מגישים:
319097036
312512619
חלק א'

1.מכיוון שDOH משתמש בפרוטוקול TCP וDNS רגיל משתמש בUDP יתרון נוסף מעבר לאבטחה הוא שבמצב של איבוד פקטות לפי פרוטוקול TCP לאחר 2 RTT הבקשה לפקטה שנאבדה תשלח שוב בעוד לפי פרוטוקול UDP הבקשה תשלח שוב רק לאחר זמן קבוע דבר שכנראה יקח יותר זמן ויגרום למעבר הפקטות להיות איטי יותר במידה ויש איבוד רב של פקטות

2. א.מכיוון שהחיבור של DOH מאובטח ומוצפן הוא מקשה על ארגונים המשתמשים בו לבדוק האם קיימים וירוסים בבקשה
ב.מכיוון שDOH הינה טכנולוגיה חדישה ישנם DNS רבים שעוד לא יכולים לעבוד איתה

3.אפשר להתקין פרוקסי על הרשת המקומית וכך השימוש ברשת המקומית יתבצע בצורה רגילה ובלי אבטחה והדבר לא יפריע אבל בשליחת בקשות שאינן ברשת המקומית אלא בקשות באינטרנט נשתמש בDOH והחיפוש יהיה מאובטח.

4.

סוג המימוש	יתרונות	חסרונות
מימוש DoH ברמת האפליקציות	אופציה זו נותנת למשתמש להנות מיתרונות הDOH ובעזרת התקנה בדפדפן בלבד	ברמת האפליקציה יכול להיות שיתבצע דילוג על בקשות DOH מבלי שהמשתמש יקבל התראה על כך
מימוש DoH ברמת שרת proxy* ברשת	אפשרות זו נותנת למשתמש להשתמש בDOH במחשב שלא יכול להתקין את הplugin	הבקשות נהיות מאובטחות רק אחרי שהן עוברות את שרת הפרוקסי
מימוש DoH ברמת שרת proxy מקומי	אפשרות זו נותנת למשתמש להשתמש בDOH במחשב שלא יכול להתקין את הplugin	הבקשות נהיות מאובטחות רק אחרי הרשת המקומית
התקנת plugin המממש DoH	הבקשות מאובטחות מהרגע הראשון גם למכשירים הקיימים ברשת המקומית וגם באינטרנט	לא כל מחשב יוכל להשתמש במימוש זה

המימוש המועדף לדעתנו הוא מימוש DoH ברמת שרת proxy מקומי מכיוון שהוא מאפשר להשתמש בDOH גם במכשירים שאינם יכולים להתקין את הplugin והוא נותן כמעט את כל היתרונות שDOH יכול להציע לנו מלבד העובדה שהבקשות יהיו מאובטחות רק מעבר לרשת המקומית אך זהו ויתור שמאפשר לנו להפעיל את זה בכל המכשירים ולא רק באלו שאפשר להתקין בהם את הplugin

5.יתרון ברור שיש לDOH לעומת 53DO הוא עקב השימוש בTCP ברגע שפקטה תאבד במהלך התקשורת מכיוון שDOH משתמש בTCP לפי פרוטוקול זה לאחר RTT 2 הבקשה לפקטה תשלח מחדש לעומת שימוש ב53DO הממומש עם UDP שבמקרה זה לאחר איבוד פקטה הבקשה אליה תשלח מחדש לאחר זמן קבוע כלשהו שככל הנראה יקח יותר מ2 RTT לכן במקרים בהם יש איבוד רב של פקטות DOH יהיה מהיר יותר

חלק ב'

