

**מעבדה מספר 1**

**TCP**

1. **הכנות:**
2. על מנת לבצע את המעבדה יש להפעיל את המכונות הוירטואליות של kali1,kali2.
3. יש לבדוק תקינות התקשורת בין התחנות, למחשב המארח (המחשב עצמו- פיסית המחשב עליהם אתם יושבים) ולרשת האינטרנט.

במידת הצורך יש לשנות את כתובת הip של המכונות הוירטואליות (eth0) כך שיהיו על אותה רשת כמו הכרטיס רשת הווירטואלי שקיים על המארח.



Virtualbox host kali1

192.168.56.1 192.168.56.100

1. את הwireshark נריץ ממכונת ה-kali
2. תחילת הסנפה:
3. פתח את הwireshark והתחל להסניף מהכרטיס רשת הפעיל.
4. וודא שאכן אתה רואה מידע שנשלח על הכרטיס.

(על מנת לראות תעבורה יש לייצר אותה, פתח דפדפן והיכנס לאתר כלשהו).

1. **ידע מקדים נחוץ למעבדה:**

**מודל שבע השכבות (ISO-OSI)**

המודל מגדיר את השכבות (רמות) במערכת תקשורת שלמה:

שכבה 1 – השכבה הפיזית – the physical layer:

אחראית על החיבור הפיזי של הקשר, העברה וקבלה של המידע הגולמי והמרת ביטים אנאלוגיים או דיגיטליים לאותות חשמליים או אופטיים.

שכבה 2 – data link layer:

מסנכרנת את המידע העובר על גבי השכבה הפיסית. דואגת לניהול רצף סינכרוני ולוגי של המידע בין שני מקורות ברשת.

שכבה 3 – שכבת הרשת – network layer:

מנווטת את המידע בין שני מקורות ברשת, דרך הערוצים השונים ברשת. אחראית על חלוקה לחבילות של ההודעות היוצאות ואיחוי של החבילות הנכנסות לכדי הודעות, עבור השכבות הגבוהות יותר.

שכבה 4 – שכבת התעבורה – transport layer:

מקבלת הודעות מהשכבות מעליה, מחלקת אותן ליחידות קטנות יותר, אותן היא מעבירה לשכבת הרשת. בשכבה זו נקבע האם השליחה תעשה בצורה מקבילית או דרך נתיב אחד. שכבה זו דואגת כי המידע מועבר ומתקבל בשכבת הרשת, ללא טעויות ובדרך היעילה ביותר.

שכבה 5 - Session Layer:

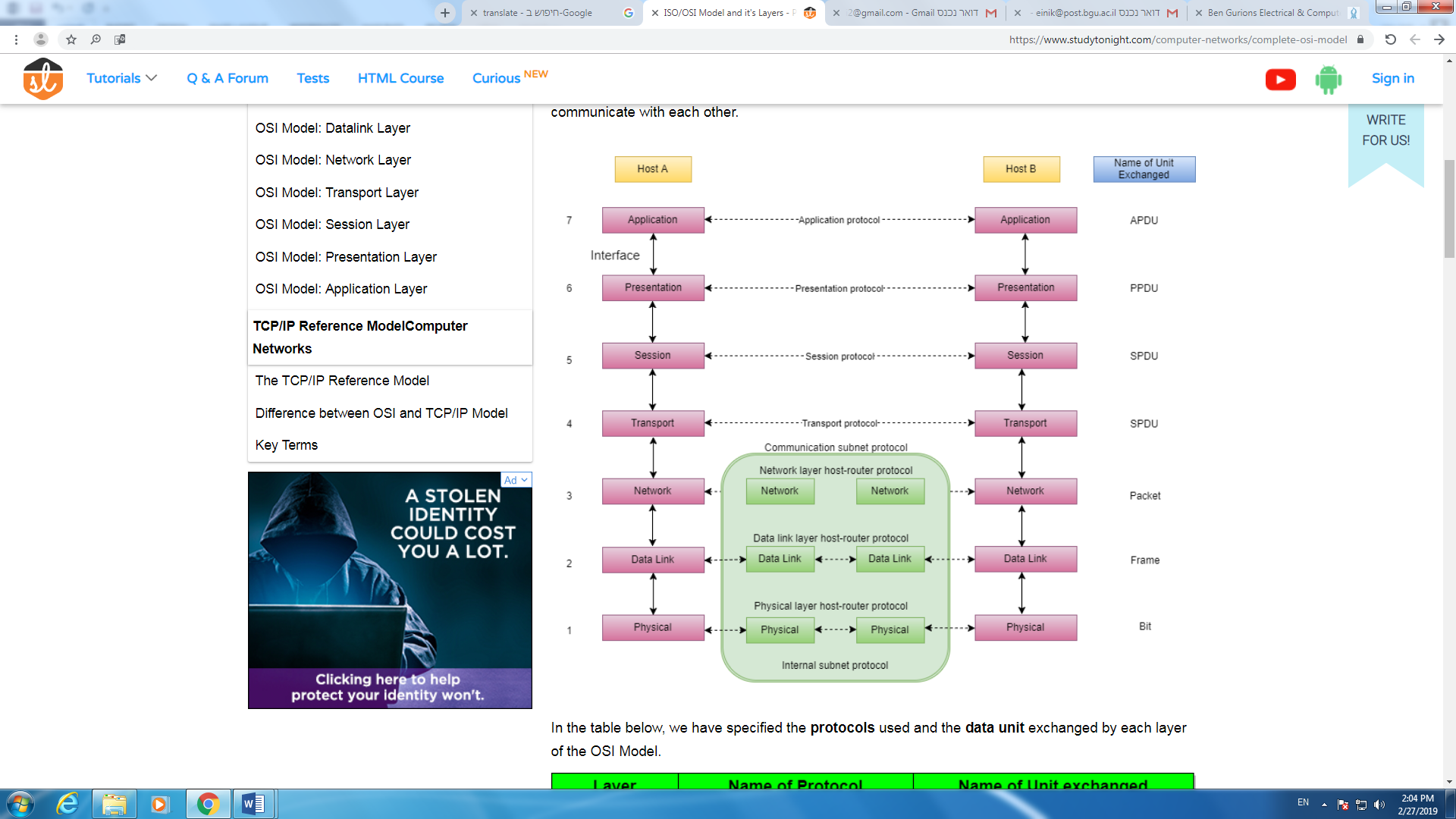
מנהלת ומסנכרנת את העברות המידע בין אפליקציות או מערכות שונות.

