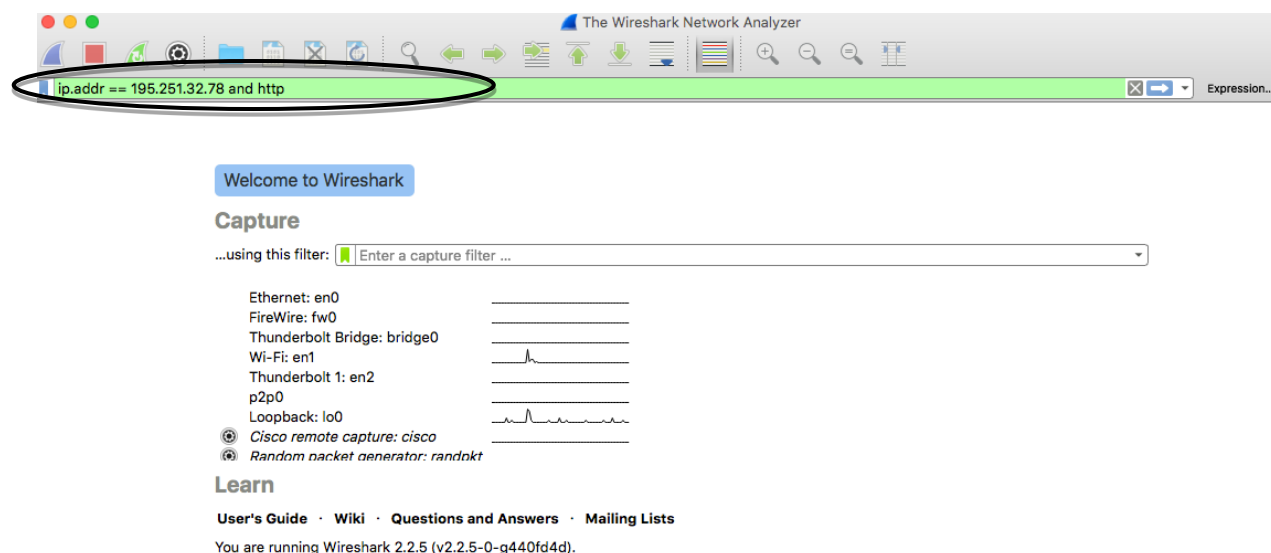


Εργαστηριακή Άσκηση

Ερώτηση 1

Για την εκτέλεση της Ερώτησης 1, θα πρέπει να χρησιμοποιήσετε το εργαλείο *Wireshark*, το οποίο είναι δωρεάν διαθέσιμο στο <http://www.wireshark.org/download.html> για λειτουργικά συστήματα Windows/Linux/Mac. Το Wireshark είναι ένα εργαλείο το οποίο «συλλαμβάνει» και εμφανίζει την κίνηση του δικτύου στον υπολογιστή σας. Αφού ξεκινήσετε το λογισμικό Wireshark θα εμφανιστεί ένα παράθυρο παρόμοιο με αυτό της παρακάτω εικόνας. Για τη χρήση του, θα πρέπει να κάνετε διπλό κλικ στην κατάλληλη διεπαφή που χρησιμοποιείτε για την πρόσβασή σας στο Διαδίκτυο (μέσα από τη διαθέσιμη λίστα του εργαλείου), όπως εμφανίζεται στην παρακάτω εικόνα (π.χ. την ενσύρματη σύνδεση Ethernet ή τη σύνδεση μέσω Wi-Fi).



Χρησιμοποιήστε το πεδίο του φίλτρου «απεικόνισης» (display filter) για να παρουσιάζονται μόνο τα πακέτα που σας ενδιαφέρουν¹.

Σε αυτήν την ερώτηση, θα καταγραφούν τα μηνύματα HTTP που παράγονται κατά την επίσκεψη μιας ιστοσελίδας με χρήση ενός πλοηγού ή φυλλομετρητή Ιστού². Προτού αρχίσετε την καταγραφή φροντίστε να αδειάσετε την προσωρινή μνήμη (cache) του πλοηγού από την επιλογή *Ιστορικό > Εκκαθάριση πρόσφατου ιστορικού*.

Επειδή τα μηνύματα HTTP μεταφέρονται σε περισσότερα από ένα πακέτα, αφού ξεκινήσετε το Wireshark, μεταβείτε στο μενού *Edit>Preferences*, κάντε κλικ στο σύμβολο δίπλα στο *Protocols* στην αριστερή πλευρά του παραθύρου, κατόπιν εντοπίστε και κάντε κλικ στο πρωτόκολλο HTTP και βεβαιωθείτε ότι όλες οι επιλογές περί ανασύνθεσης και αποσυμπίεσης είναι επιλεγμένες. Στη συνέχεια, στο πρωτόκολλο TCP βεβαιωθείτε ότι το *Allow subdissector to reassemble TCP streams* είναι επιλεγμένο και φροντίστε το *Validate the TCP*

¹ Το φίλτρο που εφαρμόζεται στην εικόνα είναι απλώς ένα παράδειγμα.

