

פרויקט רשותת תקשורת – דוח מסכם חלק 1

מגישים:

- אורי אמסלם – 206508731
- אופק דרורי – 211785027
- שרון שאול – 209321314

שלב 1: ייצור קובץ הנתונים (CSV)

לצורך ביצוע הסימולציה, בחרנו להתמקד בפרוטוקול HTTP בשכבה היפיימוס.

Traffic את קובץ הקלט (group02_http_input.csv) הפכנו באמצעות עיבוד של לכידת תעבורת אמיתית (Capture) וסינון המידע הרלוונטי.

תהליך העבודה כלל:

1. **סינון ומיפוי:** סיננו חבילות HTTP מתוך קובץ PCAP גולמי וביצענו מיפוי של כתובות ה-IP של `.web_server` ו-`client_browser`.
2. **בנייה התרבות:** הקובץ מדמה תהליך גלייה סטנדרטי הכלול:
 - בקשה **GET** למשיכת קבצים (HTML).
 - בקשה **POST** המדמת שליחת נתונים לשרת.
 - תשובה שלט (Status 200 OK) בהתאם לגודל המידע (Payload) המקורי.
3. **התאמת זמנים:** שמרנו על הפרשי הזמן (Timestamp) המקורי שהסימולציה ב-Wireshark תהייה אמינה ותשקף Latency אמיתי של רשת.

msg_id	app_protocol	src_port	dst_port	message	timestamp
1	HTTP	client_browser	web_server	GET /index.html HTTP/1.1	0.026
2	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 256)	0.03
3	HTTP	client_browser	web_server	GET /style.css HTTP/1.1	0.041
4	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 512)	0.045
5	HTTP	client_browser	web_server	GET /script.js HTTP/1.1	0.048
6	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 2920)	0.055
7	HTTP	client_browser	web_server	GET /images/logo.png HTTP/1.1	0.065
8	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 1460)	0.072
9	HTTP	client_browser	web_server	POST /api/data HTTP/1.1	0.084
10	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 1460)	0.09
11	HTTP	client_browser	web_server	GET /index.html HTTP/1.1	0.1
12	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 1460)	0.107
13	HTTP	client_browser	web_server	GET /style.css HTTP/1.1	0.115
14	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 512)	0.118
15	HTTP	client_browser	web_server	GET /script.js HTTP/1.1	0.13
16	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 1460)	0.137
17	HTTP	client_browser	web_server	GET /images/logo.png HTTP/1.1	0.144
18	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 512)	0.147
19	HTTP	client_browser	web_server	POST /api/data HTTP/1.1	0.152
20	HTTP	web_server	client_browser	HTTP/1.1 200 OK (size 512)	0.161

שלב 2: הרצת סימולציה ולכידה ב-Wireshark

לאחר הcnnet הקובץ, טענו אותו לSUBBIT העובדה (Jupyter/VSCode) והרכנו את הסקריפט המדמה את שליחת ההודעות (Encapsulation). במקביל, ביצענו האזנה לטעורה באמצעות Wireshark על הממשק המקומי (Loopback / Adapter).

ממצאים וניתוח:

זההינו התאמה מלאה בין הנתונים הלוגיים בקובץ CSV לבין המנות שנלכדו בפועל:

- **Encapsulation :** ראיינו כיצד ההודעה משכבה האפליקציה (למשל GET /index.html) נועפה בתוך סגמנט TCP, לאחר מכן בחבילת IP ולבסוף בתוכן Ethernet Frame.
- **דגלים (Flags):** בטעורה ה-HTTP, הבחנו בדגל-h Push מופעל יחד עם ה-ACK בעת העברת המידע, מה שמייד עלה דחיפת מיידית של המידע לשכבה האפליקציה.

No.	Time	Source	Destination	Protocol	Length	Info
221	10.681528	127.0.0.1	127.0.0.1	TCP	68	54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=24
222	10.681683	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
223	10.783358	127.0.0.1	127.0.0.1	TCP	70	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=26
224	10.783427	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
225	10.885566	127.0.0.1	127.0.0.1	TCP	67	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=23
226	10.885635	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
227	10.987245	127.0.0.1	127.0.0.1	TCP	70	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=26
228	10.987309	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
229	11.088610	127.0.0.1	127.0.0.1	TCP	67	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=23
230	11.088674	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
231	11.190061	127.0.0.1	127.0.0.1	TCP	71	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=27
232	11.190125	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
233	11.291726	127.0.0.1	127.0.0.1	TCP	73	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=29
234	11.291805	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
235	11.393218	127.0.0.1	127.0.0.1	TCP	71	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=27
236	11.393282	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
237	11.494527	127.0.0.1	127.0.0.1	TCP	67	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=23
238	11.494594	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
239	11.595895	127.0.0.1	127.0.0.1	TCP	71	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=27
240	11.595957	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0
241	11.697455	127.0.0.1	127.0.0.1	TCP	68	[TCP Retransmission] 54163 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=24
242	11.697509	127.0.0.1	127.0.0.1	TCP	44	12345 → 54163 [RST] Seq=1 Win=0 Len=0

אתגרים ופתרונות

במהלך העבודה נתקלנו במנות RST שהופיעו בלכידה.

מניתוח התופעה עליה כי מערכת הפעלה שלחה הודעות אלו כאשר התקבלו חבילות לפורט סגור (לא תהליך מאزن מצד השני). הפתרון היה לוודא סyncron בין שליחת הבקשות לבין הרצת השרת המאזין לפני תחילת השידור.