שכבה 6 – תצוגה - The Presentation Layer:

דואגת כי הודעות מועברות כך שהמערכת המקבלת תוכל לקרוא ולהשתמש במידע הנשלח. שכבה זו מטפלת בתחביר ובסמנטיקה ומתבצעות בה תרגום והצפנה של המידע המועבר.

שכבה 7 - Application Layer:

השכבה העליונה המוצגת למשתמש. בשכבה זו יש שימוש בפרוטוקולים שונים המאפשרים גישה ותקשורת עם מערכות הרשת עבור המשתמש.



**תקשורת TCP**

1. Transmission control protocol (ר"ת TCP) הוא פרוטוקול תקשורת נתונים הפועל בשכבות התעבורה של מודל ה-OSI ובמודל ה-TCP/IP. TCP מבטיח העברה אמינה של נתונים בין שתי תחנות ברשת מחשבים באמצעות יצירת חיבור מקושר (Connection Oriented), תוך שימוש בפרוטוקול ה-IP לצורך העברת הנתונים. TCP מעביר את הנתונים שהועברו באמצעות IP, מוודא את נכונותם, ומאשר את קבלת הנתונים במלואם או מבקש שליחה מחדש של נתונים שלא הגיעו בצורה תקינה.
2. הקמת קשר – לחיצת יד משולשת Three-way handshake

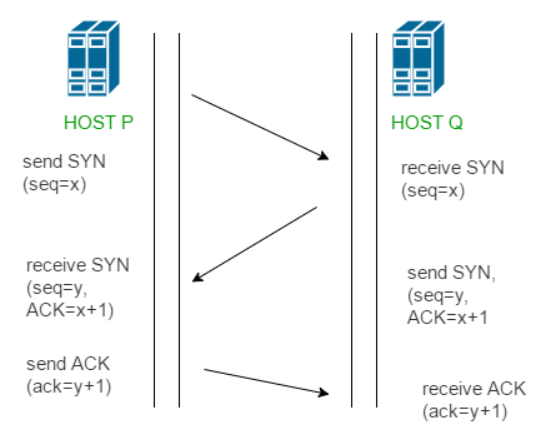
בפרוטוקול TCP, טרם התחלת העברת נתונים בין שרת ולקוח, על שני הצדדים להקים קשר TCP. על השרת להיות מוכן להקמת הקשר על ידי כניסה למצב האזנה, בעוד הלקוח מצדו, צריך ליזום את הקמת הקשר בעזרת פרוטוקול לחיצת היד המשולשת.

הפרוטוקול מורכב משלושה שלבים:

**שלב א' - SYN:** הלקוח שולח הודעה מיוחדת לפתיחת קשר. בהודעה זו דגל ה – SYN "למעלה" ובנוסף היא נושאת מספר רנדומלי X המהווה את תחילת רצף ההודעות מצדו של הלקוח (sequence number).

**שלב ב' - SYN-ACK**: השרת מקבל את ההודעה ושולח בתגובה הודעת אישור קבלה ואישור פתיחת קשר מצדו. בהודעה זו דגלי ה- SYN וה- ACK בפתיח נושאים ערך "1" ובנוסף היא מכילה מספר רנדומלי Y המהווה את תחילת הרצף מצדו של השרת. המספור ההתחלתי הוא מספר אקראי שמכונה Initial Sequence Number או בקיצור ISN. הסיבה לכך שהמספור לא מתחיל ב-0 אלא במספר אקראי היא חשש מערבוב שתי שיחות שנערכות במקביל וחשש מניחוש המספר על ידי גורם שלישי. משקיבל השרת את הודעת פתיחת הקשר מהלקוח הוא יכול כבר להתחיל לשלוח נתונים. בעזרת ה- ACK שהועבר לצד א, יוכלו הצדדים לתקשר.

**שלב ג' -ACK**: הלקוח מיידע את השרת על סיום מיסוד הקשר בהודעת ACK, בה דגל ה- ACK בפתיח נושא ערך "1". הלקוח מיידע את השרת על מספורו ההתחלתי.



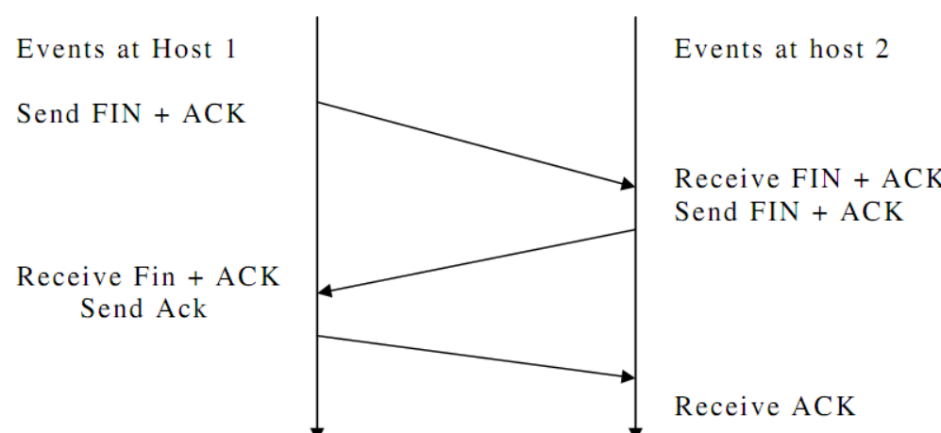
1. סגירת קשר - סגירת קשר מתבצעת באופן עצמאי כאשר כל צד מעביר חבילת FIN, והצד השני מאשר. לאחר אישור חבילת FIN לא יקבל הצד שאישר את החבילה חבילות נוספות. מכיוון ששני הצדדים צריכים לבקש את ניתוק הקשר (ולאשר אותו) ניתן לאפשר חיבור "חצי פתוח" אם רק אחד הצדדים סגר את הקשר.