² Προτείνεται η χρήση του Firefox, αφού πρώτα έχετε καθαρίσει την προσωρινή του μνήμη (cache).

*checksum if possible*³ να **μην** είναι επιλεγμένο. Τέλος, πιέστε OK για να κλείσει το παράθυρο και εφαρμοστούν οι αλλαγές σας. Με τις επιλογές αυτές, μηνύματα που μεταφέρονται σε περισσότερα από ένα πακέτα θα αποκωδικοποιηθούν από το Wireshark ως πλήρη μηνύματα HTTP και όχι αποσπασματικά.

Όπως ήδη γνωρίζετε, μια σελίδα HTML είναι μια συλλογή αντικειμένων (π.χ. ενός αρχείου HTML, διαφόρων εικόνων JPEG, αρχείων ήχου, JavaScripts, Java applets κ.λπ.) και μπορεί να περιέχει παραπομπές προς άλλες ιστοσελίδες. Στην ερώτηση αυτή, θα μελετήσετε την κίνηση που παρατηρείται στην περίπτωση όπου ο πλοηγός ιστού κατεβάζει μια σελίδα που περιέχει ενσωματωμένα αντικείμενα (embedded objects).

Ξεκινήστε μια καταγραφή (επιλογή: *Capture>Start* ή *Ctrl+E*), επισκεφτείτε την ιστοσελίδα <http://edu-dy.cn.ntua.gr/links.html>⁴ και σταματήστε την καταγραφή (επιλογή: *Capture>Stop* ή *Ctrl+E*) όταν φορτωθεί πλήρως η σελίδα. Η συγκεκριμένη σελίδα περιλαμβάνει διευθύνσεις URL που αναφέρονται σε δύο εικόνες που βρίσκονται σε διαφορετικούς από τον edu-dy.cn.ntua.gr εξυπηρετητές ιστού. Μόλις η σελίδα φορτωθεί πλήρως, σταματήστε την καταγραφή.

Εφαρμόστε κατάλληλο φίλτρο ώστε να παραμείνουν μόνο τα μηνύματα του πρωτοκόλλου HTTP.

- 1.1 Εφαρμόστε κατάλληλο φίλτρο απεικόνισης (Apply a display filter) ώστε να παραμείνουν μόνο τα **πρώτα** τεμάχια TCP των τριμερών χειραψιών που διεξήχθησαν. Πόσες συνδέσεις TCP έγιναν;
- 1.2 Με ποιους υπολογιστές έγιναν οι συνδέσεις TCP;
- 1.3 Εφαρμόστε κατάλληλο φίλτρο ώστε να παραμείνουν μόνο οι εντολές HTTP προς εξυπηρετητές ιστού. Πόσες εντολές HTTP τύπου GET έχει καταγράψει το Wireshark;
- 1.4 Ποια είναι η διεύθυνση IP προορισμού κάθε εντολής HTTP τύπου GET; Εξηγήστε.
- 1.5 Καταγράψτε τα ονόματα των αρχείων εικόνων που κατέβασε ο πλοηγός από κάθε εξυπηρετητή.

Ερώτηση 2

Υπάρχουν περιπτώσεις όπου το ζητούμενο αρχείο HTML είναι αρκετά μεγάλο και δεν χωράει σε ένα τεμάχιο TCP. Στην περίπτωση αυτή, το μήνυμα HTTP τεμαχίζεται στο στρώμα μεταφοράς. Ξεκινήστε μια νέα καταγραφή με το Wireshark. Με τον πλοηγό ιστού επισκεφθείτε τη σελίδα <http://edu-dy.cn.ntua.gr/long.html>⁵ και σταματήστε την καταγραφή μόλις ολοκληρωθεί το φόρτωμα της σελίδας.

³ Η επιβεβαίωση του TCP checksum (επειδή γίνεται στην κάρτα δικτύου) παρενοχλεί τη διαδικασία επανασύνθεσης.

⁴ Παρατηρείστε ότι χρησιμοποιείται <http://> αντί για <https://>

⁵ Παρατηρείστε ότι χρησιμοποιείται <http://> αντί για <https://>

2.1 Πόσες συνδέσεις TCP έγιναν;

2.2 Ποιο είναι το μέγεθος της MSS (Maximum Segment Size) που ανακοινώνει η κάθε πλευρά;

Εφαρμόστε κατάλληλο φίλτρο απεικόνισης, ώστε να παραμείνουν μόνο μηνύματα του πρωτοκόλλου HTTP.

2.3 Πόσες αιτήσεις HTTP τύπου GET στάλθηκαν από τον πλοηγό ιστού;

2.4 Να καταγραφεί η γραμμή κατάστασης (status line) της απόκρισης που επιστρέφει ο εξυπηρετητής.

