בס"ד

פרויקט גמר

ת"ז מגישים: 209326776, 322998287

מטרת הפרויקט שלנו היא לבנות מערכת תקשורת בין קליינט לבין מספר שרתים:

1. DHCP
2. DNS local server
3. Application server: אפליקציית HTTP המבצעת redirect לשרת אחר שמשם יורד הקובץ

<u>אופן ההגשה:</u> תקיית ההגשה כוללת שלושה תקיות

לdns&dhcp תקייה אחת הכוללת client מחובר , בתוך תקייה זו נמצאות הקלטות הווירשארק הרלוונטיות.

התקייה של ה tcp ושל הrudp שייכות לחלק השלישי של העבודה וגם בהם נמצאים ההקלטות הרלוונטיות

<u>אופן ההרצה:</u>

נפתח ארבעה טרמינילים מתוך התקייה ונריץ קודם כל את השרתים ולאחר מכן את הלקוח,
הפקודות להרצה בכל אחד מהטרמינילים:

```
sudo python3 DHCP_server.py

sudo python3 DNS_server.py


sudo python3 cleint.py
```

נשים לב שהcleint שלנו בנוי כך שקודם כל ירוץ הDHCP לאחריו ה DNS ולאחריו הHTTP
בלקוח נראה את ההדפסות הבאות המבצעות מעקב על הקוד

```
#####DHCP#####
create DISCOVER packet..
DONE!
.
Sent 1 packets.
The discovery packet has been sent!
create REQUEST packet..
DONE!
.
Sent 1 packets.
The request packet has been sent!
#####DNS#####
b'hello.world.'
```

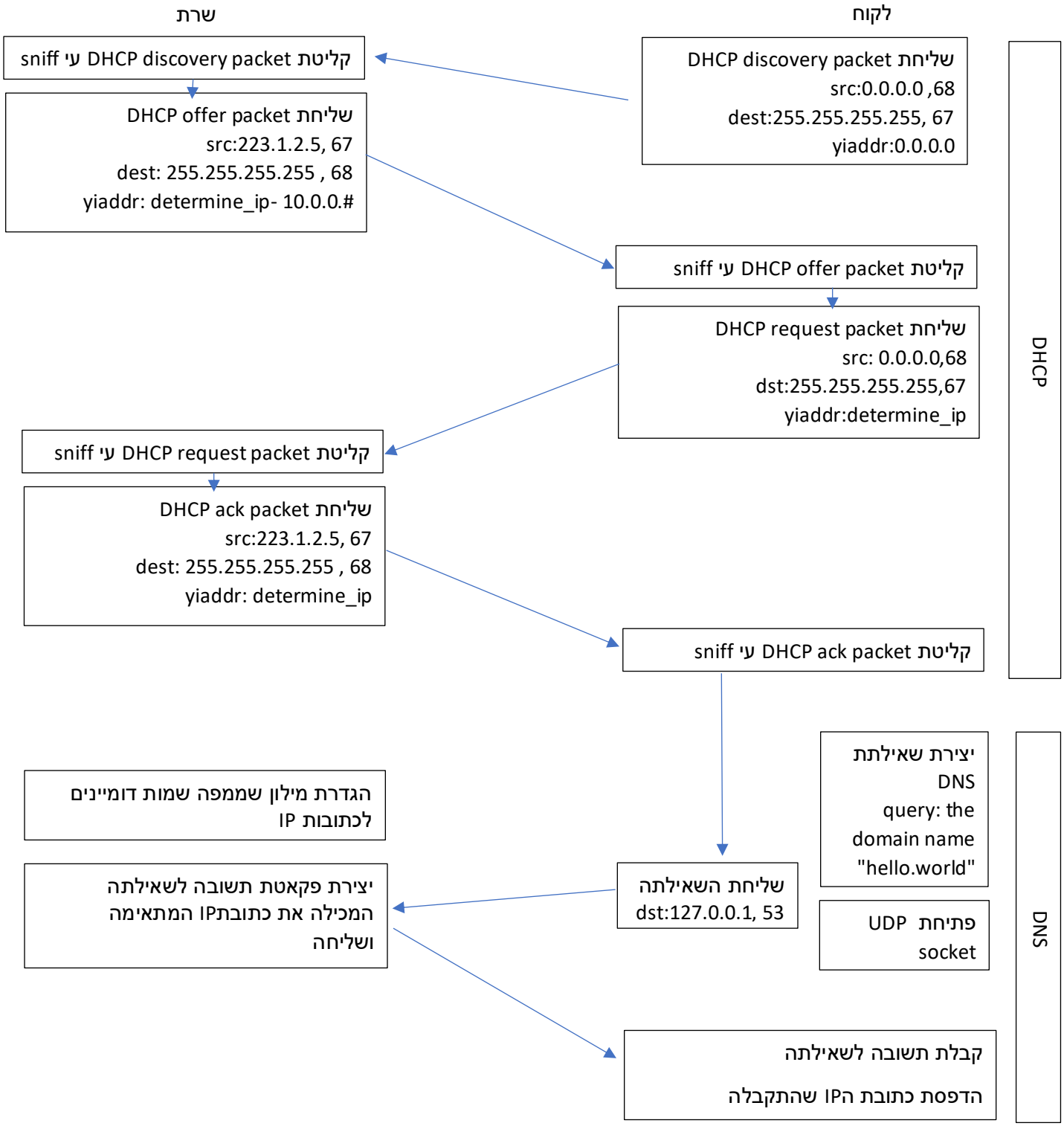
בשרתים נראה את ההדפסות הבאות:

```
renana@renana:~/Desktop/project$ sudo python3 DHCP_server.py
running server
got a packet
it is a DISCOVERY packet!
create OFFER packet..
DONE!
.
Sent 1 packets.
The offer packet has been sent!
got a packet
it is a REQUEST packet!
create ACK packet..
DONE!
.
Sent 1 packets.
The ack packet has been sent!
```

```
renana@renana:~/Desktop/project$ sudo python3 DNS_server.py
running server
handling dns query from client 127.0.0.1 for domain name b'hello.world.'
```

בס"ד

<u>דיאגרמת מצבים:</u>



בס"ד

הסבר על אופן מימוש חלק DNS-2:

server:
בהתחלה מוגדר מילון הממפה את שמות הדומיין אל כתובות הIP,
הגדרנו את המחלקה MyUDPHandler שהיא תת מחלקה של
`socketserver.BaseRequestHandler`
בתוך המחלקה מימשנו את פונקציית הhandle כך שכאשר השרת יקבל שאילתת DNS הפונקציה הזאת תפעל ותטפל בשאילתה.