לרוב מתבצעת סגירת הקשר על ידי לחיצת יד משולשת באמצעות שליחת ההודעות הבאה:

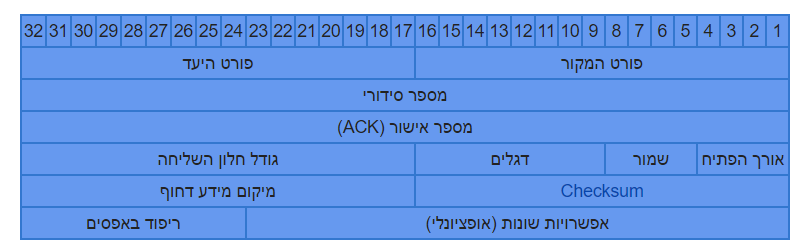
**FIN-ACK**: תחנת המקור מיידעת שהיא מעוניינת בסגירת הקשר ושולחת הודעת ACK על ההודעה האחרונה שהתקבלה והודעת ניתוק (הודעה בפרוטוקול TCP בה דגלי ה- ACK וה- FIN נושאים ערך "1").

**FIN-ACK**: הודעה זהה מתחנת היעד המאשרת את קבלת בקשת הניתוק ומבקשת לנתק את הקשר גם כן (מנקודה זו לא מועברים עוד נתונים בין התחנות).

**ACK**: הודעה מצד תחנת המקור המאשרת את סגירת הקשר. (הודעה מסוג כזה לא נחשבת לחבילה הנושאת מידע)



1. **מבנה חבילת TCP**



* פורט מקור (16 bits) – שדה המכיל את מספר הפורט במחשב המקור
* פורט היעד (16 bits) – שדה המכיל את מספר הפורט במחשב היעד
* מספר סידורי - Sequence number (32 bits) – מכיל את המספר הסידורי של הבייט הראשון בהודעה הנוכחית. אם ביט ה – SYN מורם, זהו המספר הסידורי הראשוני - ISN.
* מספר אישור – ACK (32 bits)- שדה זה הינו רלוונטי רק אם ביט ה – ACK "למעלה". מכיל את המספר הסידורי הבא שהשולח מצפה לקבל.
* אורך הפתיח (4 bits) – יש להכפיל בארבע לקבלת המספר הנכון.
* שמור (6 bits) – לא בשימוש.
* דגלים (6 bits) – SYN,FIN, ACK, RST, PSH, URG
* גודל חלון השליחה (16 bits) – מכיל את כמות הבייט שהשולח מעוניין לקבל. חלון זה משמש לשליטה על זרימת ההודעות בין שני הצדדים. בצורה זו לא יתאפשר מצב בו צד אחד מציף את השני בהודעות מעבר לגודל הבאפר שלו, מצב אשר יגרום לאובדן הודעות.
* מיקום מידע דחוף (16 bits) – מידע המוגדר דחוף, מועבר ישירות למענו ואינו מועבר לבאפר ההודעות המתקבלות. הודעת TCP יכולה להכיל גם מידע דחוף וגם מידע טריוויאלי, כאשר המידע המוגדר דחוף נכתב בראש מיקום הדאטה בהודעת ה – TCP. שדה זה מכיל את המיקום בו נגמר המידע הדחוף ומתחיל המידע הטריוויאלי בהודעה.

**שאלות:**

1. פתח wireshark ותתחיל להסניף את תעבורת הרשת שעוברת במחשב. כנס לדפדפן וגלוש באינטרנט. חפש הקמת תקשורת בTCP (Three-way handshake) צלם והסבר את התהליך. שים דגש איזה דגלים עולים.
2. הצג בטבלה את ההבדלים בין תקשורת TCP לתקשורת UDP
3. הצג 4 פרוטוקולים מוכרים שמשתמשים בתקשורת TCP

**שאלות בקוד:**

1. השאלות הבאות תצטרך לכתוב קוד בפיתון. בתיקיה נמצאים 2 קבצים – server.py, client.py. במשימות הבאות עליך להוסיף קוד לקבצים הללו. להפעלת הקבצים הרץ תחילה את הserver ולאחריו את הclient הserver יחכה עד אשר הclient יקליד משהו.

עבור על שני הקבצים ותבין כיצד הם עובדים.

בשאלות הבאות תצטרך להוסיף פעולות רק לserver.

1. בכל פעם שהserver מקבל משהו מהclient הוא צריך להדפיס למסך-

"The massage from the client is: [message]"

1. הserver לא מבדיל בין אותיות גדולות לאותיות קטנות, עזרו לו שבכל קריאה של ההודעה זה לא יהיה משנה אם האותיות גדולות או קטנות.
2. הserver יודע לענות על דברים מסוימים כמו מה השעה, פרטים עליו, מה הIP שלו. (אם עובדים בזוגות יש לרשום את שני השמות והת"ז מופרדים בפסיק)

* כאשר הserver מקבל time הserver ישלח את התאריך והשעה (ניתן להשתמש בdatetime.now()
* כאשר הserver מקבל name הserver ישלח את השם שלך

"my name is [name]"

* כאשר הserver מקבל id ה server ישלח את הת"ז שלך

"my ID is [ID]"

* כאשר הserver מקבל ip הserver ישלח את הip שלו

"my ip is [IP]"

* בנוסף הserver מבצע בדיקה מעמיקה של התרגיל שלכם ונותן לכם ציון מ0 עד 100.

כאשר הserver מקבל getmygrade הserver יגריל מספר בין 0 ל 100 וישלח אותו

"your grade is [rand\_num]"

* בכל קבלה אחרת הserver יחזיר

"I don’t know what to do with: [message]"

1. כעת הclient שולח (שלח floodattac ) – הסבירו מה הclient מבצע ומה קורה בserver.
2. הציעו 3 פתרונות לבעיה של הserver, כלומר כיצד הserver יוכל למנוע את ההתקפה. (פתרונות תיאורטיים, לא צריך לממש אותם).
3. כעת עליכם לממש בserver את אחת הדרכים שחשבתם עליהם. בסיום התהליך הserver ינתק את ההתקשרות עם הclient. (רמז- אתם יכולים להשתמש בפונקציה time.time() ובcounter שמוגדרים לכם בתחילת הקוד)
4. פתח wireshark הצג והסבר את הדברים הבאים
5. Three-way-handshake בין הserver לclient.
6. כל אחת מהפקודות שהserver מבצע
7. floodattack
8. סיום התקשרות עם הclient

**SYN Flooding attack \ Denial-Of-Service (DOS) attack**

טרם סיום לחיצת היד המשולשת, השרת מאחסן את הקשר החצי פתוח בתור ייעודי, בעל תכולה סופית. במתקפה זו, גורמים להתמלאות מהירה של התור ועל ידי כך מניעת השרת מלקבל הודעות פתיחת קשרים חדשים. בצורה זו, על אף שצד השרת פעיל, הוא מנוע ממתן שירות עבור לקוחות.

