דו"ח ניסוי: הדמיית מתקפת סייבר מסוג DdoS ובחינת השפעותיה

שמות הסטודנטים:

דור כהן - 211896279

ברוך יפראימוב – 208526012

תאריך: 7.4.2025

1. מטרת המעבדה

מטרת המעבדה היא לבחון ולהבין את אופן הפעולה של מתקפת SYN Flood – אחת מטכניקות התקיפה הנפוצות בעולם ממתקפות ה־ DoS (Denial of Service) תוך שימוש בשתי שפות תכנות שונות C ו־Python.
במהלך המעבדה ביצענו תקיפה מדומה על שרת Apache שמדמה סביבה אמיתית של שרת אינטרנט פעיל. המתקפה התבצעה על ידי שליחת מספר רב מאוד של בקשות TCP מסוג SYN מבלי להמשיך את ה־3way handshake- ובכך ליצור עומס על שרת היעד.

במהלך הניסוי, נבחנו המדדים הבאים:

- **מהירות שליחת חבילות SYN** על ידי תוכניות התקיפה שנכתבו ב־C וב־Python.
- **השפעת ההתקפה על השרת** באמצעות ניתוח זמן תגובה (RTT) של פינגים שנשלחו אליו משרת ניטור.
- **השוואת יעילות** של כל שפת תכנות, הן מבחינת הביצועים (כמות ומהירות השליחה), והן מבחינת עומס שנגרם לשרת.
- **הסקת מסקנות** בנוגע להתאמה של כל שפה לתרחישי תקיפה מסוג זה, תוך התייחסות לזמן פיתוח, שליטה בפרוטוקול וביצועי הרשת.

בנוסף, המעבדה מדגישה את הקשר בין מבנה התוכנית לבין השפעתה על רשת תקשורת אמיתית ,ואת החשיבות של תכנון נכון, הבנה של שכבות התקשורת ויעילות של קוד בסביבות עומס גבוה.

2. סביבת הניסוי

לצורך ביצוע המעבדה והדמיית תרחיש של מתקפת SYN Flood בעולם האמיתי, הקמנו סביבה וירטואלית שכללה שלוש מכונות נפרדות, שכל אחת מהן ממלאת תפקיד ייעודי:

TYPE	VM IP	OS	תפקיד
Attacker	10.211.55.5	Kali GNU/Linux Rolling	מכונת התקיפה שמריצה את תוכניות ה־SYN Flood שכתבנו בשפת C ובשפת Python על המכונה שאנו תוקפים.
Target	10.211.55.3	Ubuntu 24.04 LTS with apache2	שרת Apache המדמה שרת אינטרנט אמיתי, אשר מקבל את ההתקפה.
Monitor	10.211.55.4	Ubuntu 24.04 LTS	מכונה ניטורית שתפקידה לבדוק את השפעת ההתקפה על השרת. היא שולחת פינגים כל 5 שניות אל ה־Target במהלך ההתקפה, ומודדת את זמן התגובה.(RTT)

3. מה נתבקשנו לבצע:

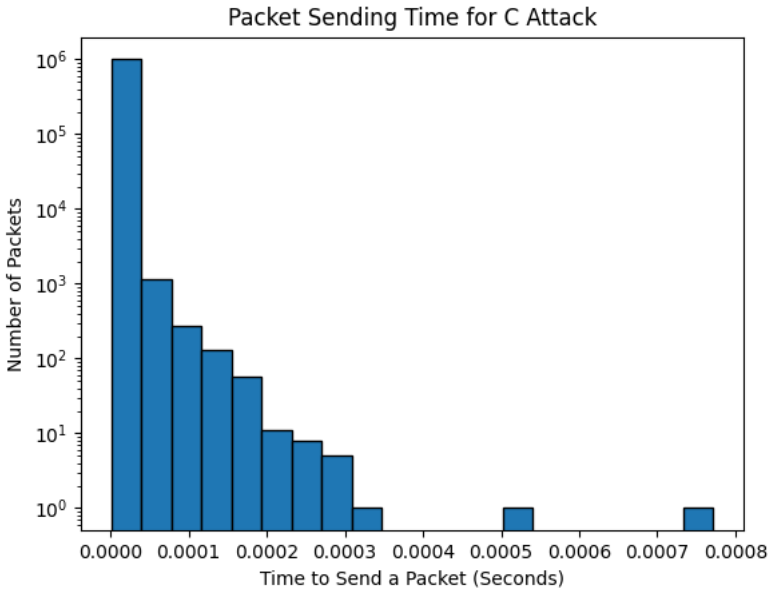
במסגרת המעבדה נתבקשנו לבצע סימולציה של מתקפת DDOS בסביבת מעבדת סייבר. התוכנית כללה שימוש בשלושה מחשבים – תוקף, יעד ומפקח – כאשר התוקף שלח 1 מיליון חבילות SYN (במספר 10,000 חבילות בלולאה של 100 חזרות) למכונה המריצה שרת Apache, והמפקח שלח פינגים לשרת במהלך ההתקפה. בנוסף, נכתבו שני תוכניות התקפה – אחת בשפת C ואחת בפייתון – כדי למדוד את זמני השליחה הממוצעים של החבילות, ואת זמני תגובת הפינג (RTT). עלינו לאסוף את המדידות לקבצי תוצאות, לייצר גרפים באמצעות matplotlib עם ציר y לוגריתמי וניתוח סטטיסטי (ממוצע וסטיית תקן), ולהציג את ההבדלים בין שתי הגישות בתקשורת ובביצועים.

לאחר ביצוע ההתקפה ואיסוף הנתונים, ניתחנו את הממצאים שהתקבלו תוך השוואת הביצועים בין תוכנית ההתקפה בשפת C לזו שנכתבה בפייתון. הנתונים נאספו לקבצי תוצאות הכוללים את זמני השליחה של כל חבילה ופינגים שנשלחו, ובכך קיבלנו תמונה מלאה של ביצועי ההתקפה והשפעתה על השרת.

בנוסף, יצרנו גרפים באמצעות matplotlib המציגים את התפלגות זמני השליחה וה – RTT-כאשר ציר ה y-הוגדר כערכים לוגריתמיים, והגרפים מאפשרים זיהוי ברור של ההבדלים והחריגות בביצועים. ניתוח סטטיסטי מעמיק (כולל ממוצע וסטיית תקן) הושם כדי להצביע על עיקרי השוני בין שתי הגישות, מה שמספק תובנות חשובות לגבי יעילות ההתקפה וההשפעה על המערכת.

4. תוצאות וגרפים

4.1. זמני שליחת – SYN (תוכנית ב־C)

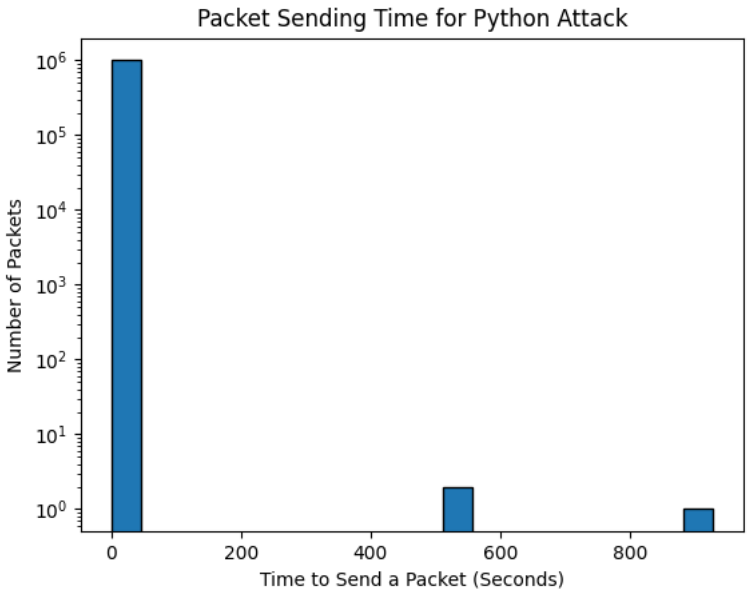


Syn_pkts_c.png : 1 Figure

- **ממוצע:** 0.00000491 חבילות לשנייה
- **סטיית תקן :** 0.00000449 שניות
- **מספר חבילות שנשלחו:** 1000000 חבילות.
- **משך (בשניות) ההתקפה:** 7.393 שניות.