במקרה שקיבלנו שם דומיין שנמצא במילון נכניס את כתובת הip המתאימה לתוך פאקטת התשובה ונשלח.
במקרה שקיבלנו שם דומיין שלא נמצא במילון נוצרת פאקטת תשובה דיפולטיבית – פאקטה זו תחזור כאשר אין שרת dns שיודע לענות על השאילתה שלנו, (המידע הבינארי שמייצג את הפאקטה הזו הוא מידע אשר הסנפנו באמצעות scapy כשניסנו לגלוש לדומיין לא קיים)

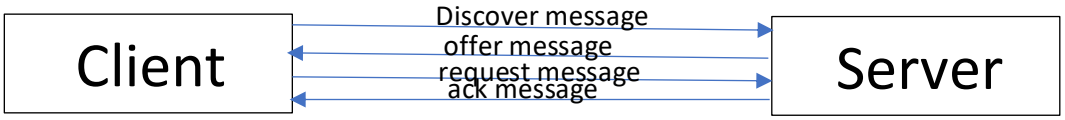
קצת על `socketserver.BaseRequestHandler` :
המודל SocketServer הוא מסגרת בה ניתן ליצור שרתים ,
הRequest handler מקבל בקשה ומטפל בה. מתודת הhandle היא המתודה שבסופו של דבר עושה את העבודה כאשר אצלנו היא אחראית על החזרת פקאטת תשובה המכילה IP address אל הclient.

בס"ד

Wireshark

חלק ראשון- DHCP:

פרוטוקול DHCP הוא דרך לקבל IP address, תהליך זה קורה בארבעה שלבים



נסנן לפי dhcp:

dhcp

No.	Time	Source	Destination	Protocol	Length	Info
5	21.229774623	0.0.0.0	255.255.255.255	DHCP	300	DHCP Discover - Transaction ID 0x87654321
6	21.230326805	10.0.2.2	10.0.2.15	DHCP	592	DHCP Offer - Transaction ID 0x87654321
7	22.254830038	223.1.2.5	255.255.255.255	DHCP	306	DHCP Offer - Transaction ID 0x87654321
8	23.294427196	0.0.0.0	255.255.255.255	DHCP	300	DHCP Request - Transaction ID 0x87654321
9	24.326390812	192.168.1.1	255.255.255.255	DHCP	306	DHCP ACK - Transaction ID 0x87654321

ניתן להבחין בארבעת השלבים (ממורקר בצהוב)

החבילה הראשונה שנשלחת היא חבילת DHCP discovery – זוהי החבילה שנשלחת כדי למצוא שרתי DHCP ולבקש מהם הצעה עם IP address .

```
▸ Frame 5: 300 bytes on wire (2400 bits), 300 bytes captured (2400 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
▸ User Datagram Protocol, Src Port: 68, Dst Port: 67
▾ Dynamic Host Configuration Protocol (Discover)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x87654321
    Seconds elapsed: 0
  ▸ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: 31:36:3a:31:30:3a (31:36:3a:31:30:3a)
    Client hardware address padding: 38373a36343a63323a31
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▾ Option: (53) DHCP Message Type (Discover)
      Length: 1
      DHCP: Discover (1)
  ▸ Option: (51) IP Address Lease Time
  ▸ Option: (1) Subnet Mask (255.255.255.0)
  ▸ Option: (255) End
```

בשכבה השלישית יש שימוש בפרוטוקול IP, הsrc address היא הכתובת 0.0.0.0, שמציינת כי החבילה נשלחה מה client והיות ואין לו עדיין כתובת IP הוא משתמש בכתובת הזו.

הdst address היא הכתובת 255.255.255.255 המציינת כי החבילה נשלחת בbroadcast אל כל הישויות ברשת וזאת מכיוון שהלקוח מנסה להגיע אל כל הDHCP servers והוא לא יודע את הכתובות שלהם.

בשכבה הרביעית יש שימוש בפרוטוקול UDP עם הportים 67,68 במקרה של DHCP הסיבה להעדפת UDP על פני TCP היא שאנו רוצים לשלוח הודעה בbroadcast מה שלא אפשרי בפרוטוקול TCP.

בשכבה החמישית זהו פרוטוקול DHCP אשר מזוהה כאן כסוג discovery .

החבילה השנייה שנשלחת היא DHCP offer – חבילת ההצעה של השרת ,

נשים לב שאצלנו התקבלו שתי חבילות מסוג DHCP offer מצב זה אפשרי כיוון שאם יש שני שרתי DHCP ברשת שניהם יכולים לענות ללקוח שמבקש כתובת IP עם הצעה.

בס"ד

נסתכל על החבילה הראשונה מביניהם:

```
▸ Frame 6: 592 bytes on wire (4736 bits), 592 bytes captured (4736 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 10.0.2.2, Dst: 10.0.2.15
▸ User Datagram Protocol, Src Port: 67, Dst Port: 68
▾ Dynamic Host Configuration Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x87654321
    Seconds elapsed: 0
  ▸ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 10.0.2.15
    Next server IP address: 10.0.2.4
    Relay agent IP address: 0.0.0.0
    Client MAC address: 31:36:3a:31:30:3a (31:36:3a:31:30:3a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name: Ubuntu 22.04.1 .pxe
    Magic cookie: DHCP
  ▾ Option: (53) DHCP Message Type (Offer)
      Length: 1
      DHCP: Offer (2)
  ▾ Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  ▸ Option: (3) Router
  ▸ Option: (6) Domain Name Server
  ▸ Option: (15) Domain Name
  ▾ Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (86400s) 1 day
  ▸ Option: (54) DHCP Server Identifier (10.0.2.2)
  ▸ Option: (255) End
    Padding: 000000000000000000000000000000000000000000000000000000000000000000000000…
```

נשים לב כי בsrc address 10.0.2.2 הוא כתובת הIP של השרת הDHCP השולח את ההצעה,
הdst address נמצאת כתובת הIP אשר הלקוח ביקש (קורה כאן כי כנראה הלקוח ביקש במופרש את הכתובת הזאת)
בשכבת הDHCP במסגרת הצהובה נמצאת הכתובת שהשרת מציע ללקוח 10.0.2.15
בנוסף לכתובת הIP השרת שולח פרטים נוספים:
שדה הlease time שהוא הזמן המובטח ללקוח שכתובת הIP שהוצעה לו תהיה מוקצת עבורו ולא תוקצה לאף אחד אחר, אצלנו הזמן הוא יום אחד, לאחר יום הלקוח לא יוכל להמשיך להשתמש בכתובת הזאת כיוון שיכול להיות שהיא הוקצתה לאחרים. הוא יוכל להמשיך להשתמש בכתובת הזאת אם הוא חידש את הזמן.
בנוסף נשלח הsubnet mask וזאת מכיוון שהמחשב צריך להכיר לא רק את כתובת הIP שלו אלא גם את מזהה הרשת שלו ובשביל זה הוא צריך להכיר את הsubnet mask שלו ולכן זה נשלח בתוך הודעת הDHCP.

נסתכל על חבילת הoffer השנייה:

בס"ד

```
▸ Frame 7: 306 bytes on wire (2448 bits), 306 bytes captured (2448 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 223.1.2.5, Dst: 255.255.255.255
▸ User Datagram Protocol, Src Port: 67, Dst Port: 68
▾ Dynamic Host Configuration Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x87654321
    Seconds elapsed: 0
  ▸ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 10.0.0.1
    Next server IP address: 223.1.2.5
    Relay agent IP address: 0.0.0.0
    Client MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▸ Option: (53) DHCP Message Type (Offer)
  ▾ Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (3600s) 1 hour
  ▾ Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  ▸ Option: (3) Router
  ▸ Option: (255) End
```

בsrc address 223.1.2.5 הוא כתובת הIP של השרת הDHCP השולח את ההצעה,
הdst address נמצאת כתובת הbroadcast 255.255.255.255 וזאת מהסיבה כי למחשב אין עדיין כתובת IP שאמורה לקבל את הבקשה.
בשכבת הDHCP במסגרת הצהובה נמצאת הכתובת שהשרת מציע ללקוח 10.0.0.1
בנוסף לכתובת הIP השרת שולח פרטים נוספים:
שדה הlease time כאן הזמן הוא שעה אחת, בנוסף גם כאן נשלח הsubnet mask .