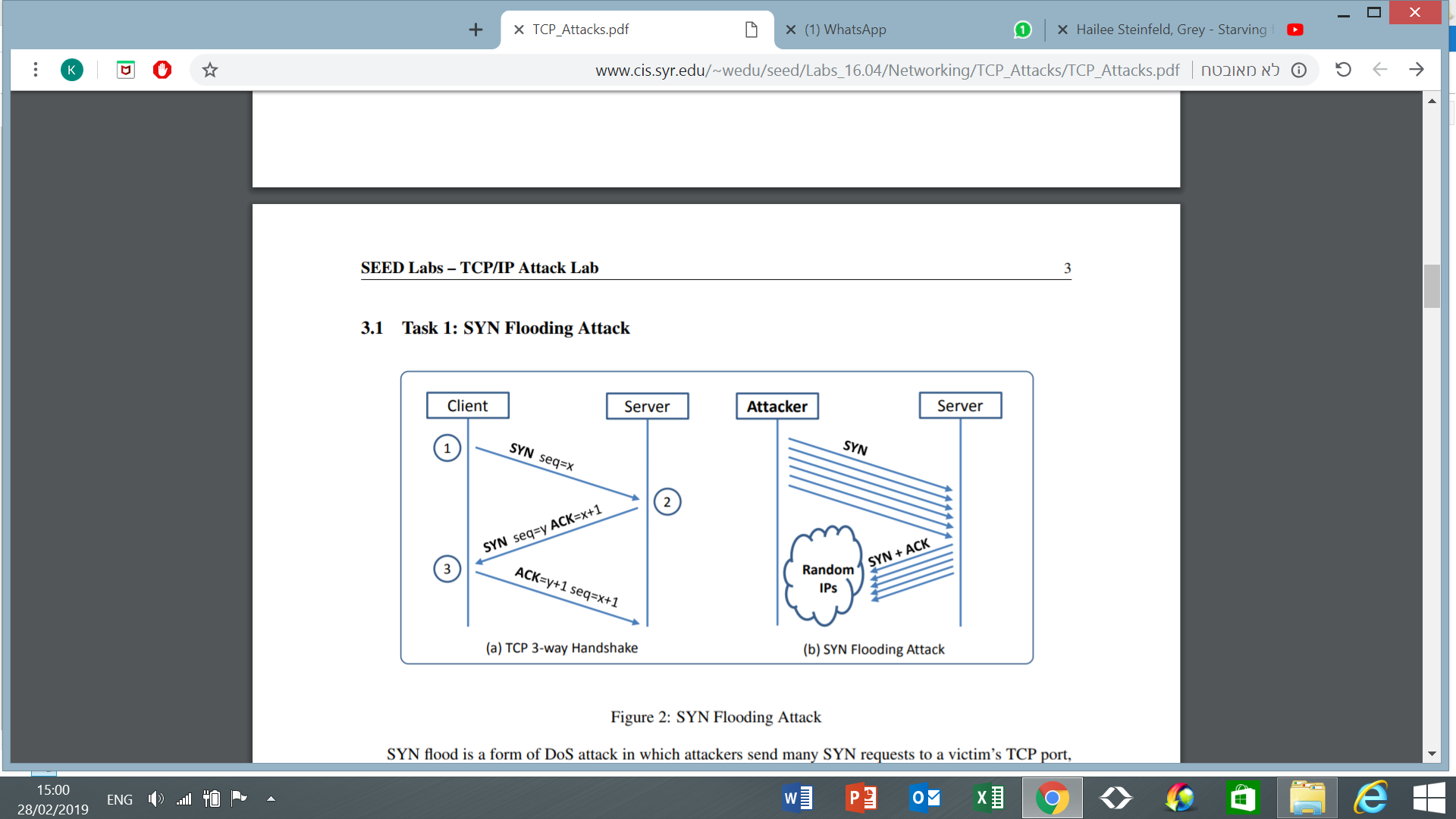
הוצאה של קשר חצי פתוח מהתור נעשית בשלושה מקרים:

1. בקבלת הודעת ה – ACK מהלקוח וסיום לחיצת היד המשולשת.
2. לאחר חלוף הזמן המוקצה לקבלת הודעת ה – ACK מהלקוח. זמן זה יכול להיות מאוד ארוך במונחים של זמן CPU.
3. קבלת הודעת איפוס (RST) עבור קשר חצי פתוח.

כל שיש לעשות על מנת לבצע את המתקפה אם כן, הינו כדלקמן:

1. שליחה עקבית של הודעות בקשה לפתיחת קשר, מכתובות שונות (אחרת המתקפה תחסם בקלות על ידי ה – firewall).
2. אין לבצע את השלב השלישי בלחיצת היד המשולשת.

עם קבלת הודעה לפתיחת קשר, השרת שולח חזרה הודעה המאשרת את קבלת ההודעה שנשלחה ומבקשת פתיחת קשר מצד השרת עם הלקוח. אם הודעה זו מגיעה לכתובת שלא שלחה קודם לכן הודעת בקשה לקשר מהשרת הנתון, תשלח הודעת איפוס (RST) מהכתובת האחרונה חזרה לשרת וזו תגרור הוצאת הקשר החצי פתוח מהתור. יחד עם זאת, מקרה זה אינו שכיח ולכן ללא השלמת לחיצת היד המשולשת, רוב הקשרים החצי פתוחים יישארו בתור עד חלוף הזמן המוקצה.



