Όνοματεπώνυμο: Πυλιώτης Αθανάσιος		**Ομάδα: 3**
Όνομα PC/ΛΣ: DESKTOP-5DLG3IF		**Ημερομηνία: 22/11/2022**
Διεύθυνση IP: 147.102.202.133	**Διεύθυνση MAC:**	**98:54:1b:bd:69:97**

Εργαστηριακή Άσκηση 7
Πρωτόκολλα TCP και UDP

Απαντήστε στα ερωτήματα στον χώρο που σας δίνεται παρακάτω και στην πίσω σελίδα εάν δεν επαρκεί. Το φυλλάδιο αυτό θα παραδοθεί στον επιβλέποντα.

1

1.1 host 147.102.202.133

1.2 ip.dst == 1.1.1.1 or ip.dst == 2.2.2.2 or ip.dst == 147.102.40.1

1.3 Destination Port 23

1.4 tcp.port == 23

1.5 Syn στο πακέτο TCP

1.6 5 προσπάθειες

1.7 A: 1.009526, 2.009928, 4.009885, 8.02053

B: 1.004924, 2.019078, 4.000531, 8.000435

1.8 Παρατηρούμε πως (με ελάχιστες αποκλείσεις) η λογική των προσπαθειών είναι η ίδια: Στέλνει μια πρώτη απευθείας μήπως πάρει απάντηση. Μετά προσπαθεί ξανά (retransmission) και τη πρώτη φορά περιμένει 1 sec, τη δεύτερη περιμένει 2 sec, τη Τρίτη 4 sec και τη τέταρτη 8 sec, πρακτικά διπλασιάζοντας τον χρόνο που παίρνει μεταξύ των διαδοχικών προσπαθειών κάθε φορά με την ελπίδα ότι θα γίνει η σύνδεση.

1.9 Την πρώτη μόνο, την αίτηση σύνδεσης με τον εξυπηρετητή TCP, που έχει το flag SYN που αναφέραμε και παραπάνω, ενώ δεν έλαβε ποτέ ACK και SYN απάντηση από τον εξυπηρετητή.

1.10 Απλώς εγκαταλείπει τη προσπάθεια, αφού δεν στέλνει πακέτο σχετικό με απόλυση σύνδεσης.

1.11 tcp and ip.host == 147.102.40.1

1.12 5 προσπάθειες

1.13 Η βασική διαφορά είναι πως λαμβάνουμε απάντηση από τον εξτυπηρετητή με flags ACK, RST, άρα δεν θέλει να συνδεθούμε. Επειδή δεν το αποδεχόμαστε, κάνουμε retransmission άλλες 4 φορές μετά από περίπου 0.5 sec μήπως και μας στείλει flags ACK, SYN. Αυτό γίνεται τόσο γρήγορα επειδή ξέρουμε πως έγινε η σύνδεση, ενώ στην άλλη δίναμε χρόνο μήπως υπάρχει σφάλμα σύνδεσης.

1.14 flags ACK, RST

1.15 flag RST

1.16 Header: 20 bytes, Data: 0 bytes

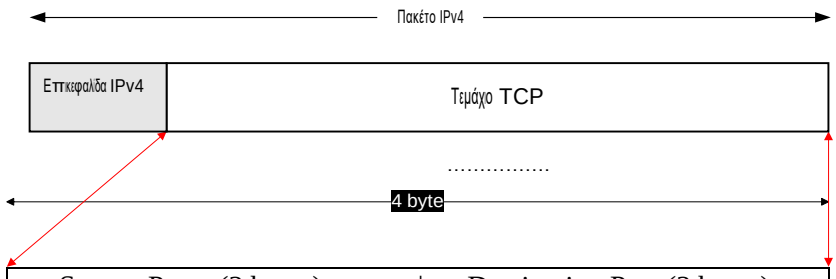
1.17 Transmission Control Protocol, Src Port: 23, Dst Port: 49272, Seq: 1, Ack: 1, Len: 0

Source Port: 23

Destination Port: 49272

[Stream index: 13]

[Conversation completeness: Incomplete (37)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 0
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 2961042056
0101 .... = Header Length: 20 bytes (5)
Flags: 0x014 (RST, ACK)
000. .... = Reserved: Not set
...0 .... = Accurate ECN: Not set
.... 0... .... = Congestion Window Reduced: Not set
.... .0.. .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
....1 .... = Acknowledgment: Set
.... 0... = Push: Not set
.... .1.. = Reset: Set
....0. = Syn: Not set
....0 = Fin: Not set
[TCP Flags: ·····A·R··]
Window: 0
[Calculated window size: 0]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x3ae7 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]



Source Port (2 bytes)	Destination Port (2 bytes)
Sequence Number (4 bytes)	
Acknowledgement number (4 bytes)	
Header Length(4 bits) | Flags (12 bits)	Window (2 bytes)
Checksum (2 bytes)	Urgent Pointer (2 bytes)

1.18 Online: Data Offset. Wireshark: Header Length

1.19 Είναι το πόσες φορές τα 32 bits έχει το header (γιατί είναι πολλαπλάσιο του 32 bits). Αφού έχουμε 5 είναι 5*32bits ή 5*4bytes = 20bytes.