2.5 Εντοπίστε την HTTP απόκριση που αφορά το αρχείο long.html. Πόσο είναι το συνολικό μέγεθος HTTP περιεχομένου (επικεφαλίδα και δεδομένα μαζί) και πόσο είναι το μήκος του αρχείου long.html;

Εφαρμόστε νέο φίλτρο απεικόνισης στο Wireshark, ώστε να παραμείνει μόνο η κίνηση IP που προέρχεται από τον εξυπηρετητή ιστού.

2.6 Ποια είναι η σύνταξη του παραπάνω φίλτρου;

2.7 Πόσα τεμάχια TCP, που περιείχαν δεδομένα, χρειάστηκαν για τη μεταφορά της απόκρισης στο μήνυμα HTTP τύπου GET; *[Υπόδειξη: Μαζί με το τελευταίο τεμάχιο της σειράς, το Wireshark εμφανίζει στο παράθυρο καταγεγραμμένων πακέτων την ανασύνθεση της απόκρισης HTTP.]*

2.8 Ποιο τεμάχιο TCP του προηγούμενου ερωτήματος περιλαμβάνει τη γραμμή κατάστασης του πρωτοκόλλου HTTP; *[Υπόδειξη: Αναζητήστε⁶ το περιεχόμενο της γραμμής κατάστασης στο παράθυρο με τα περιεχόμενα του επιλεγμένου πλαισίου.]*

2.9 Ποιο είναι το μέγεθος του περιεχομένου HTTP που μεταφέρει κάθε ένα από τα τεμάχια αυτά; *[Υπόδειξη: Αναπτύξτε το περιεχόμενο της επικεφαλίδας TCP στο παράθυρο με τις λεπτομέρειες επικεφαλίδας.]*

2.10 Γιατί το μέγεθος των πλαισίων Ethernet που μεταφέρουν τα προηγούμενα τεμάχια TCP (πλην του τελευταίου) είναι σταθερό;

2.11 Πώς προκύπτει το μέγεθος του τελευταίου εξ αυτών;

⁶ Αφού κάνουμε κλικ στο πρώτο εμφανιζόμενο πακέτο, πληκτρολογούμε Ctrl+F. Επιλογές αναζήτησης: “Packet bytes” και “String”.

Επίλυση εργαστηριακής άσκησης

1.1:

Χρησιμοποιούμε το φίλτρο:

`tcp.flags.syn==1 and tcp.flags.ack==0`

Υπάρχουν 3 TCP συνδέσεις στην εν λόγω καταγραφή.

The image shows a Wireshark packet capture window. The filter bar at the top contains the expression `tcp.flags.syn==1 and tcp.flags.ack==0`. The packet list shows three packets:

No.	Time	Source	Destination	Protocol	Length	Info
42	4.982930	192.168.1.94	147.102.40.15	TCP	66	54069 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
54	5.300289	192.168.1.94	147.102.222.213	TCP	66	54070 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
62	5.340600	192.168.1.94	2.18.169.59	TCP	66	54071 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

The packet details pane for the selected packet (No. 42) shows:

- Frame 42: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
- Ethernet II, Src: RivetNet_8c:10:77 (9c:b6:d0:8c:10:77), Dst: Netgear_28:74:4b (e0:46:9a:28:74:4b)
- Internet Protocol Version 4, Src: 192.168.1.94, Dst: 147.102.40.15
- Transmission Control Protocol, Src Port: 54069, Dst Port: 80, Seq: 0, Len: 0

The packet bytes pane shows the raw data in hexadecimal and ASCII.

1.2:

Οι IP διευθύνσεις προορισμού είναι οι: 147.102.40.15, 147.102.222.213 και 2.18.169.59

1.3:

Εφαρμόζουμε το φίλτρο: `http.request` ώστε να παραμείνουν μόνο οι αιτήσεις HTTP από τον υπολογιστή μας προς τους υπολοίπους. Έχουν καταγραφεί 4 εντολές τύπου HTTP GET.

The image shows a Wireshark packet capture window titled 'ask2_1_imp.pcapng'. 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 is divided into three panes. The top pane, 'http.request', shows a list of captured packets. The middle pane shows the details of the selected packet (Frame 45). The bottom pane shows the raw packet data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
45	5.068430	192.168.1.94	147.102.40.15	HTTP	399	GET /links.html HTTP/1.1
70	5.387738	192.168.1.94	2.18.169.59	HTTP	352	GET /img/logo.gif HTTP/1.1
73	5.390710	192.168.1.94	147.102.222.213	HTTP	366	GET /common_images/pyrforos.gif HTTP/1.1
90	5.502351	192.168.1.94	147.102.40.15	HTTP	321	GET /favicon.ico HTTP/1.1

Frame 45: 399 bytes on wire (3192 bits), 399 bytes captured (3192 bits) on interface 0
> Ethernet II, Src: RivetNet_8c:10:77 (9c:b6:d0:8c:10:77), Dst: Netgear_28:74:4b (e0:46:9a:28:74:4b)
> Internet Protocol Version 4, Src: 192.168.1.94, Dst: 147.102.40.15
> Transmission Control Protocol, Src Port: 54069, Dst Port: 80, Seq: 1, Ack: 1, Len: 345
> Hypertext Transfer Protocol