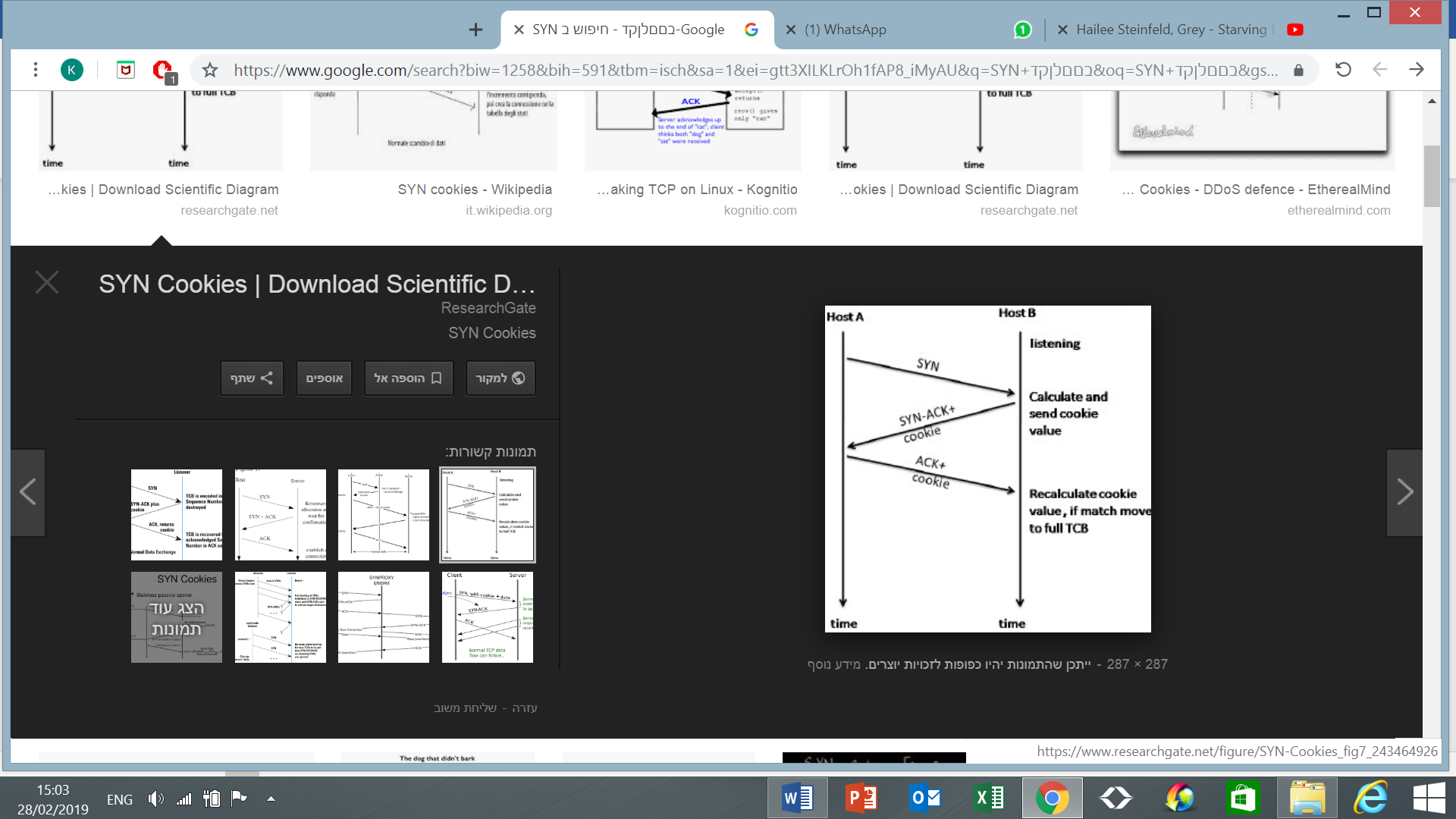
**שיטת המנע**

שיטה יעילה להגנה מפני המתקפה נקראת SYN cookies. השיטה הומצאה על ידי דניאל ג' ברנשטיין בשנת 1996. שיטה זו כיום מהווה חלק סטנדרטי ממערכות ההפעלה השונות. העיקרון המרכזי בשיטה הינו לא להקצות משאבים כלל בצד השרת, עבור הקמת הקשר, עד קבלת הודעת ה – ACK מהלקוח, בשלב השלישי בלחיצת היד המשולשת.

שיטה זו פותרת אמנם את בעיית ההצפה אך פותחת שער למתקפה אחרת. מאחר ולא מוקצים משאבים כלל, אין דרך לדעת האם הודעת ה – ACK שנתקבלה בשרת הינה כמשוב לשלב השני של לחיצת היד המשולשת או הודעה מזויפת. כתוצאה מכך, ניתן יהיה להציף את השרת בהודעות ACK מזויפות שיגררו הקמת קשרי TCP ריקים. למתקפה זו פוטנציאל הרס גדול משל המקורית, שכן המשאבים המוקצים לקשר שלם גדולים מאשר לקשר חצי פתוח.

SYN cookies פותרת זאת בצורה הבאה: לאחר קבלת בקשה לפתיחת קשר מהלקוח, השרת מחשב ערך, H, בעזרת מפתח האש, הידוע רק לו, מהכתובת ממנה נשלחה ההודעה, מספר הפורט והמספר הסידורי. H ישלח כמספר הסידורי בהודעת ה – SYN+ACK, שתשלח כתגובה ללקוח, מהשרת. אם ההודעה המקורית הגיעה מתוקף, הודעת התגובה תגיע לכתובת מזויפת ולא נקבל הודעת ACK לסיום לחיצת היד המשולשת. אולם, השרת טרם הקצה משאבים כלשהם לקשר ולכן לא נגרם כל נזק. אם ההודעה המקורית נשלחה מלקוח רלוונטי, הוא יקבל את הודעת ה – SYN+ACK, עם המספר הסידורי H, ויחזיר הודעת ACK עם הערך H+1, בשדה מספר האישור. בקבלת הודעה זו, השרת יכול לחשב חזרה את ערך ההאש ולבדוק האם ההודעה הגיע מלקוח רלוונטי או מתוקף. רק אם החישוב יראה על לקוח רלוונטי, יקצה השרת משאבים לקשר ה – TCP. מפתח ההאש בו משתמש השרת ידוע רק לו ועל כן, יהיה קשה מאוד לתוקפים לזייף הודעות ACK שיגררו הקמת קשרי סרק.