הלקוח קיבל שתי הצעות שונות לכתובות IP כעת עליו לבחור אחת מהן ואז לשלוח בקשה לקבל באמת את הכתובת הזאת, וכאן אנו מגיעים אל החבילה השלישית.

החבילה השלישית שנשלחת היא DHCP request- חבילת הבקשה בה הלקוח מציין מאיזה משרתי הDHCP הוא רוצה לקבל את כתובת הIP (ושאר הפרטים שהשרת הציע לו).

בס"ד

```
▸ Frame 8: 300 bytes on wire (2400 bits), 300 bytes captured (2400 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
▸ User Datagram Protocol, Src Port: 68, Dst Port: 67
▾ Dynamic Host Configuration Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x87654321
    Seconds elapsed: 0
  ▸ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 10.0.0.1
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: 31:36:3a:31:30:3a (31:36:3a:31:30:3a)
    Client hardware address padding: 38373a36343a63323a31
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▾ Option: (53) DHCP Message Type (Request)
      Length: 1
      DHCP: Request (3)
  ▾ Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (3600s) 1 hour
  ▾ Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  ▾ Option: (255) End
      Option End: 255
```

הdst address נמצאת כתובת הbroadcast 255.255.255.255 וזאת מהסיבה שגם שאר השרתים ידעו שהלקוח בחר בשרת הספציפי הזה וידעו לא לשמור עבורו את הכתובת,
למשל במקרה שלנו ניתן לראות כי הלקוח בחר בכתובת (מסגרת אדומה) 10.0.0.1 שזוהי כתובת הIP שהציע שרת הDHCP השני, אזי במקרה הזה הלקוח שולח הודעת DHCP request לכולם וכך שרת הDHCP הראשון שהציע ללקוח כתובת IP מקבל גם הוא את הDHCP request ומבין כי הלקוח לא בחר בו.
ניתן לראות כי הפרטים הכלולים בחבילת הoffer השניה כמו הlease time , subnet mask כלולים גם הם כאן ונשלחים גם הם בתוך חבילת הDHCP request.

החבילה הרביעית והאחרונה שנשלחת היא חבילת הDHCP ack שמטרתה לציין שהשרת קיבל את הבקשה של הלקוח וכעת הוא יכול להשתמש בכתובת הIP שהוקצתה עבורו.

```
▸ Frame 9: 306 bytes on wire (2448 bits), 306 bytes captured (2448 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 192.168.1.1, Dst: 255.255.255.255
▸ User Datagram Protocol, Src Port: 67, Dst Port: 68
▾ Dynamic Host Configuration Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x87654321
    Seconds elapsed: 0
  ▸ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 10.0.0.1
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▾ Option: (53) DHCP Message Type (ACK)
      Length: 1
      DHCP: ACK (5)
  ▾ Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (3600s) 1 hour
  ▾ Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  ▾ Option: (3) Router
      Length: 4
      Router: 10.0.0.18
  ▾ Option: (255) End
      Option End: 255
```

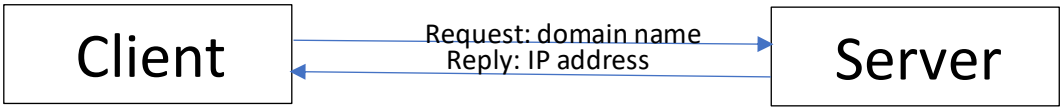
ניתן לראות כי השרת חוזר על הפרטים שהוא נתן ללקוח.
כעת השרת יכול להשתמש בכתובת שנתנה לו עי שרת הDHCP.

בס"ד

חלק שני- DNS local server:

פרוטוקול DNS הוא פרוטוקול הממפה (מתרגם) שמות דומיין אל כתובות IP .

פרוטוקול הDNS פועל באמצעות שאילתה ותשובה.



חבילה ראשונה- חבילת הquery:

```
▸ Frame 10: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 37612, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x0000
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ hello.world: type A, class IN
        Name: hello.world
        [Name Length: 11]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

ניתן לראות כי פרוטוקול הDNS נשלח בשכבה החמישית מעל פרוטוקול הUDP הנמצא בשכבה הרביעית.

שדה הtransaction id מכיל את המזהה של השאילתה הנוכחית שאצלנו הוא 0x0000 כך נוכל לראות שהתשובה שנקבל מהשרת שייכת לשאילתה זאת.

השדה flags מציין כי כאן מדובר בשאילתה סטנדרטית.

השדות תחת הריבוע הירוק הם שדות המתארות את הרשומות שמכילה חבילת DNS , חבילת השאלה שלנו מכילה רשומה של שאלה אחת ללא רשומות נוספות.

בשדה Name ניתן לראות את השם הדומיין המלא, אצלנו השאלה היא עבור hello.world

השדה Type הוא הסוג הרשומה שעליה שואלים, אצלנו סוג A הוא רשומה הממפה בין שם דומיין לבין כתובות IP.

השדה Class הוא שדה המציין את סוג הרשת – IN .

חבילה שניה – חבילת הresponse:

```
▸ Frame 11: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 53, Dst Port: 37612
▾ Domain Name System (response)
    Transaction ID: 0x0000
  ▸ Flags: 0x8100 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ hello.world: type A, class IN
        Name: hello.world
        [Name Length: 11]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▾ Answers
    ▸ hello.world: type A, class IN, addr 10.0.0.17
    [Request In: 10]
    [Time: 0.002479339 seconds]
```

השדה transaction id שערכו 0 באופן זהה לחבילת הquery וכך אנו יודעים כי חבילת התשובה הזאת היא החבילה השייכת לחבילת הquery שלנו.
מהשדה flags ניתן לראות כי התשובה חזרה ללא שגיאות.
השדות תחת הריבוע הירוק מסמנות את הרשומות שיש לנו – רשומה של שאילתה אחת ורשומה של תשובה אחת לשאילתה.
תחת הריבוע האדום אנו רואים את רשומת השאילתה שלנות וזאת מהסיבה שבחבילות התשובה שרתי הDNS משכפלים את השאילתה שנשלחה אליהם ושולחים אותה חזרה אל השולח.
בריבוע הסגול נמצאת רשומת התשובה