הגרף מציג את התפלגות זמני שליחת החבילות במתקפת SYN שבוצעה באמצעות קוד בשפת C. בציר האופקי מוצגים זמני השליחה של חבילות בשניות, ובציר האנכי (בלוגריתם) – מספר החבילות שנשלחו בכל טווח זמן. מהגרף ניתן להסיק כי רוב מוחלט של החבילות נשלחו בפרק זמן קצר מאוד, של 0–50 מיקרו־שניות, מה שמעיד על ביצועים מהירים ויעילים במיוחד של הקוד בשפת C. ככל שזמן השליחה מתארך, מספר החבילות פוחת באופן חד, דבר שמצביע על התפלגות אקספוננציאלית. נתון זה מדגיש את יתרונה של שפת C ביכולת לשלוח חבילות בתדירות גבוהה ובזמן מינימלי, תכונה קריטית במתקפות מסוג DoS ו־DDoS שבהן נדרש להציף את היעד בכמות מקסימלית של חבילות בזמן קצר.

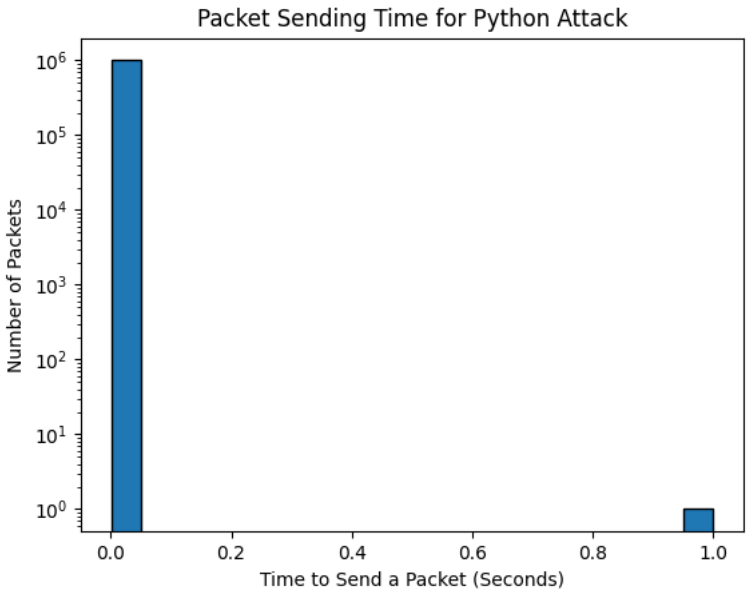
4.2. זמני שליחת – SYN (תוכנית בpython)



- **ממוצע:** 0.036746791 חבילות לשנייה
- **סטיית תקן :** 1.183 שניות
- **מספר חבילות שנשלחו:** 1000000 חבילות.
- **משך (בשניות) ההתקפה:** 36806.854 שניות (או ~10.2 שעות).