שיטת SYN cookies מאפשרת, אם כן, הגנה הן מ – SYN flooding attack, שכן אף משאב לא יוקצה להקמת קשר עם קבלת בקשה לפתיחת קשר והן מפני ACK flooding attack, שכן משאבים יוקצו רק עבור לקוחות רלוונטיים.



**שאלות חזרה בנושא SYN Flooding attack:**

1) תאר את אופן ביצוע מתקפת SYN Flooding.

2) תאר את נקודת התורפה המנוצלת עבור מתקפת ה- SYN Flooding.

3) תחת אילו תנאים מתרחשת הוצאת קשר חצי-פתוח מן התור המיועד לה?

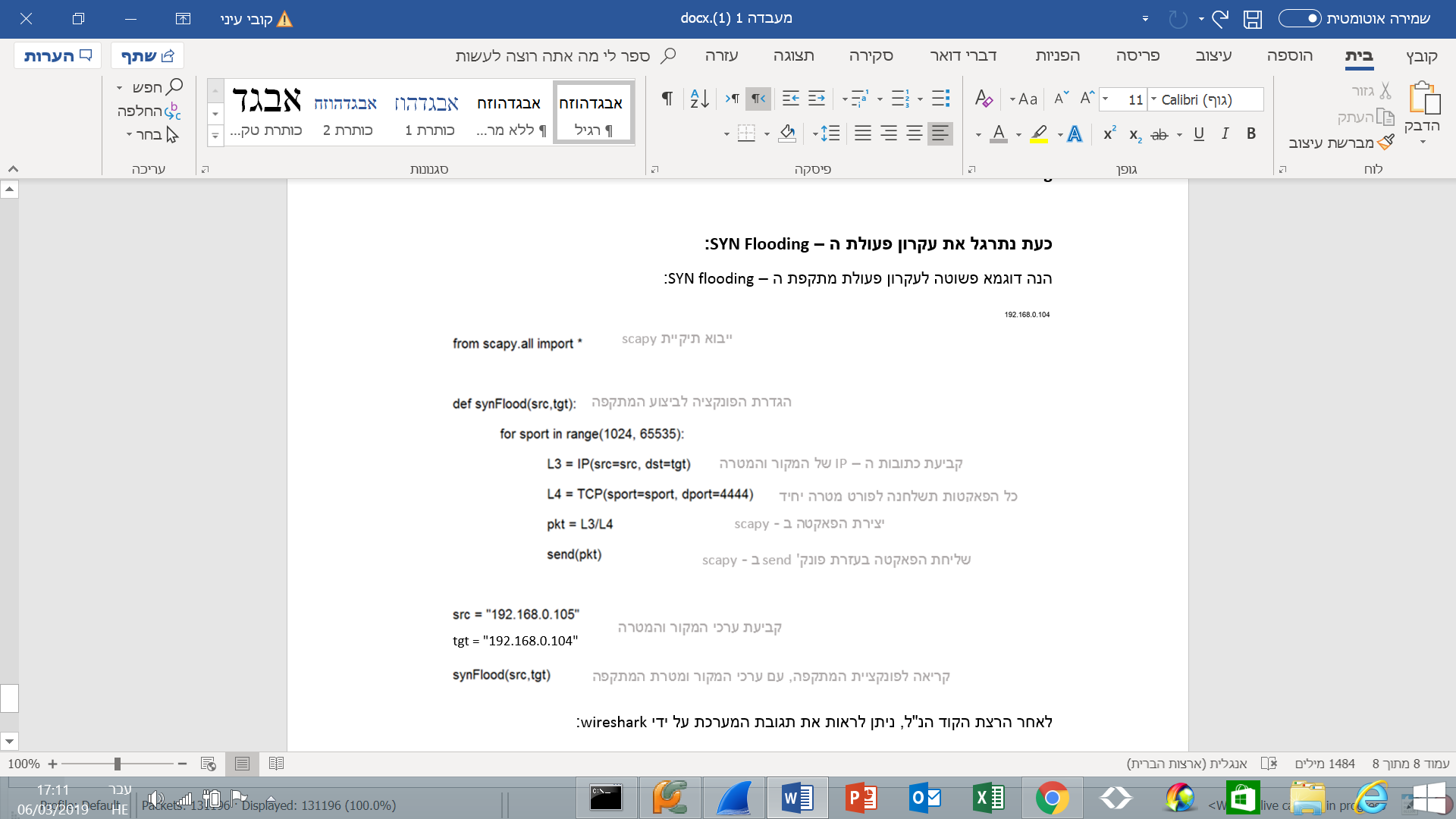
4) תאר את שיטת המנע SYN – cookies.

5) מהו העיקרון המנחה בשיטת המנע SYN – cookies?

