5 ae ec f6 00c0 6b b5 ad 66 e7 49 24 8a 29 cb 59 25 3d 61 15 04	····+ ew vv·····X ···Q···· k··f·IS·)·Y%=a··
00d0 93 47 16 e1 12 26 77 6d 4d 7b bb 33 73 f7 2a f5 00e0 35 12 55 6d 54 4f 24 85 e5 24 92 74 45 b2 28 d8 00f0 4f 32 04 e4 3e 88 ff 00 75 a9 4d 8c d6 d3 80 c7	-G&vm M{-3S-*- 5-UmTOS\$-tE-(-
Frame (863 bytes) Reassembled TCP (34829 bytes)	۳
Help	¥ <u>c</u> lose

Gambar 4.15 Standar sub Tree Wireshark jika tidak menggunakan scripting.

Setelah implementasi pembuatan *scripting* selesai, kemudian pada Gambar 4.16 menjelaskan *scripting* akan dijalankan dengan dua perintah, dua perintah tersebut akan menyesuaikan tempat pada tampilan Wireshark yang akan dijalankan. Sebelum menjalankan *scripting*, pengguna terlebih dahulu memeriksa folder plugin di dalam direktori Wireshark, atau membuka menu bar About Wireshark (Help – About Wireshark), kemudian klik tab folders.



Gambar 4.16 Direktori Global Plugins untuk memasukkan dan modifikasi scripting.

Pengguna dapat memeriksa *scripting* yang telah dipasang pada tab plugins untuk memastikan munculnya *scripting*, jika tidak muncul gunakan kombinasi keyboard (ctrl+shift+L). Pada gambar 4.1, informasi yang digunakan untuk definisi protokol disajikan, baris pertama menunjukkan jenis informasi, kemudian baris kedua menunjukkan sumber URI (Uniform Resource Identifier), baris ketiga menunjukkan versi HTTP baris keempat menunjukkan URL (Uniform Resource Locator), serta baris kelima adalah gabungan dari URL dan URI. Menjalankan perintah program yang sebelumnya mengetahui salah satu paket IP sumber ke IP destinasi dan detail HTTP seperti pada gambar 4.1.



Gambar 4.17 Detail upstream respons pada pengujian pertama.

Setelah melakukan pencarian paket seperti di atas, TSA akan menghasilkan jumlah tap seperti pada Gambar 4.18, gambar tersebut hanya mendapatkan 3 detail, yaitu packet counter, duplicate ACK, dan retransmission. Hasil akhir pada TSA berbeda ketika IP source ke IP destinasi telah selesai dihitung, kemudian menghitung kembali secara arus balik dari IP destinasi ke IP source.



Gambar 4.18 Hasil dissector Wireshark pengujian pertama.

4.2.3 Pengujian Kedua

Pada Gambar 4.18, pengujian kedua menunjukkan respons HTTP yang berbeda, tetapi *IP* destinasi sama seperti pengujian pertama.

				caj	pture.pcap			_ a 📀		
Eile Ed	lit <u>V</u> iew <u>G</u> o	Capture Analyze Statistics	s Telephony <u>W</u> ireless <u>1</u>	ools <u>H</u> elp						
	2 💿 📒	🗎 🖹 🖉 🔍 🔇 🕽	> > I= → 📃 🔳		11					
Apply	a display filter	<ctrl-></ctrl->						Expression +		
No.	Time	Source	Destination	Protocol	Length JPE	G File Interchange Format	 Upstream response HTTP request 	Info		
731	7 939.32660	1 114.4.168.130	10.1.41.229	HTTP	602 🗸			HTTP/1.1 2		
725	7 930.34975	0 172.217.26.65	10.1.41.229	HTTP	168 /			HTTP/1.1 2		
650	4 858.38033	7 88.99.65.49	10.1.41.229	HTTP	1367 🗸			HTTP/1.1 2		
642	9 852.27085	8 50.87.150.19	10.1.41.229	HTTP	1080 /			HTTP/1.1 2		
638	7 847.12312	5 184.105.132.210	10.1.41.229	HTTP	815 🗸			HTTP/1.1 2		
630	3 839.80490	6 64.62.214.195	10.1.41.229	HTTP	1429 /		1	HTTP/1.1 2		
622	2 827.61118	7 46.101.198.69	10.1.41.229	HTTP	763 🗸		1	HTTP/1.1 2		
615	5 822.53592	1 104.27.132.140	10.1.41.229	HTTP	69 🖌			HTTP/1.1 2		
538	1 738.57841	5 104.27.180.141	10.1.41.229	HTTP	1272 /			HTTP/1.1 2		
526	8 724.67629	7 172.217.26.68	10.1.41.214	HTTP	672 🗸		1	HTTP/1.1 2		
526	2 724.25182	4 172.217.26.68	10.1.41.214	HTTP	1447 /		1	HTTP/1.1 2		
4										
Frame Ether Inter	Frame 630397: 1429 bytes on wire (11432 bits), 1429 bytes captured (11432 bits) > Ethernet II, Src: Routerbo_33:c5:1e (cc:2d:e0:13:c5:1e), Dst: Tp-LinkT_0d:50:1d (18:a6:f7:0d:50:1d) > Internet Protocol Version 4, Src: 64.c2:14.195, Dst: 10.1.41.292									
+ Trans	smission Cor	trol Protocol, Src Por	t: 80, Dst Port: 507	49, Seq: 669	733, Ack:	432, Len: 1363				
· [933	Reassembled	TCP Segments (671095	bytes): #608711(1428), #613151(1	428), #614	920(1428), #614937(1	428), #614921(1428), #615428(1428), #615468(1428)		
нуре	text Transf	er Protocol								
> JPEG	File Interd	nange Format								
[R) [R) [R) [R) [H)	equest Metho equest URI: equest Vers: pst: www.the	<pre>child request od: GET] /parents_day/wallpaper ion: HTTP/1.1] ebolidayspot.com]</pre>	s/images_hd/parents-	day-wish.jpg	1]					
- Short	tcut HTTP da	ta								
[R	equest URL:	http://www.theholidays	pot.com/parents_day/	wallpapers/i	mages_hd/p	arents-day-wish.jpg]				
4								•		
0 2	Upstream resp	onse HTTP request (http_resp)			Packets	: 1006012 · Displayed: 1006012 (100.0	%) Profile: Default		