הגרף מציג את התפלגות זמני השליחה של חבילות SYN במהלך מתקפה שבוצעה באמצעות סקריפט Python, כאשר בציר האופקי (בשניות) מופיע זמן השליחה לכל חבילה, ובציר האנכי (בלוגריתם) מספר החבילות שנשלחו בכל טווח זמן. ניתן לראות שרוב החבילות נשלחו בפרק זמן קצר במיוחד (בסביבות עשרות מיקרו־שניות), אך יחד עם זאת קיימות חריגות המתבטאות בזמני שליחה ארוכים בהרבה. הממוצע המחושב עומד על כ־0.0367 חבילות לשנייה, בעוד שסטיית התקן (כ־1.183 שניות) מציגה שונות גבוהה למדי, המעידה על תנודות משמעותיות בין זמני השליחה. בסך הכול נשלחו כמיליון חבילות במשך כ־36,806.854 שניות (כ־10.2 שעות), כאשר המאפיינים האיטרטיביים של Pythonוהאופי המפורש של פירוש הקוד בזמן ריצה גורמים להארכת זמני השליחה בפועל ולהיווצרות פיזור רחב בנתונים.

4.3. זמני שליחת – SYN (תוכנית ב־Python: גרסת קיבוץ שליחת הודעות של 10K.)



Syn_pkts_p.png : 2 Figure

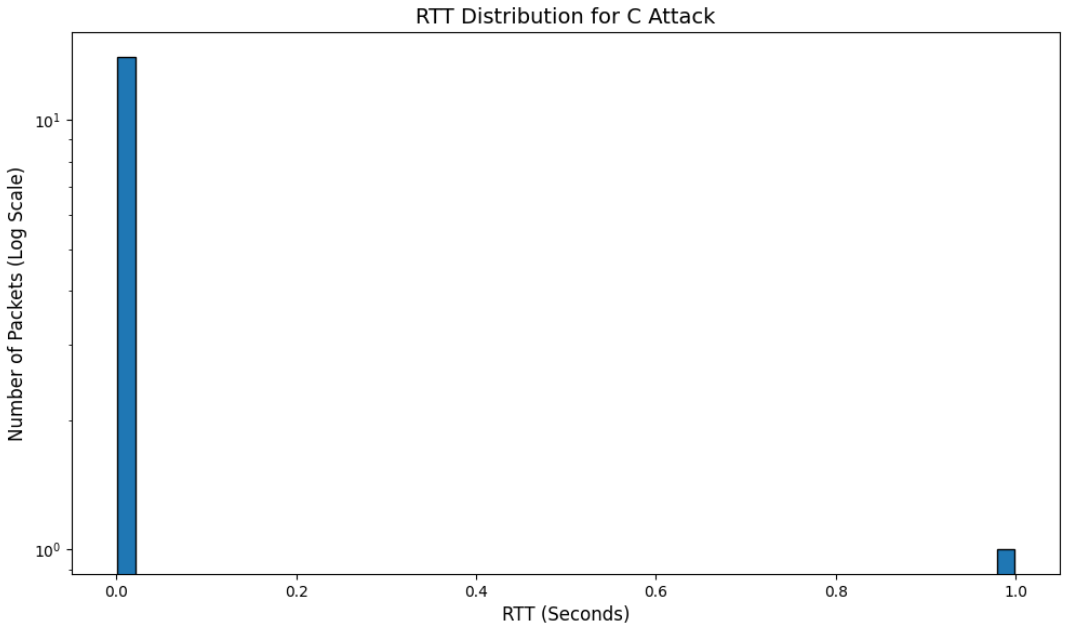
- **ממוצע**: 0.557147 חבילות לשניה
- **סטיית תקן :**0.000558 שניות.
- **מספר חבילות שנשלחו**: 1000000 חבילות.
- **משך (בשניות) ההתקפה:** 558.391 שניות.

הגרף מציג את התפלגות זמני השליחה של חבילות SYN במהלך מתקפת DDOS, כאשר החבילות נשלחות בקבוצות של 10,000 בכל איטרציה. על ציר ה-X מופיע הזמן במיקרושניות ועל ציר ה-Y מופיע מספר החבילות שנרשמו בכל טווח זמן. רוב הקבוצות מציגות זמני שליחה עקביים של פחות מ-100 מיקרושניות, כאשר מתועד מקרה חריג אחד בו זמן השליחה התארך עד לכדי שנייה שלמה.