```
Answers
  hello.world: type A, class IN, addr 10.0.0.17
    Name: hello.world
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 0 (0 seconds)
    Data length: 4
    Address: 10.0.0.17
```

השדה Name הוא שם הדומיין hello.world לו מיועדת התשובה.
השדה Type הוא סוג הרשומה- סוג A .
השדה Class הוא סוג הרשת IN
השדה ttl מציין כמה זמן לשמור את הרשומה
השדה Data Length מציין אורך המידע – 4, וזאת מכיוון שהמידע שלנו הוא כתובת IP שהיא באורך ארבעה בתים.
השדה Address הוא הData של השאילתה שאצלנו הוא כתובת הIP 10.0.0.17 השייכת לשם הדומיין hello.world עליו השאילתה נשלחה.

בס"ד

חלק שלישי- Application server:

בחלק הזה יש שני חלקים: חלק ראשון של RUDP וחלק שני של tcp

נתחיל בחלק של RUDP יש לנו שלושה קבצים: client server proxy ככה שה client and proxy מדברים ביניהם ב http והproxy עושה את redirect לסרבר , הלוקח מדבר עם הסרבר בrudp ומוריד ממנו את הקובץ

כרגע הקבצים מכוונים על עבודה עם מכונה אחת וירוצו כמו שצריך. אנחנו עשינו עם 2 מכונות כדי להראות תעבורה נכונה בוויראשק וגם לנסות עם IP שונים. כדי שירוץ על 2 מכונות צריך לעשות:

נקדים קודם שפתחנו עוד מכונה וירטואלית בעלת IP שונה

(הבודק צריך לשים לב שהוא משנה את בהתאם למה שיש אצלו במכונות,

בקובץ של הלקוח בקבועים במשתנה של

```
proxy_server
```

צריך להכניס את IPשל הproxy וגם בקובץ של הproxy בקבועים במשתנה

```
server_address
```

צריך להכניס את IP של השרת)

, ככה שיש לנו 2 מכונות (עשינו זאת כדי שיראו טוב בוויראשק את התקשורת כי ככה הIP שונים)

הראשונה יושבת על IP 10.0.2.15 בה נריץ את server and proxy

והשנייה יושבת על 10.0.2.4 בה נריץ את הclient

בס"ד

הסבר על תהליך הורדת קובץ RUDP:

בשלב זה נממש את תהליך ההעברה ע"י Reliable UDP כלומר UDP אמין המדמה חיבור TCP

בחלק זה החיבור הנעשה עם שרת הפרוקסי הינו עדיין בתור חיבור TCP וכאשר אנחנו מקבלים את הכתובת של השרת אליו הוא מפנה אותנו אנחנו יוצרים איתו תקשרות להעברת הקובץ באמצעות פרוטוקול UDP כאשר לפרוטוקול זה הוספנו אפשרויות מסוימות על מנת ליצור אותו אמין

צד השרת –

מצד השרת אנו קוראים את הקובץ ומחלקים אותו לפאקטות בגודל שקבענו מראש עבור כל פאקטה נצמיד אליה מספר סידורי שרץ באופן סדרתי מהמספר 1 עד מספר הפאקטות אשר חילקנו אליו את הקובץ , בנוסף עבור כל פאקטה או מצמיד גם את ה checksum של אותה פאקטה כך שהלקוח יוכל לוודא כי הפאקטה הגיע בשלמות וללא שינויים או נזק בדרך .

בשלב זה השרת מכניס כמות מסויים קבוע מראש של פאקטות לחלון השליחה ומשם שולח ללקוח ומחכה לקבל אישור כי הלקוח קיבל את החבילה כאשר הוא מקבל אישור הוא מוחק את הפאקטה מהחלון ומחכה לקבל את כל האישורים עבור אותו חלון.

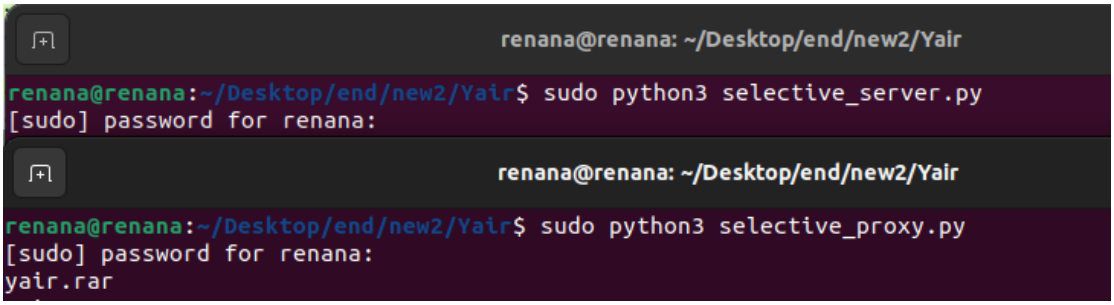
כאשר הוא אינו מקבל אישור הוא שולח את החבילה שוב.

צד הלקוח-

מצד הלקוח עבור כל פאקטה שנשלחת אנחנו נעשה בדיקה לבדוק שה checksum שנשלח באמת תואם ל checksum של הפאקטה ואם כן נשמור את הפאקטה עם המספר הסידורי במילון כאשר המספר הסידורי הוא המפתח ונשלח Ack לסרבר (אישור על קבלת החבילה) אשר מכיל את מספר החבילה שהתקבלה , כאשר הלקוח מקבל חבילה שקטנה מגודל הפאקטה שנקבע מראש אנו יודעים כי זוהי החבילה האחרונה בקובץ ובעצם נשלח לשרת פקודת סיום ונכתוב את הקובץ החדש.

נתחיל בחלק ראשון בהדגמת ריצה והקלטות מהווירשאק

נפתח את 2 טרמינלים במכונה הראשונה ונריץ



בס"ד

נפתח את הטרמינל במכונה השנייה ונריץ

```
renana@renana: ~/Desktop/new2/Yair
renana@renana:~/Desktop/new2/Yair$ sudo python3 selective_client.py
[sudo] password for renana:
Enter file name: yair.rar
yair.rar
10.0.2.15:30776/yair.rar
yair.rar
('10.0.2.15', 30776)
File path: yair.rar
Server address: ('10.0.2.15', 30776)
Error: the server could not open the file
renana@renana:~/Desktop/new2/Yair$ sudo python3 selective_client.py
Enter file name: Yair.rar
Yair.rar
10.0.2.15:30776/Yair.rar
Yair.rar
('10.0.2.15', 30776)
File path: Yair.rar
Server address: ('10.0.2.15', 30776)
renana@renana:~/Desktop/new2/Yair$
```

אפשר לראות שבפעם הראשונה אצל השרת לא היה קיים קובץ כזה לכן הוא לא שלח אולם בפעם השנייה כתבתי כמו שצריך וזה מצא ושלח (שמתי קובץ אחד בשם yair.txt כדי להראות שהוא מצליח לשלוח קובץ גדול, וקובץ שני בשם renana.txt לבדיקה שלכם כדי לבדוק עם איבוד פאקטות)

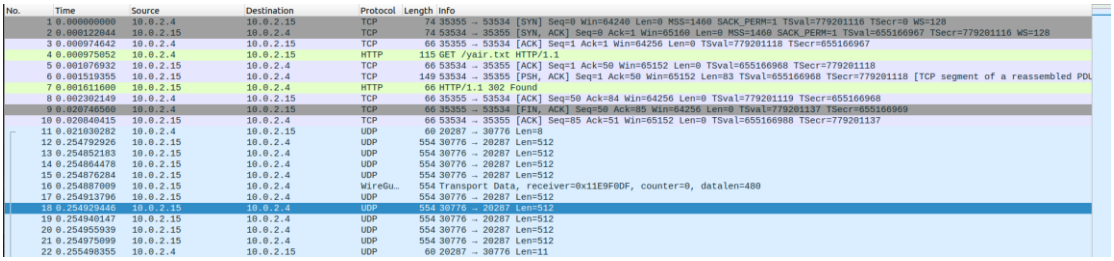
```
renana@renana: ~/Desktop/end/new2/Yair
renana@renana:~/Desktop/end/new2/Yair$ sudo python3 selective_server.py
[sudo] password for renana:
Received request for file: yair.rar
Could not open file
^[[A^[[A^[[B^[[AReceived request for file: Yair.rar
Sent packet 0
Sent packet 1
Sent packet 2
Sent packet 3
Sent packet 4
Sent packet 5
Sent packet 6
Sent packet 7
Sent packet 8
Sent packet 9
Received ACK for packet 0
Received ACK for packet 1
Received ACK for packet 2
Received ACK for packet 3
Received ACK for packet 4
Received ACK for packet 5
Received ACK for packet 6
Received ACK for packet 7
Received ACK for packet 8
Received ACK for packet 9
Sent packet 9
file sent successfully
```