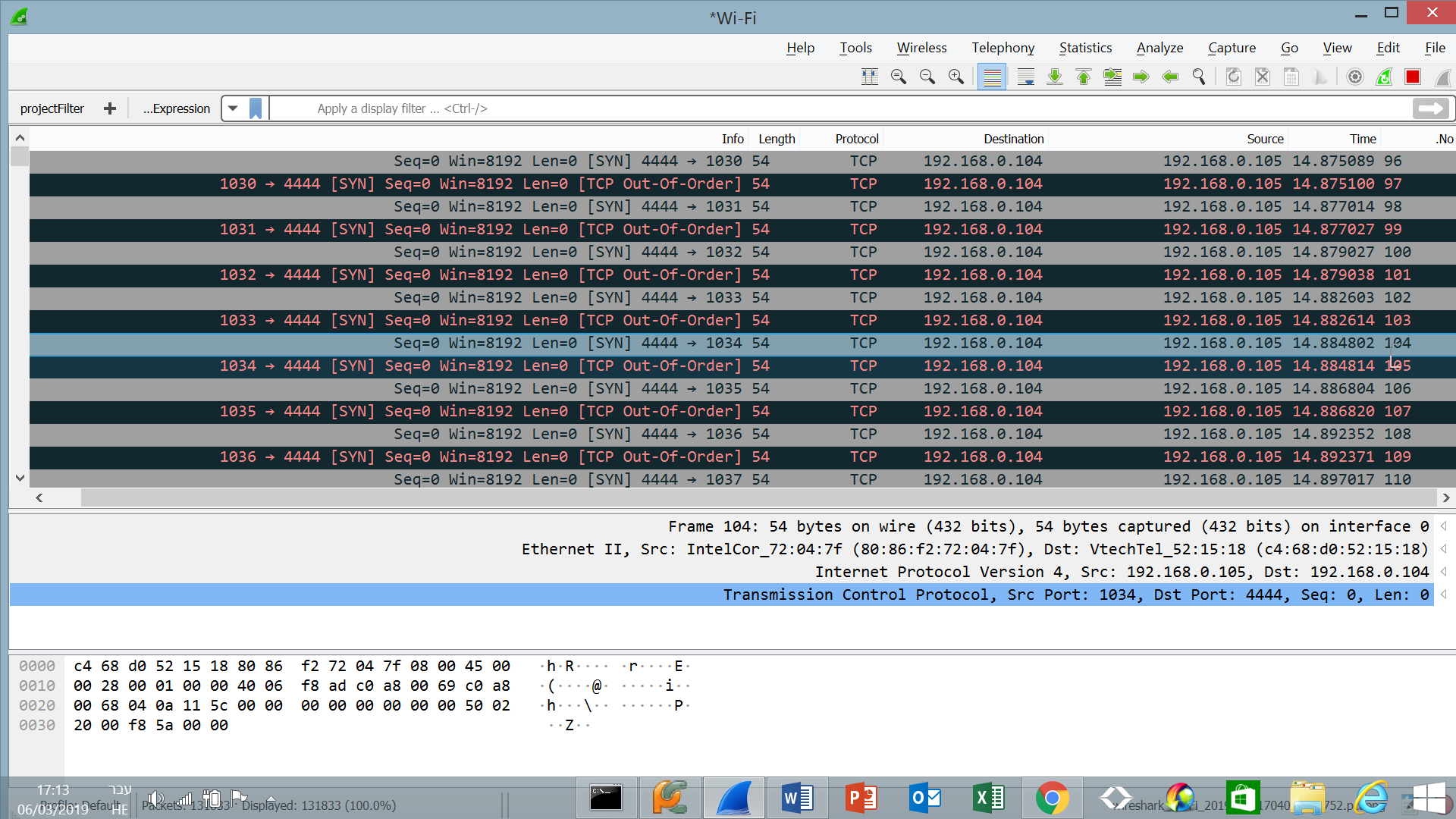
6)טיפול במתקפת SYN Flooding עלול לפתוח שער למתקפה נוספת. תאר את המתקפה הזו וכיצד שיטת SYN cookies מתמודדת אתה. האם מתקפה זו מסוכנת יותר מ – SYN – Flooding?

**כעת נתרגל את עקרון פעולת ה – SYN Flooding:**

הנה דוגמא פשוטה לעקרון פעולת מתקפת ה – SYN flooding:



לאחר הרצת הקוד הנ"ל, ניתן לראות את תגובת המערכת על ידי wireshark:



כמו שניתן לראות, קטע הכותב שכתבנו גורם להצפת כתובת המטרה בהודעות SYN המגיעות לכאורה מפורטים שונים.

**עתה, נבצע מתקפת SYN Flooding על השרת שכתבנו:**

היכנסו לסביבת העבודה kali, והריצו את השרת וכמה לקוחות.

כעת פיתחו חלון Terminal נפרד וכתבו את הפקודות הבאות:

root@kali:~# sysctl -w net.ipv4.tcp\_syncookies=0

net.ipv4.tcp\_syncookies = 0

שורה זו מבטלת את מנגנון ההגנה SYN cookies.

כעת, הכניסו את הפקודה הבאה:

root@kali:~# hping3 -c 5000 -d 120 -S -w 64 -p 4444 --flood --rand-source 10.0.2.15

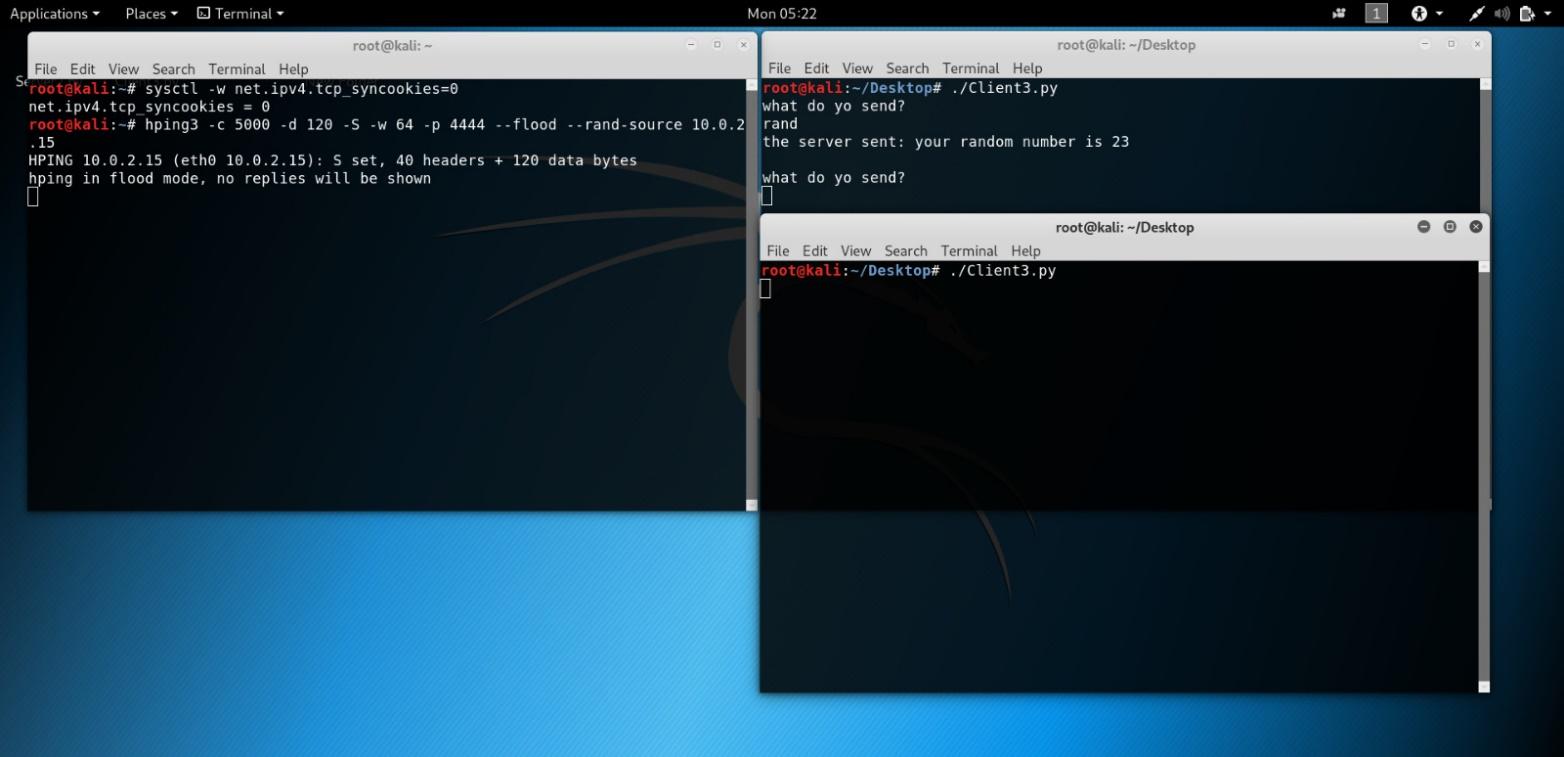
HPING 10.0.2.15 (eth0 10.0.2.15): S set, 40 headers + 120 data bytes