החלוקה לשליחה בקבוצות מאפשרת חישוב ממוצע עבור כל קבוצה, מה שמספק תמונה ברורה ויציבה של זמני השליחה הכוללים של ההתקפה. הנתונים המוצגים בגרף ממחישים את עיקרון המדידה הסטטיסטית – רוב החבילות נשלחות במהירות גבוהה, בעוד חריגות בודדות מצביעות על עיכובים מערכתיים מסוימים.

4.4. זמני RTT לפינג בזמן התקפה ב־C

גרף Pings_c.png :



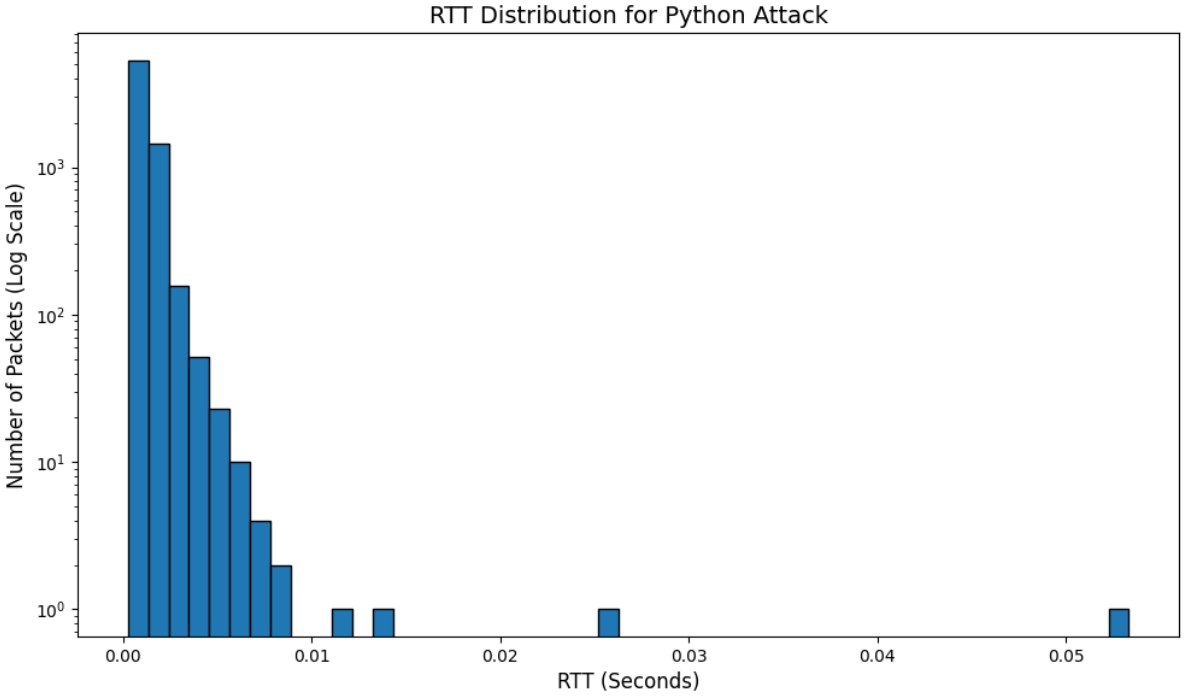
- **RTT ממוצע :** 0.00539 שניות
- **סטיית תקן :** 16.58
- **מספר פינגים שנשלחו :** 18, מתוכם 17 פינגים תקינים ו 1 פינגים שנכשלו (כלומר נגרם חוסר תקשורת מול השרת ונגרם לTIMEOUT). יש רק אחד מכיוון ששלחנו את הפקטות באופן מהיר מאוד אבל הוא מצביע על Dos
- **זמן ממוצע בין כל שליחת פינג:** 5 שניות.
- **משך ניתור המכונה:** 71.61 שניות.