```
renana@renana: ~/Desktop/end/...
renana@renana:~/Desktop/end/new2/Yair$ sudo python3 select
ive_proxy.py
[sudo] password for renana:
yair.rar
Yair.rar
SS
```

בס"ד

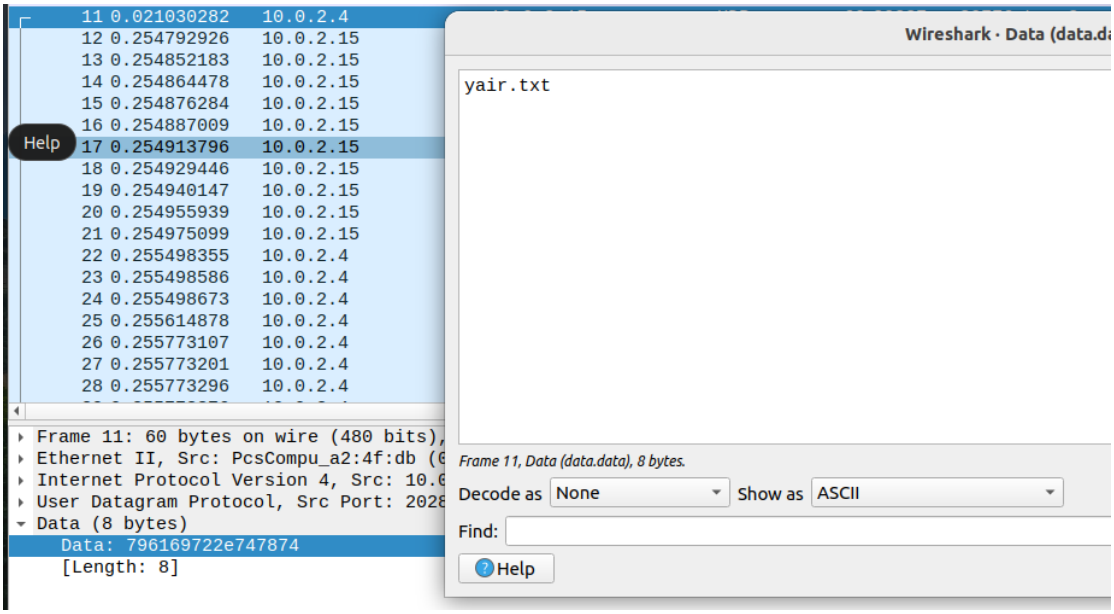
אפשר לראות שזה נשלח בפאקטות ועל כל פאקטה קיבלנו ACK כלומר זה נשלח כנדרש

נסתכל בווירשאק:



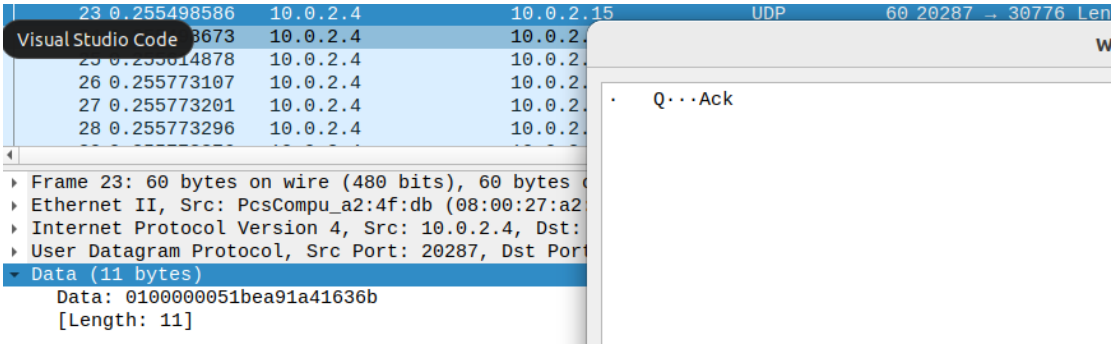
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.4	10.0.2.15	TCP	74	35355 → 53534 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=779201116 TSecr=0 WS=128
2	0.000122944	10.0.2.15	10.0.2.4	TCP	74	53534 → 35355 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=655166967 TSecr=779201116 WS=128
3	0.000974642	10.0.2.4	10.0.2.15	TCP	66	35355 → 53534 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=779201118 TSecr=655166967
4	0.000975052	10.0.2.4	10.0.2.15	HTTP	115	GET /yair.txt HTTP/1.1
5	0.001076932	10.0.2.15	10.0.2.4	TCP	66	53534 → 35355 [ACK] Seq=1 Ack=50 Win=65152 Len=0 TSval=655166968 TSecr=779201118
6	0.001519355	10.0.2.15	10.0.2.4	TCP	149	53534 → 35355 [PSH, ACK] Seq=1 Ack=50 Win=65152 Len=83 TSval=655166968 TSecr=779201118 [TCP segment of a reassembled PDU
7	0.001611600	10.0.2.15	10.0.2.4	HTTP	66	HTTP/1.1 302 Found
8	0.002302149	10.0.2.4	10.0.2.15	TCP	66	35355 → 53534 [ACK] Seq=50 Ack=84 Win=64256 Len=0 TSval=779201119 TSecr=655166968
9	0.020746560	10.0.2.4	10.0.2.15	TCP	66	35355 → 53534 [FIN, ACK] Seq=50 Ack=85 Win=64256 Len=0 TSval=779201137 TSecr=655166969
10	0.020840415	10.0.2.15	10.0.2.4	TCP	66	53534 → 35355 [ACK] Seq=85 Ack=51 Win=65152 Len=0 TSval=655166988 TSecr=779201137
11	0.021030282	10.0.2.4	10.0.2.15	UDP	60	20287 → 30776 Len=8
12	0.254792926	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
13	0.254852183	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
14	0.254864478	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
15	0.254876284	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
16	0.254887009	10.0.2.15	10.0.2.4	WireGu…	554	Transport Data, receiver=0x11E9F0DF, counter=0, datalen=480
17	0.254913796	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
18	0.254929446	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
19	0.254940147	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
20	0.254955939	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
21	0.254975099	10.0.2.15	10.0.2.4	UDP	554	30776 → 20287 Len=512
22	0.255498355	10.0.2.4	10.0.2.15	UDP	60	20287 → 30776 Len=11

רואים בשורה 4 את בקשת HTTP מה PROXY ובשורה 7 את התגובה שלו עם סטטוס 302 (סטטוס בין 301 ל 399 הם קשורים לredirect) אחכ משורה 11 זה חיבור של הלקוח עם הסרבר בudp נראה את הבקשה שלו בשורה 11(רואים שמבקש קובץ מסוים שנמצא אצל הסרבר)



משורה 12 עד 21 זה פאקטות שהשרת שלח ללקוח ואחכ הלקוח מחזיר ACK על כל פאקטה

לדוג נסתכל בשורה 23 על ה data ונראה שנשלח ה sequence and Ack



בס"ד

איבוד פאקטות 10%