0000 e0 46 9a 28 74 4b 9c b6 d0 8c 10 77 08 00 45 00 - F (TK... ..w...E-
0010 01 81 0d c4 40 00 80 06 6e 37 c0 a8 01 5e 93 66 - ...@... n7...^..f
0020 28 0f d3 35 00 50 17 3d 92 3b 96 63 48 b9 50 18 - (.5.P = ;cH.P-
0030 01 01 12 2c 00 00 47 45 54 20 2f 6c 69 6e 6b 73 - ...GE T /links
0040 2e 68 74 6d 6c 20 48 54 54 50 2f 31 2e 31 0d 0a - .html HT TP/1.1
0050 48 6f 73 74 3a 20 65 64 75 2d 64 79 2e 63 6e 2e - Host: ed u-dy.cn.
0060 6e 74 75 61 2e 67 72 0d 0a 55 73 65 72 2d 41 67 - ntua.gr; User-Ag
0070 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 - ent; Moz illa/5.0
0080 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e - (Window s NT 10.
0090 30 3b 20 57 69 6e 36 3a 3b 20 78 36 34 3b 20 72 - 0; Win64 ; x64; r
00a0 76 3a 36 36 2e 30 29 20 47 65 63 6b 6f 2f 32 30 - v:66.0) Gecko/20
00b0 31 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 36 - 100101 F irefox/6

Request: Boolean Packets: 113 · Displayed: 4 (3.5%) · Dropped: 0 (0.0%) Profile: Default

1.4:

Για τα 4 αυτά HTTP GET, οι IP διευθύνσεις προορισμού ήταν αντίστοιχα:

- 147.102.40.15 – Εξυπηρετητής edu-dy.cn.ntua.gr
- 2.18.169.59 – Εξυπηρετητής www.mit.edu
- 147.102.222.213 – Εξυπηρετητής old.ntua.gr
- 147.102.40.15 – Εξυπηρετητής edu-dy.cn.ntua.gr

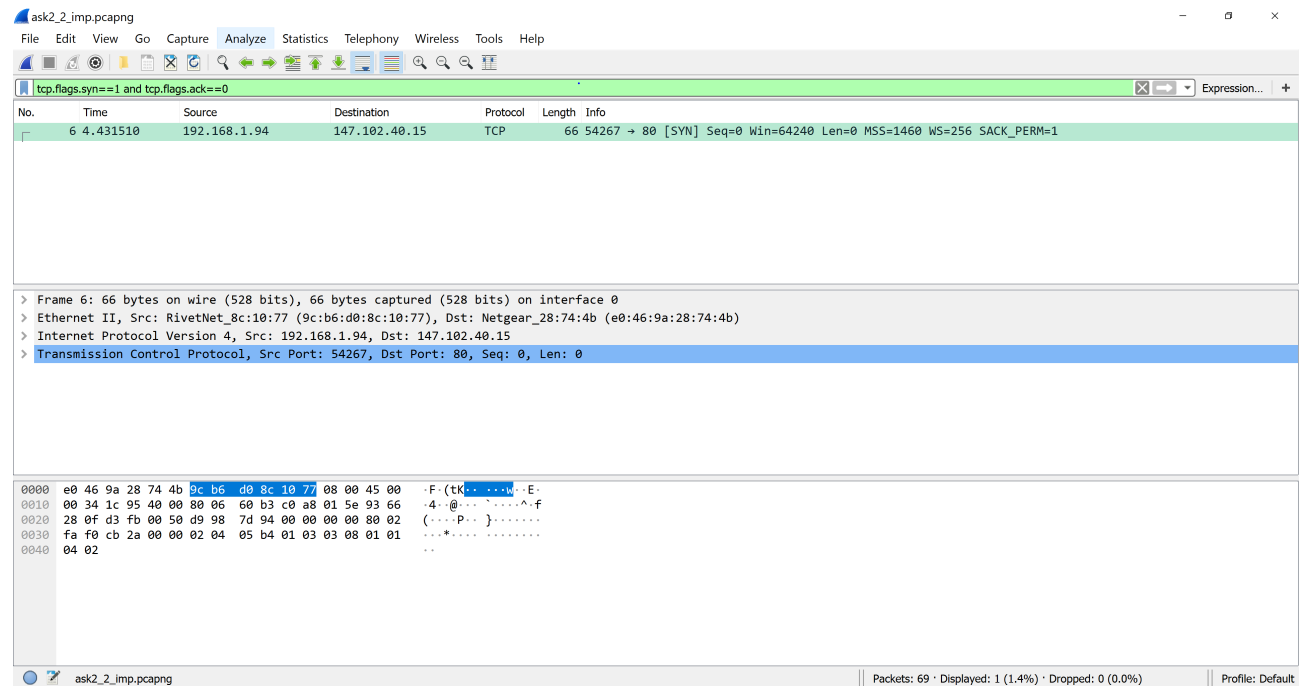
1.5:

Παρατηρούμε δύο HTTP GET προς τον υπολογιστή 147.102.40.15: ένα με σκοπό το κατέβασμα της σελίδας links.html και ένα με σκοπό το κατέβασμα του εικονιδίου favicon.ico. Επίσης, παρατηρούμε ένα HTTP GET προς τον υπολογιστή 147.102.222.213, με σκοπό το κατέβασμα της εικόνας pyrforos.gif, καθώς και ένα προς τον υπολογιστή 2.18.169.59 (του MIT), με σκοπό το κατέβασμα της εικόνας logo.gif.