Gambar 4.19 Detail upstream respons pada pengujian kedua.

Hasil dari Gambar 4.20, tiga detail tap yang sama telah ditampilkan. Hasil penjumlahan berbeda dengan pengujian pertama, pengujian kedua mendapatkan tap lebih banyak daripada pengujian pertama.



Gambar 4.20 Hasil dissector Wireshark pengujian kedua.

4.2.4 Hasil Perbandingan

Jumlah Hasil TSA dengan grafik statistik endpoint dari perhitungan telah ditampilkan, salah satu total paket pengujian mempunyai kemiripan yang sama. Pada Gambar 4.21 dapat dilihat persamaan dan perbedaan antara TSA dengan statistik endpoint, TSA lebih lengkap jika ditambahkan dengan 5 detail, sedangkan statistik endpoint hanya menghitung jumlah paket dengan satu arah dan harus mencari IP destinasi jika ingin menghitung jumlah paket untuk menyesuaikan, total seluruh paket dari TSA dan endpoint sebanyak 1298 paket. Jumlah paket data dalam 2 protokol yang berbeda sebanyak 1.298 paket, transfer paket sebanyak 1.288 paket, dan terima paket sebanyak 10 paket.

Wireshark · Tcp Sequence Numbers Analyze 🛛 💡)					Wir
	Ethernet - 135	i IPv4 · 3545	0 IPv6-87	TCP - 54436	UDP - 632	9
Total	Address	* Packets	Bytes Tx	Packets Tx By	tes Dy B	Packets
Packets counter: 1298	64.31.24.108	2	132	0	0	2
Duplicate ACK: 8	64.31.32.114	2	132	õ	õ	2
Retransmission: 264	64.31.224.44	2	132	ō	0	2
East Petransmission: A	64.35.175.101	1	66	0	0	1
Tase Minday A	64.35.183.146	1	66	0	0	1
Zero window: 0	64.38.92.7	2	132	0	0	2
Window Full: 0	64.42.194.92	2	132	0	0	2
	64.43.8.120	2	132	0	0	2
$64.62.214.195 \rightarrow 10.1.41.229$	64.46.102.163	2	132	0	0	2
Packets counter: 1288	64.48.111.95	2	132	0	0	2
Publicate ARK, 7	64.52.180.132	2	132	0	0	2
Dupilcate Ack. 7	64 60 42 5	2	132	0	0	
Retransmission: 261	64.62.214.195	1 298	1 865 k	1 200	1.864 k	10
Fast Retransmission: 0	64 64 23 242	1,270	66	0	0	1
Zero Window: 0	64.65.70.136	2	132	ő	õ	2
Window Full: 0	64.67.71.52	2	132	0	0	2
	64.73.148.29	2	132	0	0	2
	64.74.13.132	2	132	0	0	2
10.1.41.229 -> 64.62.214.195	64.75.136.156	2	132	0	0	2
Packets counter: 10	64.75.218.43	2	132	0	0	2
Duplicate ACK: 1	64.77.3.155	1	66	0	0	1
Retransmission: 3	64.79.67.219	2	132	0	0	2
Fact Detroperiosion, 0	64.79.131.108	2	132	0	0	2
	64.83.127.112	2	132	0	0	2
Zero Window: 0	64.89.211.238	2	132	0	0	
Window Full: 0	64 93 159 70	2	132	0	ő	
	64 93 188 123	2	132	0	0	5
	64.94.230.101	2	132	ő	ő	5
	64.06.50.01	2	122		<u> </u>	
Highlight:		tion 🗆 🗆	imit to displ	av filter		
			anne eo dispe	ay much		
¥ Close	Help					

Gambar 4.21 Perbandingan TSA dengan grafik statistik endpoint.

Hasil berikutnya ialah perbandingan hasil output HTTP biasa dengan HTTP Request Response. Pada Gambar 4.22, menjelaskan antara kedua fungsi dengan hasil yang berbeda tergantung pada hasil paket yang telah ditampilkan. Perbandingan antara protokol HTTP Request tidak banyak memerlukan tempat atau lebih singkat dibandingkan dengan HTTP biasa yang memiliki banyak detail paket, serta posisi HTTP Request Response berada di bawah karena disebut sebagai protokol baru, terdiri dari request method, URI, Versi dan Host.



Gambar 4.22 Perbandingan protokol HTTP Script Lua dengan protokol HTTP default.

Bedasarkan gambar 4.23, Hasil output paket di *terminal* yang diproses melalui protokol HTTP Request Response, kadang-kadang mempunyai masalah seperti *error*, "HTTP Data not in stream" dan "Request data loss", biasanya hal ini disebabkan oleh paket yang terenkripsi, atau tidak *compatible*.



Gambar 4.23 Hasil Output HTTP Scripting Lua jika tidak menemukan protokol HTTP.