hping in flood mode, no replies will be shown

שורה זו מבצעת את המתקפה עצמה, על ידי מודול hping3 המוטמע במערכת ההפעלה kali.

שימו לב להכנסת כתובת ה – IP והפורט עליהם מאזין הסרבר שרשמתם.

לאחר הרצת שורה זו, נסו להתחבר לשרת על ידי לקוח נוסף.



עתה, יש לבטל את המתקפה על ידי לחיצת clt+c בטרמינל בו מתבצעת המתקפה:

^C

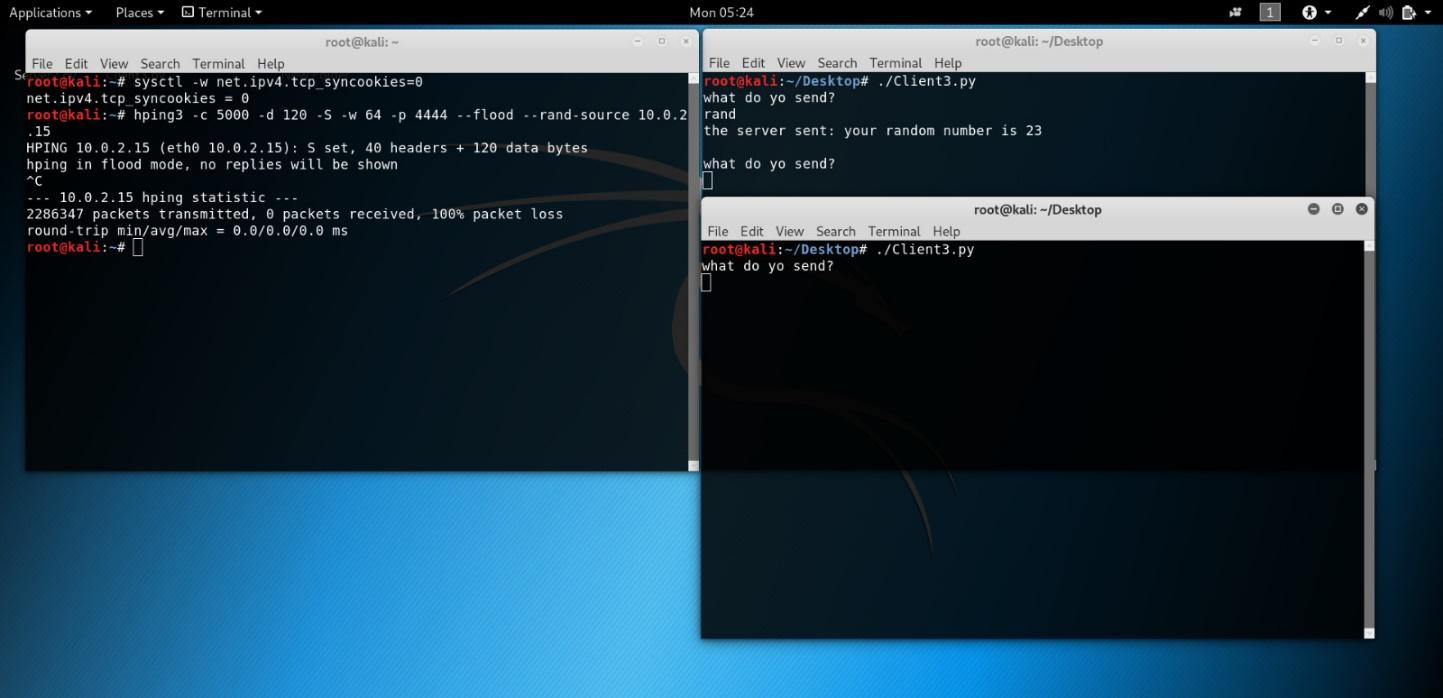
--- 10.0.2.15 hping statistic ---

422121 packets transmitted, 0 packets received, 100% packet loss

round-trip min/avg/max = 0.0/0.0/0.0 ms

root@kali:~#

לאחר כמה דק', ניתן לראות כי הלקוח יצליח להתחבר לשרת.



הפעילו את מנגנון ההגנה SYN cookies על ידי הרצת השורה:

root@kali:~# sysctl -w net.ipv4.tcp\_syncookies=1

net.ipv4.tcp\_syncookies = 1

הריצו שנית את שורת ה -hping3, ונסו להתחבר לשרת על ידי לקוח חדש.

כמו שניתן לראות, מנגנון ההגנה מאפשר המשך עבודה כרגיל, למרות הרצת המתקפה על השרת.