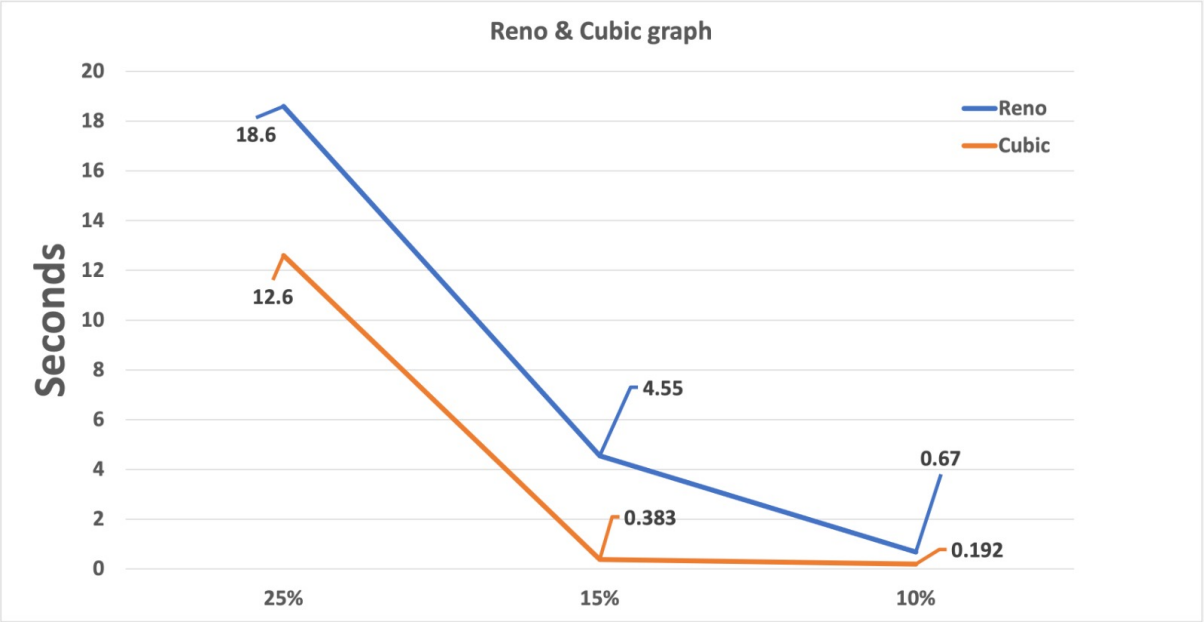
(מצורפים צילומי מסך בעמוד הבא)

אפשר לראות מהטבלה שהאלגוריתם של reno עובד טוב יותר בצורה משמעותית מהאלגוריתם של cubic אנו חושבים שזה קורה מכיוון הדרך שבה כל אחד מהאלגוריתמים מגדיר את גודל החלון שלו, reno מקטין את גודל החלון שלו בחצי כשהוא הוא מקבל 3 שכפולים של acks.
cubic מגדיר את גודל החלון בהתאם לזמן שעבר מהפעם האחרונה שהייתה congestion

טבלת סיכום ממוצע זמני הגעה:

30%	25%	20%	15%	10%	זמן ממוצע \ איבוד פאקטות
timeout	timeout	18.6	4.55	0.675	reno
timeout	timeout	12.6	0.383	0.192	cubic

גרף:



צילומי מסך של הטרמינל של הרצת measure עם אחוזי איבוד פקטות משתנים

10%

```
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ sudo tc qdisc add dev lo root netem loss 10%
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ ./Measure
Current Congestion Control -> Cubic
-----------------------------------------------------------------------------------------
Total receiving time with Cubic: 0.961243 seconds
Average receiving time with Cubic: 0.192249 seconds
-----------------------------------------------------------------------------------------
Current Congestion Control -> Reno
-----------------------------------------------------------------------------------------
Total receiving time with Reno: 3.378535 seconds
Average receiving time with Reno: 0.675707 seconds
netanel@ubuntu:~/CLionProjects/Networking_Ex4$
```

15%

```
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ sudo tc qdisc change dev lo root netem loss 15%
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ ./Measure
Current Congestion Control -> Cubic
-----------------------------------------------------------------------------------------
Total receiving time with Cubic: 1.919046 seconds
Average receiving time with Cubic: 0.383809 seconds
-----------------------------------------------------------------------------------------
Current Congestion Control -> Reno
-----------------------------------------------------------------------------------------
Total receiving time with Reno: 22.766744 seconds
Average receiving time with Reno: 4.553349 seconds
netanel@ubuntu:~/CLionProjects/Networking_Ex4$
```

20%

```
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ sudo tc qdisc change dev lo root netem loss 20%
[sudo] password for netanel:
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ ./m
Current Congestion Control -> Cubic
-------------------------------------------------------------------------------------
Total receiving time with Cubic: 63.160577 seconds
Average receiving time with Cubic: 12.632115 seconds
-------------------------------------------------------------------------------------
Current Congestion Control -> Reno
-------------------------------------------------------------------------------------
Total receiving time with Reno: 93.222909 seconds
Average receiving time with Reno: 18.644582 seconds
netanel@ubuntu:~/CLionProjects/Networking_Ex4$
```

25%

```
netanel@ubuntu:~/CLionProjects/Networking_Ex4$ ./s
Current Congestion Control -> Cubic
-------------------------------------------------------------------------------------
You have successfully connected to the server
You have successfully connected to the server
You have successfully connected to the server
You have successfully connected to the server
You have successfully connected to the server
Total sending time with Cubic: 0.010854 seconds
Average sending time with Cubic: 0.002171 seconds
-------------------------------------------------------------------------------------
Current Congestion Control -> Reno
-------------------------------------------------------------------------------------
ERROR! connection has failed!: Connection timed out
netanel@ubuntu:~/CLionProjects/Networking_Ex4$
```