1.20 Όχι

1.21 IPv4 Total Length – IPv4 Header Length = 52 – 20 = 32 bytes.

1.22 32 bytes

1.23 Ναι υπάρχει. Η διαφορά οφείλεται πως όταν του στέλνει το πρώτο μέρος της χειραψίας, στέλνει επίσης και τις επιλογές που είπαμε πως πρέπει να αποδεχτούν οι εξυπηρετητές μεταξύ τους. Συνεπώς στέλνει 12 bytes παραπάνω για τα options :

Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

TCP Option - Maximum segment size: 1460 bytes

TCP Option - No-Operation (NOP)

TCP Option - Window scale: 8 (multiply by 256)

TCP Option - No-Operation (NOP)

TCP Option - No-Operation (NOP)

TCP Option - SACK permitted

2

2.1 tcp and host == 147.102.202.133

2.2 Port 21

2.3 Port 20

2.4 tcp.port == 21

2.5 3 τεμάχια της τριπλής χειραψίας

2.6 SYN στο πρώτο, SYN, ACK στο δεύτερο, ACK στο τρίτο.

2.7 32 bytes, 32 bytes, 20 bytes

2.8 Δεν έχουν δεδομένα (μόνο ό,τι υπάρχει στην επικεφαλίδα, δηλαδή τα options των 12 bytes τα οποία πρέπει να συμφωνήσουν).

2.9 0.004387 + 0.000124 = 0.004511 sec

2.10 [iRTT: 0.004511000 seconds] ναι

2.11 Sequence Number (raw): 944934205 (πρώτο)

Sequence Number (raw): 3678604212 (Δεύτερο)

Στο δεύτερο το ACK στέλνει το sequence number του πρώτου και μετά το τρίτο πακέτο στέλνει στο ACK το sequence Number του δεύτερου, για να τα κάνουν acknowledge.

2.12 Στο πακέτο απάντησης από τον εξυπηρετητή FTP το ACK στέλνει το sequence number του πρώτου και μετά το τελευταίο πακέτο στέλνει στο ACK το sequence Number του δεύτερου, για να τα κάνουν acknowledge τους αριθμούς ο ένας του άλλου και αποδέχονται τη σύνδεση.

2.13 Στο πακέτο απάντησης από τον εξυπηρετητή το ACK στέλνει το sequence number του πρώτου και μετά το ΤΕΛΕΥΤΑΊΟ πακέτο στέλνει στο ACK το sequence Number του δεύτερου, για να τα κάνουν acknowledge τους αριθμούς ο ένας του άλλου και αποδέχονται τη σύνδεση.

2.14 Δεν στέλνουν δεδομένα

2.15 2^{32} $αριθμοί$ $αφού$ $είναι$ 32 $bits$.

2.16 tcp.port == 21 and (tcp.options or tcp.window_size == 8192) (προφανώς το window_size αλλάζει αλλά δουλεύει αυτό). Ή tcp.port == 21 and (tcp.options or tcp.ack == 1) αλλά δείχνει κι ένα άλλο.

2.17 Window: 8192 όταν στέλνουμε εμείς. Άρα δεν μπορεί να μας στείλει περισσότερα.

2.18 Αλλά εμείς μπορούμε να στείλουμε Window: 65535

2.19 Window

2.20 TCP Option - Window scale: 6 (multiply by 64)

2.21 Στα options της τριπλής χειραψίας είναι

2.22 TCP Option - Maximum segment size: 1460 bytes

2.23 MTU – 40 = 1460 bytes → MTU = 1500 bytes, τα οποία είναι ίδια με τη max τιμή του MTU για standard ethernet. (άρα από 1500 – 40 επειδή έχουμε IPv4.)

2.24 TCP Option - Maximum segment size: 1460 bytes στα options της τριπλής χειραψίας στο πακέτο με flag SYN που στέλνουμε.

2.25 TCP Option - Maximum segment size: 536 bytes

2.26 MTU – 40 = 576 – 40 = 536 bytes, άρα λόγω IPv4 η MSS είναι MTU βγάζω τις κεφαλίδες.

2.27 536 bytes non-fragmented (έχει 40 bytes κεφαλίδες IPv4 και TCP).

2.28 Η σημαία fin.