```
renana@renana: ~/Desktop/end/new2/Yair
renana@renana:~/Desktop/end/new2/Yair$ sudo tc qdisc add dev enp0s3 root netem loss 10%
[sudo] password for renana:
renana@renana:~/Desktop/end/new2/Yair$
```

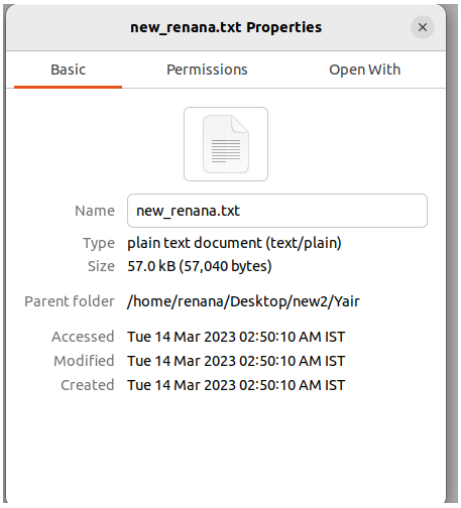
עשינו את אותם פעולות כמו מקודם רק הפעם עם איבוד פאקטות של 10% ומה ששלחנו הגיע ליעד כמו שציפנו בגודל המלא כלומר התגברנו על איבוד הפאקטות

נסתכל על הטרמינל של הסרבר ואם לדוג ACK מסוים לא התקבל ונגמר הזמן אזי הסרבר שולח מאותה חיבלה שלא התקבל ACK והלאה

```
Received ACK for packet 77
Received ACK for packet 78
Received ACK for packet 81
Received ACK for packet 82
Received ACK for packet 83
Timeout, resending packets
Sent packet 79
Sent packet 80
Sent packet 80
Sent packet 81
```

אפשר לראות שלא התקבלה תגובה על חבילה 79 לכן הסרבר שלח שוב מחבילה 79.

כפי שאפשר לראות הקובץ אצל הלקוח התקבל בדיוק באותו הגודל (אצל הלוקח הקובץ נשמר כשם המקורי + new_)



Latency

המערכת שלנו מתמודדת עם בעיית השהייה ככה שאם השרת לא קיבל ACK יותר מ2 שניות הוא שולח את הפאקטה שוב, לכן גם במקרה שהלקוח יקבל את הפאקטה אבל לא החזיר ack בשרת ישלח שוב את אותה פאקטה

בס"ד

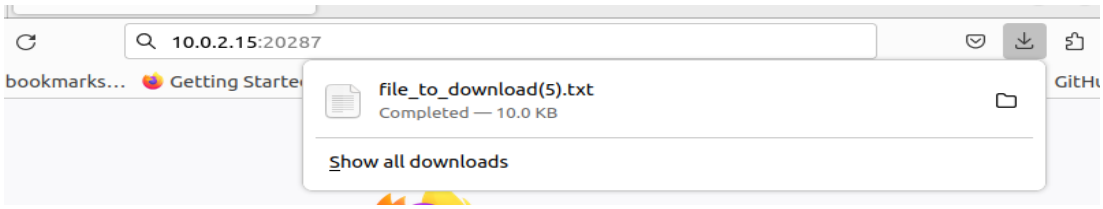
חלק שני באפליקציה הוא הTCP

פתחנו 2 טרמינלים ב10.0.2.15 והרצנו את הפקודות הבאות