Απάντηση στην Ερώτηση 2

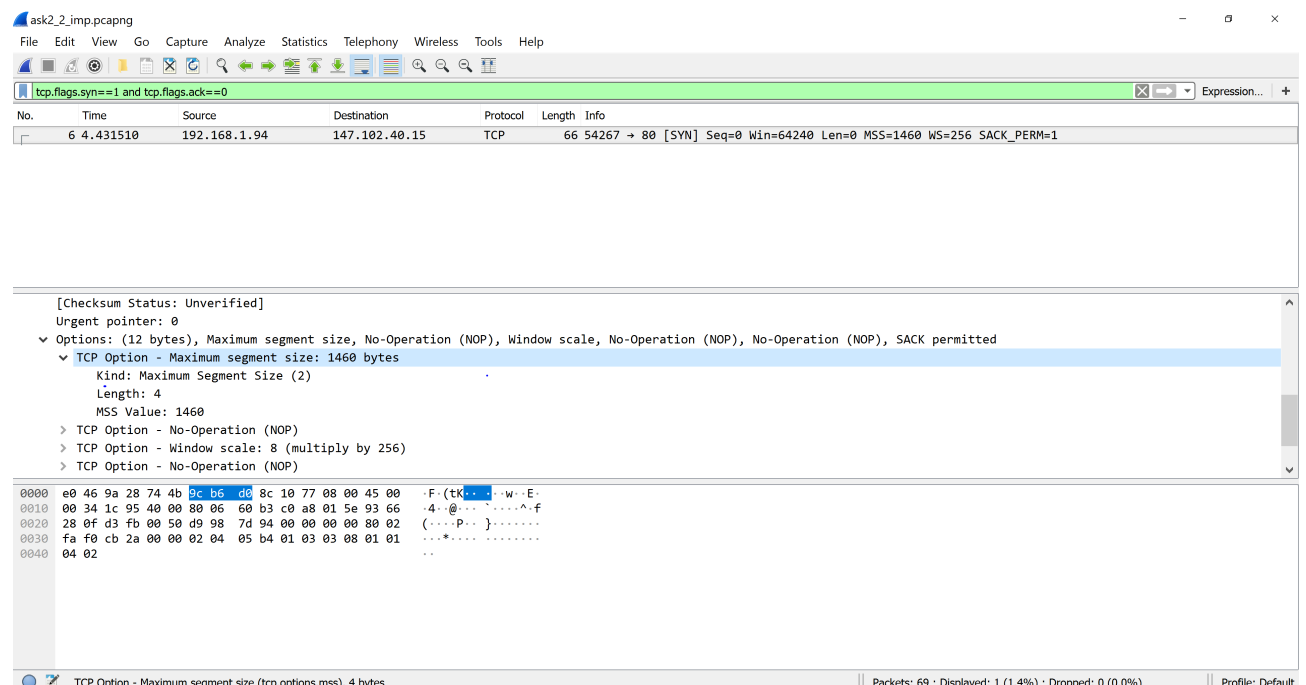
2.1:

Εφαρμόζουμε και πάλι το φίλτρο: tcp.flags.syn==1 and tcp.flags.ack==0 και διαπιστώνουμε ότι έχει εγκατασταθεί μία μόνο σύνδεση TCP.



2.2:

Αναλύουμε την επικεφαλίδα TCP. Στο συγκεκριμένο παράδειγμα του screenshot, παρατηρούμε ότι η πλευρά του δικού μας υπολογιστή ανακοινώνει MSS ίσο με 1460 bytes.



Εν συνεχεία, τροποποιούμε το φίλτρο ώστε να παραμείνουν μόνο τα δεύτερα τεμάχια κάθε τριμερούς χειραψίας. Παρατηρούμε ότι η πλευρά του εξυπηρετητή ανακοινώνει MSS ίσο με 536 bytes.

The image shows a Wireshark packet capture of a TCP segment. The packet list pane shows a packet from 147.102.40.15 to 192.168.1.94, protocol TCP, length 66 bytes. The packet details pane shows the TCP segment with the following options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, SACK permitted, End of Option List (EOL). The Maximum segment size option is expanded, showing a Kind: Maximum Segment Size (2), Length: 4, and MSS Value: 536. The packet bytes pane shows the raw data of the segment, with the MSS value 536 highlighted in blue.