2.29 tcp.port == 21 and tcp.flags.fin == 1 (το πρώτο χρησιμοποιήθηλε για να βρούμε μόνο τα πακέτα που μας ενδιαφέρουν)

2.30 Ο εξυπηρετητής με τον οποίο είμαστε συνδεδεμένοι.

2.31 2, ένα από τον άλλο εξυπηρετητή στον υπολογιστή μας, κι ένα από τον υπολογιστή μας στον άλλο TCP.

2.32 20 bytes

2.33 Δεν μεταφέρουν δεδομένα. Πρακτικά μόνο τα flags χρειάζεται να στείλει το ένα στο άλλο, άρα δεν χρειάζεται να στείλουν κάτι άλλο.

2.34 40 bytes, μόνο τα IPv4 και TCP Header. Δεν στένουν άλλα δεδομένα καθώς δεν χρειάζεται.

2.35 Ίδιο για ίδιο λόγο, 40 bytes.

2.36 80 bytes

2.37 Πρόσθεσα τα bytes των δύο πακέτων που ανταλλάσσονται κατά τον τερματισμό.

2.38 tcp.port == 20

2.39 TCP Option - Maximum segment size: 536 bytes η πλευρά του εξυπηρετητή.

TCP Option - Maximum segment size: 1460 bytes η πλευρά μας.

2.40 536 Bytes.

2.41 [iRTT: 0.002165000 seconds]

2.42 Όχι, στέλνει πολύ λιγότερα πακέτα.

2.43 118 packets

2.44 27 packets

2.45 Window: 4097

2.46 όχι είναι το μισό μέγεθος +1 byte ακόμα.

2.47 Όχι παραμένει σταθερή.

2.48 Λογικά θα ξαναέστελνε τα πακέτα που έστειλε πριν, καθώς θα είναι σαν να μην τα έλαβε.

2.49 Frame Length: 590 bytes (4720 bits), IPv4 .... 0101 = Header Length: 20 bytes (5), TCP Header 1000 .... = Header Length: 32 bytes (8)

2.50 Όχι, έχει μεγαλύτερο Header το TCP εδώ, άρα θα είναι κατά 12 bytes μικρότερη. (524 bytes)

2.51 Θα έσπαγε το πακέτο σε fragments στη πηγή, αλλά γενικά πρέπει να ξέρουν πως ο host μπορεί να τα δεχτεί.

2.52 2743132188 – 2743073891 = 58297 bytes

2.53 61440 bytes received in 0.66Seconds 92.53Kbytes/sec.

2.54 όχι. Οι αναμεταδόσεις στο Wireshark φαίνονται με διαφορετικό χρώμα (έγιναν απλά όχι για μεταφορά δεδομένων, απλά για μεταφορά syn flag.

3

3.1 Για να εμφανίζονται μόνο δεδομένα: ftp-data

3.2 94.65.141.44

3.3 0.014626, μεγαλύτερος κατά πολύ από τον προηγούμενο

3.4 Το πλήθος που στέλνεται κάθε φορά διπλασιάζεται, άρα αυξάνεται εκθετικά.

3.5 Είναι σύμφωνο, έστειλε συνολικά 4 τεμάχια.

3.6 Τα τεμάχια που έστειλε κατά το δεύτερο RTT ήταν 6, κατά το τρίτο 11, κατά το τέταρτο 18. Αυξάνεται με τον αριθμό των ACK που λαμβάνει.

3.7 Τα ACK που στάλθηκαν αντίστοιχα είναι 1, 8 και 14

3.8 Ναι είναι παρόμοιο. Στο πρώτο έστειλε 4 τεμάχια, όσα προηγουμένως, στο δεύτερο 4 τεμάχια (λιγότερα) και στο τρίτο 2 τεμάχια (λιγότερα πάλι) .

4

4.1 capture filter: udp

4.2 Source Port (2 bytes), Destination Port (2 bytes), Length (2 bytes), Checksum (2 bytes)

4.3 8 bytes

4.4 98 bytes (8 bytes header + 90 bytes data)

4.5 UDP header bytes + UDP data bytes

4.6 8 Bytes, only header

4.7 minimum data size is 0 and maximum size for IPv4 packets is 576 bytes. Thus, 576 – 20 – 8 = 548 bytes of UDP Payload (when headers of IPv4 and UDP are minimum).

4.8 MAX IPv4 – Ipv4 Header, thus 556 bytes maximum size of UDP.

4.9 Nope, nothing else.

4.10 display filter: dns

4.11 IPv6 address: fe80::1, which is my default gateway atm.

4.12 User Datagram Protocol, Src Port: 62215, Dst Port: 53

4.13 User Datagram Protocol, Src Port: 53, Dst Port: 62215

4.14 Η 53 καθώς η θύρα 62215 είναι για ιδιωτική χρήση όπως γνωρίζουμε.