הגרף מציג את התפלגות זמני ה־ RTTבמהלך מתקפה שבוצעה באמצעות סקריפט שנכתב בשפת C. ציר ה־ X מייצג את זמן התגובה בפינג (RTT) בשניות, בעוד שציר ה־ Y (בסקלת לוגריתם) מייצג את כמות החבילות שנמדדו בכל טווח זמן.

ניתן להבחין שרוב הודעות הפינג קיבלו מענה בטווחים קצרים מאוד – סביב 5 – msדבר אשר מצביע על פינגים שהתבצעו כנראה בתוך אותה רשת מקומית ,(LAN) ולכן זמני התגובה שלהם כמעט אפסיים.במהלך הבדיקה נשלחו **18 פינגים** ,מתוכם **17 התקבלו בהצלחה** ואילו פינג אחד נכשל (timeout) .

אף על פי שרוב החבילות זכו למענה מהיר, עצם הופעת ה־ timeoutמצביעה על כך שהשרת קרס או הפך לבלתי זמין בשלב מסוים של ההתקפה. העובדה שנראה רק כישלון אחד בלבד נובעת ככל הנראה מהקצב הגבוה של שליחת הפינגים – כלומר, ההתקפה גרמה לנפילת השרת במהירות, אך מאחר והמדידה בוצעה בטווח זמן קצר, הספקנו לתעד רק תגובה חריגה אחת. ממצא זה מחזק את ההשערה כי השרת לא עמד בעומס שהופעל עליו במהלך ההתקפה.

4.5. זמני RTT לפינג בזמן התקפה ב-Python



RTT ממוצע : 1.17 שניות

סטיית תקן : 0.441 שניות

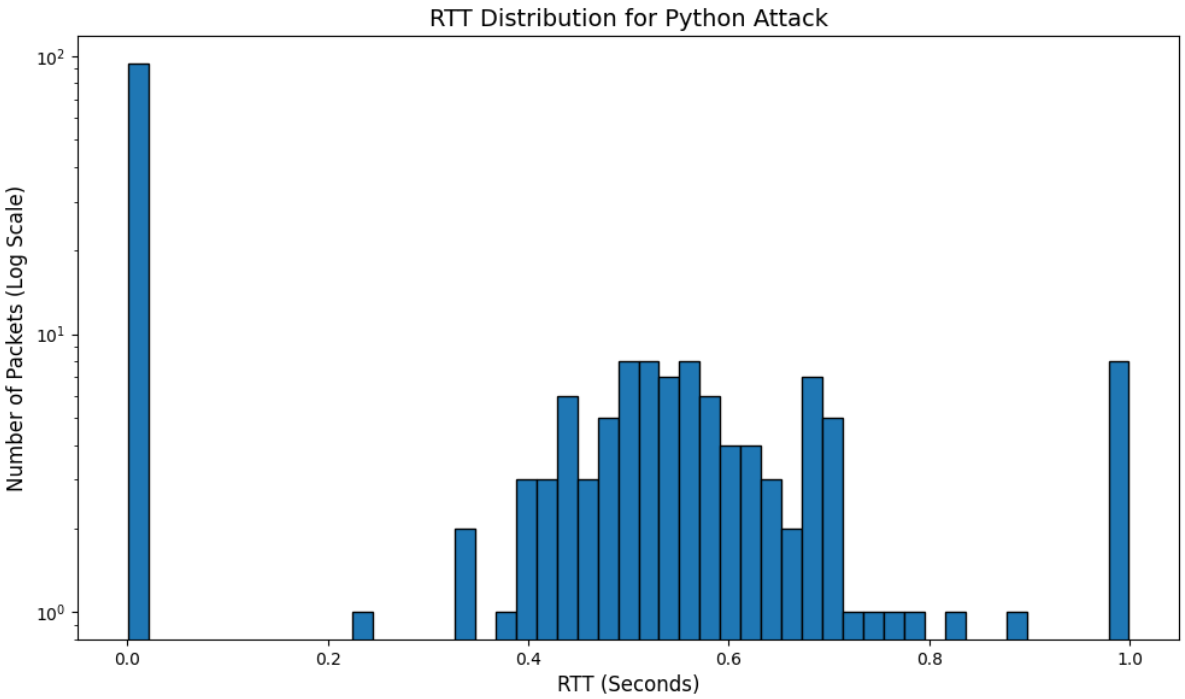
מספר פינגים שנשלחו : 6974

זמן ממוצע בין כל שליחת פינג: 5 שניות.