Σημείωση: Ως MSS θα επιλεγεί η μικρότερη εκ των δύο τιμών που ανακοινώνονται, δηλ. εν προκειμένω η τιμή των 536 bytes.

2.3:

Εφαρμόζουμε το φίλτρο http και παρατηρούμε ότι στάλθηκαν δύο HTTP αιτήσεις τύπου GET.

The image shows a Wireshark packet capture of HTTP traffic. The packet list pane shows four packets, all of type GET. The packet details pane shows the first packet, which is a GET request for /long.html. The packet bytes pane shows the raw data of the request, including the HTTP headers and the body.

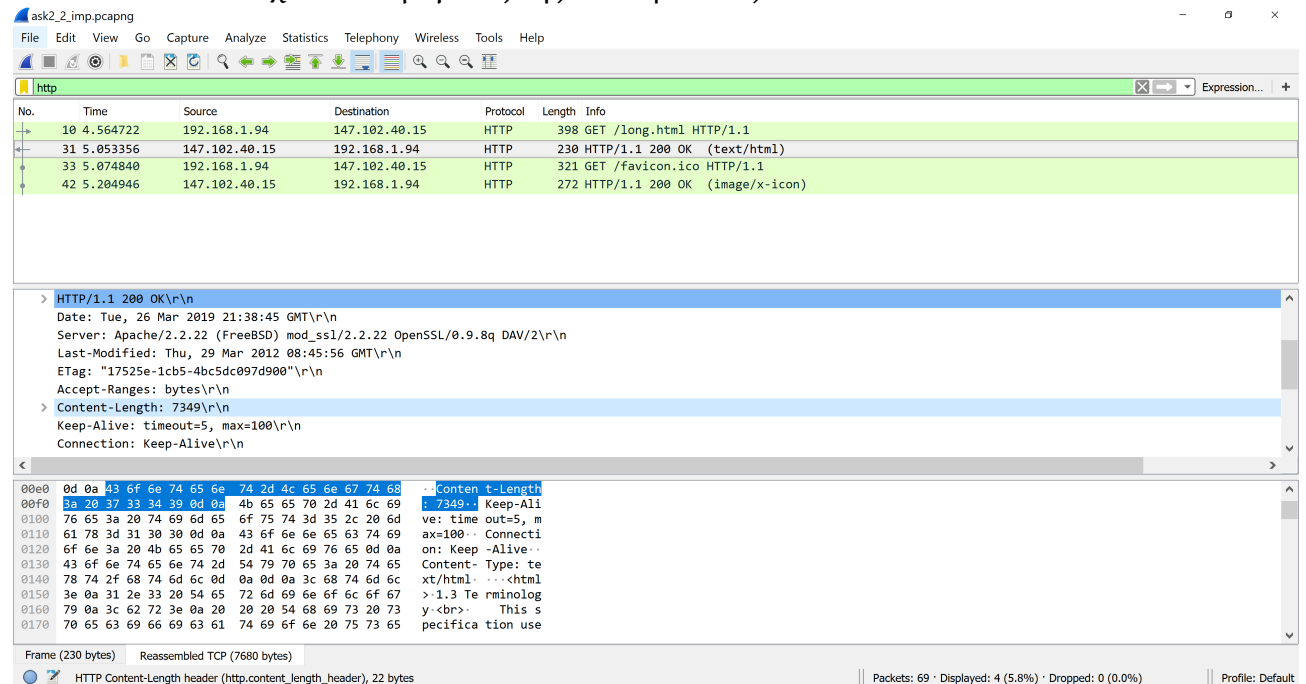
2.4:

Η απόκριση του εξυπηρετητή είναι HTTP/1.1 200 OK και στις δύο περιπτώσεις.

2.5:

Επικεντρωνόμαστε στην απόκριση που αφορά το αντικείμενο long.html. Αναλύουμε την επικεφαλίδα HTTP και εντοπίζουμε το πεδίο Content-Length, που έχει τιμή 7349 bytes. Αυτό σημαίνει ότι το μέγεθος της σελίδας long.html είναι ίσο με 7349 bytes.

Παρατηρούμε, επίσης, ότι το συνολικό HTTP περιεχόμενο έχει μέγεθος 7680 bytes, εκ των οποίων τα 7349 bytes είναι το μέγεθος της σελίδας ενώ τα υπόλοιπα αντιστοιχούν στο μέγεθος της επικεφαλίδας HTTP.



2.6:

Επειδή θέλουμε να εμφανίζεται η κίνηση που προέρχεται από τον εξυπηρετητή 147.102.40.15, επιλέγουμε ως φίλτρο το: ip.src==147.102.40.15

2.7:

Χρειάστηκαν συνολικά 15 τεμάχια TCP. *Σημείωση:* Το πρώτο τεμάχιο TCP που εμφανίζεται στο screenshot δεν το μετράμε, καθώς αντιστοιχεί στην δεύτερο μέρος της τριμερούς χειραψίας. Ωστόσο, σύμφωνα με την υπόδειξη που μας δίνεται, συμπεριλαμβάνουμε στην καταμέτρηση το επισημασμένο (επιλεγμένο) τεμάχιο που φαίνεται στο screenshot (αυτό για το οποίο εμφανίζεται HTTP στη στήλη Protocol), επειδή είναι το τελευταίο της σειράς.