```
renana@renana:~/Desktop/end/new2/Yair$ sudo python3 tcp_server.py
```

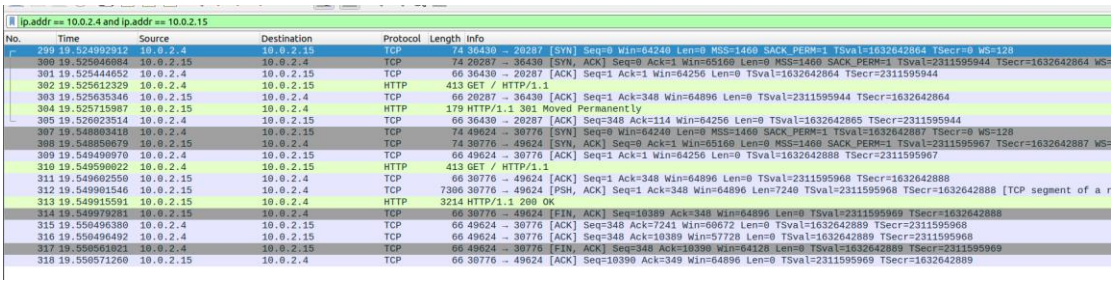
```
renana@renana:~/Desktop/end/new2/Yair$ sudo python3 tcp_redirect_server.py
```

לאחר מכן פתחנו את הדפדפן ב10.0.2.4 ורשמנו בכתובת URL 10.0.2.15:20287



(רואים מה הכנסנו בכתובת url ושזה הוריד את הקובץ מהשרת)

(שוב עשינו זאת כדי ש"הלקוח" יהיה עם IP שונה מזה של הסרבר redirect כדי שנוכל לראות טוב בוויראשק את התעבורה)

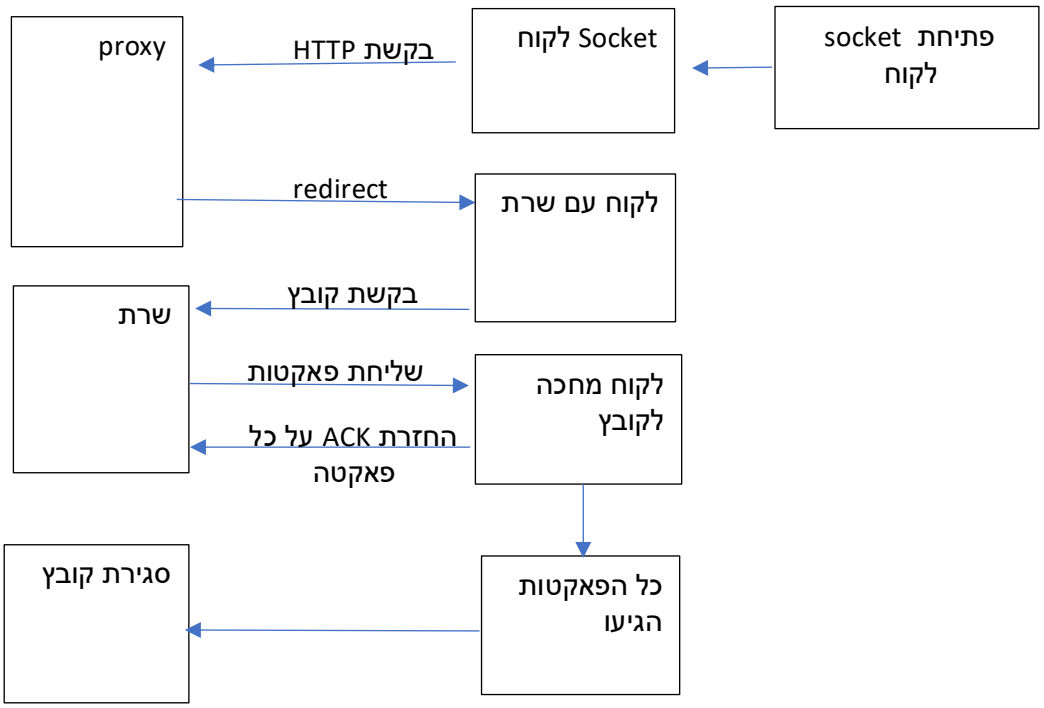


ip.addr == 10.0.2.4 and ip.addr == 10.0.2.15

No.	Time	Source	Destination	Protocol	Length	Info
299	19.524992912	10.0.2.4	10.0.2.15	TCP	74	36430 → 20287 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1632642864 TSecr=0 WS=128
300	19.525046084	10.0.2.15	10.0.2.4	TCP	74	20287 → 36430 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2311595944 TSecr=1632642864 WS=
301	19.525444652	10.0.2.4	10.0.2.15	TCP	66	36430 → 20287 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1632642864 TSecr=2311595944
302	19.525612329	10.0.2.4	10.0.2.15	HTTP	413	GET / HTTP/1.1
303	19.525635346	10.0.2.15	10.0.2.4	TCP	66	20287 → 36430 [ACK] Seq=1 Ack=348 Win=64896 Len=0 TSval=2311595944 TSecr=1632642864
304	19.525715987	10.0.2.15	10.0.2.4	HTTP	179	HTTP/1.1 301 Moved Permanently
305	19.526023514	10.0.2.4	10.0.2.15	TCP	66	36430 → 20287 [ACK] Seq=348 Ack=114 Win=64256 Len=0 TSval=1632642865 TSecr=2311595944
307	19.548803418	10.0.2.4	10.0.2.15	TCP	74	49624 → 30776 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1632642887 TSecr=0 WS=128
308	19.548850679	10.0.2.15	10.0.2.4	TCP	74	30776 → 49624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2311595967 TSecr=1632642887 WS=
309	19.549490970	10.0.2.4	10.0.2.15	TCP	66	49624 → 30776 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1632642888 TSecr=2311595967
310	19.549590022	10.0.2.4	10.0.2.15	HTTP	413	GET / HTTP/1.1
311	19.549602550	10.0.2.15	10.0.2.4	TCP	66	30776 → 49624 [ACK] Seq=1 Ack=348 Win=64896 Len=0 TSval=2311595968 TSecr=1632642888
312	19.549901546	10.0.2.15	10.0.2.4	TCP	7306	30776 → 49624 [PSH, ACK] Seq=1 Ack=348 Win=64896 Len=7240 TSval=2311595968 TSecr=1632642888 [TCP segment of a r
313	19.549915591	10.0.2.15	10.0.2.4	HTTP	3214	HTTP/1.1 200 OK
314	19.549979281	10.0.2.15	10.0.2.4	TCP	66	30776 → 49624 [FIN, ACK] Seq=10389 Ack=348 Win=64896 Len=0 TSval=2311595969 TSecr=1632642888
315	19.550496380	10.0.2.4	10.0.2.15	TCP	66	49624 → 30776 [ACK] Seq=348 Ack=7241 Win=60672 Len=0 TSval=1632642889 TSecr=2311595968
316	19.550496492	10.0.2.4	10.0.2.15	TCP	66	49624 → 30776 [ACK] Seq=348 Ack=10389 Win=57728 Len=0 TSval=1632642889 TSecr=2311595968
317	19.550561021	10.0.2.4	10.0.2.15	TCP	66	49624 → 30776 [FIN, ACK] Seq=348 Ack=10390 Win=64128 Len=0 TSval=1632642889 TSecr=2311595969
318	19.550571260	10.0.2.15	10.0.2.4	TCP	66	30776 → 49624 [ACK] Seq=10390 Ack=349 Win=64896 Len=0 TSval=2311595969 TSecr=1632642889

בס"ד

דיאגרמה :

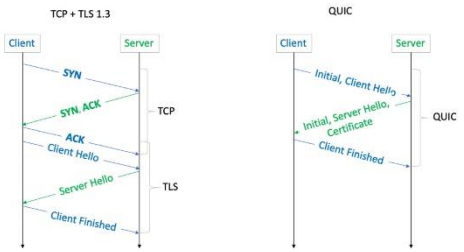

פתיחת socket לקוח
Socket לקוח
בקשת HTTP
proxy
redirect
לקוח עם שרת
בקשת קובץ
שרת
שליחת פאקטות
לקוח מחכה לקובץ
החזרת ACK על כל פאקטה
כל הפאקטות הגיעו
סגירת קובץ

בס"ד

תשובות לשאלות:

1)מנה לפחות ארבע הבדלים עיקריים בין פרוטוקול TCP ל-Quic

1. תהליך החיבור- בפרוטוקול TCP תהליך הקמת החיבור בין שרת ולקוח הוא ע"י לחיצת ידיים משולשת, לעומת זאת פרוטוקול QUIC בנוי מעל שכבת הUDP ומסיבה זאת נצרכת רק פאקטה אחת לצורך הקמת קשר.
ניתן לראות זאת באיור->



2. הצפנה – כאשר אנו משתמשים בפרוטוקול TCP לצורך הצפנה נשתמש בפרוטוקול כמו TLS - Transport layer security , לעומת זאת פרוטוקול QUIC ממזג יחד את לחיצת הידיים המשולשת ואת ההצפנה שבTCP קורת ע"י הTLS.
3. פרוטוקול TCP משתמש באלגוריתם CC כדי להתמודד עם אובדן פאקטות, לעומת זאת בפרוטוקול QUIC קיים מנגנון המאפשר לשלוט בכמות הנתונים המקסימלית שניתן לשלוח בסטרים אחד.
4. Multiplexing- החשיבות של מולטיפלקסינג זה מניעת חסימת HOL (חסימת HOL היא מצב בו כאשר אנו מבקשים כמה אובייקטים ויש אובייקט גדול שהתעכב וזה גורם לאובייקט קטן שבא אחריו להתקע).
בפרוטוקול TCP ניתן להשתמש רק בסטרים בודד במהלך חיבור,
לעומת זאת בפרוטוקול QUIC ניתן להשתמש במספר זרמים כך שבמקרה שבו יש לנו פאקטה אבודה בזרם מסויים זה ישפיע רק על הזרם הספציפי הזה- מקטין באופן משמעותי את חסימות HOL.

2)מנה לפחות שני הבדלים עיקריים בין Cubic ל-Vegas

	Vegas	Cubic
Conegetion control	שולט ב conegestion window ע"י כך שהוא מודד את זמני הנסיעה של החבילה	משתמש בפונקציית Cubic וכך הוא מתאים את קצב השליחה
מטרה	שמירה על רמת גודש יציבה	השגת תפוקה גבוהה ברשת (הגדלת קצב השליחה במהירות עד שהיא מגיעה למצב של אובדן פאקטות)