The screenshot shows a Wireshark packet capture of an HTTP 200 OK response. The packet list on the left shows a sequence of TCP segments (ACKs) and an HTTP packet. The selected packet (No. 31) is an HTTP 200 OK response from 147.102.40.15 to 192.168.1.94. The packet details pane shows the HTTP structure, including the status line 'HTTP/1.1 200 OK (text/html)', headers like 'Date: Tue, 26 Mar 2019 21:38:45 GMT' and 'Server: Apache/2.2.22', and the body content. The packet bytes pane shows the raw data, with a search for 'Content-Length' highlighting the value '7349'.

No.	Time	Source	Destination	Protocol	Length	Info
8	4.563418	147.102.40.15	192.168.1.94	TCP	66	80 → 54267 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=536 WS=64 SACK_PERM=1
11	4.716778	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=1 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
12	4.716780	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=537 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
14	4.717996	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=1073 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
15	4.717998	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=1609 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
17	4.885967	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=2145 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
18	4.885969	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=2681 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
19	4.885970	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=3217 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
20	4.885970	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=3753 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
23	4.891159	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=4289 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
24	4.891161	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=4825 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
25	4.891162	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=5361 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
26	4.891163	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=5897 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
29	5.053354	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=6433 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
30	5.053355	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=6969 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
31	5.053356	147.102.40.15	192.168.1.94	HTTP	230	HTTP/1.1 200 OK (text/html)
34	5.204114	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=7681 Ack=612 Win=65920 Len=536 [TCP segment of a reassembled PDU]
35	5.204116	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=8217 Ack=612 Win=65920 Len=536 [TCP segment of a reassembled PDU]

Transmission Control Protocol, Src Port: 80, Dst Port: 54267, Seq: 7505, Ack: 345, Len: 176
> [15 Reassembled TCP Segments (7680 bytes): #11(536), #12(536), #14(536), #15(536), #17(536), #18(536), #19(536), #20(536), #23(536), #24(536), #25(536), #26(536), #29(536), #30(536), #31(536)]
Hypertext Transfer Protocol
> HTTP/1.1 200 OK\r\n
Date: Tue, 26 Mar 2019 21:38:45 GMT\r\n
Server: Apache/2.2.22 (FreeBSD) mod_ssl/2.2.22 OpenSSL/0.9.8q DAV/2\r\n
Content-Length: 7349
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive

Frame (230 bytes) Reassembled TCP (7680 bytes)
HTTP Content-Length header (http.content_length_header), 22 bytes

Packets: 69 · Displayed: 26 (37.7%) · Dropped: 0 (0.0%) Profile: Default

2.8:

Για να εκτελέσουμε την αναζήτηση, αρχικά κάνουμε κλικ στο πρώτο πακέτο (πάνω πάνω) και κατόπιν επιλέγουμε Edit>Find Packet ή πληκτρολογούμε Ctrl+F. Εν συνεχεία, επιλέγουμε να κάνουμε αναζήτηση μέσα στα “Packet bytes” και να αναζητήσουμε ένα “String”. Στο πεδίο αναζήτησης θα πρέπει να πληκτρολογήσουμε ως κείμενο αναζήτησης το: 200 OK (με αγγλικούς χαρακτήρες).

Όταν πατήσουμε το Find, θα επισημανθεί το πακέτο που περιέχει τον συγκεκριμένο κωδικό. Όπως αναμενόταν, το πακέτο αυτό είναι το πρώτο από τα 15 της σειράς (βλ. Screenshot).

ask2_2_imp.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.src==147.102.40.15

Packet bytes Narrow & Wide Case sensitive String 200 OK Find Cancel

No.	Time	Source	Destination	Protocol	Length	Info
8	4.563418	147.102.40.15	192.168.1.94	TCP	66	80 → 54267 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=536 WS=64 SACK_PERM=1
11	4.716778	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=1 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
12	4.716780	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=537 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
14	4.717996	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=1073 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
15	4.717998	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=1609 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
17	4.885967	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=2145 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
18	4.885969	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=2681 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
19	4.885970	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=3217 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
20	4.885970	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=3753 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
23	4.891159	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=4289 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
24	4.891161	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=4825 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
25	4.891162	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=5361 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
26	4.891163	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=5897 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
29	5.053354	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=6433 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
30	5.053355	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=6969 Ack=345 Win=65920 Len=536 [TCP segment of a reassembled PDU]
31	5.053356	147.102.40.15	192.168.1.94	HTTP	230	HTTP/1.1 200 OK (text/html)
34	5.204114	147.102.40.15	192.168.1.94	TCP	590	80 → 54267 [ACK] Seq=7681 Ack=612 Win=65920 Len=536 [TCP segment of a reassembled PDU]