תגובה לשינויים	תגובה מהירה כיוון שהמדידה של העיכוב נעשת באופן נפרד עבור כל חבילה וכאשר יש עומס החלון קטן	תגובה איטית יותר, התגובה של cubic היא לפי חישוב אובדן הפאקטות. כאשר יש עומס החלון קטן באטיות ובהדרגה.

3)הסבר מהו פרוטוקול BGP ,במה הוא שונה מ-OSPF והאם הוא עובד על פי מסלולים קצרים

פרוטוקול BGP או בשמו המלא Broder Gateway Protocol הוא פרוטוקול ניתוב.
בפרוטוקול זה הרשת אומרת שהיא קיימת למי היא יכולה להגיע ואיך.
BGP כולל שתי תתי פרוטוקולים:
eBGP: BGP חיצוני – פרוטוקול ניתוב בין שני נתבי BGP ששייכים למערכת אוטונומית שונה והם שכנים, כדי למצוא את המסלול האופטימלי ברשת הפרוטוקול מעדכן את המערכות האוטונומיות על איזה כתובות IP כל נתב אחראי כדי שהם ינתבו אליו מידע רלוונטי.
iBGP: BGP פנימי – פרוטוקול ניתוב בין שני נתבי BGP בתוך אותה מערכת אוטונומית,

מטרתו של הפרוטוקול הוא לספק מידע לנתבים פנימיים ברשת (כגון רשימת כתובות IP הנמצאות מחוץ למערכת אוטונומית)

לכל מערכת אוטונומית מוקצה מספר מזהה ייחודי ASN, כדי לבנות מסלולים כל מערכת אוטונומית מפרסמת את הASN ורשימה של תחומי הכתובות שלו שאליהם הוא יכול להעביר חבילות. כל שכן מקבל את ההודעה שומר את המידע אצלו, משרשר את הASN של עצמו להודעה ומעביר את ההודעה המשורשרת הלאה לשכנים שלו.
השכנים החדשים מקבלים רשימת ASN ותחומי כתובות שעליהם לעבור כדי להגיע לאותה קבוצת כתובות. בעיה שיכולה להיווצר היא מעגל, כדי לפתור זאת כאשר מערכת אוטונומית מקבלת הודעת BGP שמכילה את הASN שלה עצמה היא מפילה אותה.

בחירת המסלול האופטימלי נעשת עבור כל מערכת אוטונומית בנפרד, המסלול האופטימלי הוא זה שיפורסם לשכנים. בחירת המסלול מתחשבת במספר גורמים כמו כמות המערכות האוטונומיות בדרך ליעד מסוים או מהירות התקשורת בין שכנים שונים. כל מערכת אוטונומית קובעת לעצמה BGP Policy שגם משפיע על בחירת המסלול.
BGP יכול לעבוד עפ"י מסלולים קצרים, אך לא תמיד, הוא יכול להתחשב גם בגורמים נוספיעץ

BGP וOBSF הם שניהם פרטוקולי ניתוב:
OBSF הוא פרוטוקול ניתוב פנימי בתוך המערכת האוטונומית, BGP הוא פרוטוקול ניתוב חיצוני בין מערכות אוטונומיות שונות.
בחירת המסלול האופטימלי בBGP נעשית עפ"י BGP Policy בOBSF זה נעשה ע"י בחירת הנתיב בו עלות התעבורה היא הזולה ביותר.

4)<u>בהינתן הקוד שפיתחתם בפרויקט זה, אנא הוסיפו את הנתונים לטבלה הזו על בסיס תהליך</u>

<u>ההודעות של הפרויקט שלכם. הסבירו איך ההודעות ישתנו אם יהיה NAT בין המשתמש</u>

<u>לשרתים והאם תשתמשו בפרוטוקול QUIC</u>

Application	Port Src	Port Dst	IP Src	IP Des	Mac Src	Mac Des
DHCP	68	67	0.0.0.0	223.1.2.5	16:10:87:64:c2:1f	ff:ff:ff:ff:ff:ff
DNS	נקבע בזמן ריצה כאשר יוצרים את הsocket	53		127.0.0.1		
TCP	36430	20287	10.0.2.4	10.0.2.15	08:00:27:a2:4f:db	08:00:27:7b:91:d6
RUDP client&proxy	20287	53534	10.0.2.4	10.0.2.15	08:00:27:a2:4f:db	08:00:27:7b:91:d6
RUDP Client&server	20287	30776	10.0.2.4	10.0.2.15	08:00:27:a2:4f:db	08:00:27:7b:91:d6

5)הסבירו את ההבדלים בין פרוטוקול ARP ל-DNS

	DNS	ARP
מטרה	להמיר שם דומיין לכתובת IP אליהם הם יפנו בזמן תקשורת כך שבני אדם יוכלו לפנות לכתובות אינטרנט בקלות	לאתר כתובות MAC עפ"י כתובות IP כאשר מחשב רוצה לשלוח מידע אל מחשב אחר באותו רשת מקומית.
שכבה	DNS נמצא בשכבת האפליקציה	ARP נמצא בשכבת הLink
מה ממפה	שמות דומיין -> כתובות IP	כתובות IP -> כתובות MAC (כתובות לוגיות של שכבת הרשת לכתובות פיזיות של שכבת הקו)
Cache	שומר את המיפוי משמות הדומיין אל כתובות הIP	שומר את המיפוי מכתובות הIP אל כתובות הMAC

בס"ד

ביבלוגרפיה:

https://docs.python.org/3/library/socketserver.html
/http://pymotw.com/2/SocketServer
/https://blog.apnic.net/2022/11/03/comparing-tcp-and-quic

http://www.tcpipguide.com/free/t_DHCPMessageFormat.htm

/https://netbeez.net/blog/tcp-vs-quic-new-transport-protocol

https://medium.com/codavel-blog/quic-vs-tcp-tls-and-why-quic-is-not-the-next-big-thing-d4ef59143efd
https://en.wikipedia.org/wiki/Border_Gateway_Protocol
https://he.wikipedia.org/wiki/BGP#iBGP_(internal_BGP)
https://he.wikipedia.org/wiki/Open_Shortest_Path_First
https://en.wikipedia.org/wiki/Address_Resolution_Protocolhttps://en.wikipedia.org/wiki/Address_Resolution_Protocol
https://he.wikipedia.org/wiki/Domain_Name_System
https://he.wikipedia.org/wiki/Address_Resolution_Protocol
https://stackoverflow.com/questions/50026438/crafting-a-dhcp-offer-packet-in-scapy

https://www.thepythoncode.com/article/dhcp-listener-using-scapy-in-python

https://www.ietf.org/staging/draft-ietf-tcpm-rfc8312bis-03.html#section-4.1.1-1

https://en.wikipedia.org/wiki/TCP_Vegas

/https://www.geeksforgeeks.org/basic-concept-of-tcp-vegas

https://cseweb.ucsd.edu/classes/wi01/cse222/papers/brakmo-vegas-jsac95.pdf

ספר רשתות מחשבים של עומר רוזנבוים ברק גונן שלומי הוד

ספר הקורס

הרצאות+ מצגות מההרצאות