Urgent pointer: 0

- > [SEQ/ACK analysis]
- > [Timestamps]
- TCP payload (536 bytes)
- [Reassembled PDU in frame: 31]
- TCP segment data (536 bytes)

0030 04 06 8f cd 00 00 48 54 54 50 2f 31 2e 31 20 32HT IP/1.1 2

0040 30 30 20 4f 4b 0d 0a 44 61 74 65 3a 20 54 75 65 00 OK..D ate: Tue

0050 2c 20 32 36 20 4d 61 72 20 32 30 31 39 20 32 31 , 26 Mar 2019 21

0060 3a 33 38 3a 34 35 20 47 4d 54 0d 0a 53 65 72 76 :38:45 G MT..Serv

0070 65 72 3a 20 41 70 61 63 68 65 2f 32 2e 32 2e 32 er: Apac he/2.2.2

A data segment used in reassembly of a lower-level protocol (tcp.segment_data), 536 bytes

Packets: 69 · Displayed: 26 (37.7%) · Dropped: 0 (0.0%) Profile: Default

2.9:

Όπως φαίνεται και στο παραπάνω screenshot, το μέγεθος περιεχομένου HTTP (δηλ. TCP payload) στο εν λόγω τεμάχιο αλλά και στα υπόλοιπα της σειράς με εξαίρεση το τελευταίο είναι: 536 bytes (όσο δηλ. και το MSS).

2.10:

Το μέγεθος των 14 πρώτων από τα 15 πλαίσια της σειράς είναι σταθερό και ίσο με 590 bytes, το οποίο προκύπτει ως εξής: 14 bytes (Ethernet επικεφαλίδα) + 20 bytes (IP επικεφαλίδα) + 20 bytes (TCP επικεφαλίδα) + 536 bytes (HTTP περιεχόμενο) = 590 bytes. Σε όλα αυτά τα τεμάχια, το TCP payload ταυτίζεται με το MSS και είναι ίσο με 536 bytes.

The image shows a Wireshark packet capture analysis of a TCP segment. The packet list shows three TCP segments (No. 8, 11, 12) from source 147.102.40.15 to destination 192.168.1.94. The selected packet (No. 12) is a TCP segment with sequence number 537, acknowledgment number 345, and length 536. The packet details pane shows the following information:

- Source Port: 80
- Destination Port: 54267
- [Stream index: 0]
- [TCP Segment Len: 536]
- Sequence number: 537 (relative sequence number)
- [Next sequence number: 537 (relative sequence number)]
- Acknowledgment number: 345 (relative ack number)
- 0101 = Header Length: 20 bytes (5)
- Flags: 0x010 (ACK)
- Window size value: 1030
- [Calculated window size: 65920]
- [Window size scaling factor: 64]
- Checksum: 0x8fcd [unverified]
- [Checksum Status: Unverified]
- Urgent pointer: 0
- [SEQ/ACK analysis]
- [Timestamps]
- TCP payload (536 bytes)
- [Reassembled PDU in frame: 311](#)
- TCP segment data (536 bytes)

The packet bytes pane shows the raw data of the TCP segment, starting with 0030 04 06 8f cd 00 00 48 54 54 50 2f 31 2e 31 20 32. The status bar indicates that the data segment is used in reassembly of a lower-level protocol (tcp.segment_data), 536 bytes.

2.11:

Όπως είδαμε στο ερώτημα 2.5, το συνολικό μέγεθος HTTP περιεχομένου (δεδομένα και επικεφαλίδα) είναι 7680 bytes. Με τα 14 πρώτα τεμάχια της σειράς μεταφέρθηκαν: $14 \times 536 = 7504$ bytes. Επομένως, απομένουν $7680 - 7504 = 176$ bytes να μεταφερθούν. Το 15ο πλαίσιο έχει μέγεθος 230 bytes διότι: 14 bytes (Ethernet επικεφαλίδα) + 20 bytes (IP επικεφαλίδα) + 20 bytes (TCP επικεφαλίδα) + 176 bytes (εναπομείναν HTTP περιεχόμενο) = 230 bytes.

The image shows a Wireshark packet capture analysis of an HTTP segment. The packet list shows several TCP segments (No. 18 to 31) from source 147.102.40.15 to destination 192.168.1.94. The selected packet (No. 31) is an HTTP segment with sequence number 7505, acknowledgment number 345, and length 176. The packet details pane shows the following information:

- Source Port: 80
- Destination Port: 54267
- [Stream index: 0]
- [TCP Segment Len: 176]
- Sequence number: 7505 (relative sequence number)
- [Next sequence number: 7681 (relative sequence number)]
- Acknowledgment number: 345 (relative ack number)
- 0101 = Header Length: 20 bytes (5)

The packet bytes pane shows the raw data of the HTTP segment, starting with 0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d. The status bar indicates that the data segment is used in reassembly of a lower-level protocol (http), 331 bytes.