משך ניתור המכונה: 36868.43 שניות.

הגרף מציג את התפלגות זמני ה־RTT של הודעות פינג שנשלחו במהלך מתקפה שבוצעה באמצעות סקריפט Python. ציר ה־X מייצג את זמן התגובה בשניות, בעוד שציר ה־Y (בסקלת לוגריתם) מציג את מספר הפינגים שנמדדו בכל טווח זמן. ניתן לראות כי מרבית הפינגים התקבלו בזמן תגובה קצר, אך בעקבות ממוצע של כ־1.17 שניות וסטיית תקן של כ־0.441 שניות, ניכרת נוכחות של ערכים גבוהים בהרבה. במהלך הבדיקה נשלחו 6,974 פינגים במרווח של 5 שניות ביניהם, והמדידה נמשכה כ־36,868 שניות (כ־10 שעות). נתונים אלו מצביעים על כך שהמתקפה יצרה עומס מתמשך על השרת או על הרשת, והובילה לעלייה ברורה בזמני התגובה, תופעה המאפיינת מצבי התשה או מגבלת משאבים הנובעים מהפעלת הסקריפט ב־Python.

4.6. זמני RTT לפינג בזמן התקפה ב־Python10K



Pings_p.png :3 Figure

- **RTTממוצע :**0.276 שניות
- **סטיית תקן :** 0.294 שניות.
- **זמן ממוצע בין כל שליחת פינג:** 5 שניות.
- **משך ניתור המכונה:** 1027 שניות
- **מספר פינגים שנשלחו :** 197, מתוכם 189 פינגים תקינים ו 8 פינגים שנכשלו (כלומר נגרם חוסר תקשורת מול השרת ונגרם לTIMEOUT)

הגרף מציג את התפלגות זמני ה־RTT במהלך מתקפת עם סקריפט שנבנה ב Python, כאשר ציר X מייצג את זמן התגובה במילישניות וציר Y (בלוגריתם) מייצג את כמות החבילות שנמדדו בכל טווח זמן. ניתן להבחין בריכוז גבוה של חבילות עם זמן תגובה כמעט אפסי, מה שנבוע מפינגים פנימיים בתוך הרשת. בנוסף, מרבית הפינגים מתרכזים בטווח שבין ms300 ל־ms700, עם מספר חבילות שקיבלו זמני תגובה חריגים של עד ms1000, הודעות פינג אשר קיבלנו עליהם TIMEOUT הגדרנו אותם בתור 1000 לטובת הצגת הגרף. התפלגות זו, יחד עם ממוצע RTT של כ-276 מילי שניות וסטיית תקן גבוהה של כ-294 מילי שניות, מצביעה על שונות גבוהה מאוד בתקשורת – סימן מובהק לאי־יציבות רשתית כתוצאה מעומס או מתקפה.

4.7. השפעת המתקפה על השרת:

כדי להמחיש את השפעת מתקפת ה־SYN Flood על השרת, נשתמש בפקודת tcpdump לצורך ניטור תעבורת הרשת. הפקודה תאפשר לנו לצפות בבקשות SYN שנשלחות אל השרת אך אינן נענות ב־ ACK, כלומר חיבורים "חצי פתוחים" שנשמרים בטבלת החיבורים של השרת. מצב זה מעמיס על משאבי השרת ועלול למנוע ממנו לטפל בבקשות חדשות: (מה שאנו רוצים לגרום בהתקפה זו)

```
sudo tcpdump -tttt -i any 'tcp[tcpflags] & tcp-syn != 0 and tcp[tcpflags] & tcp-ack == 0' | tee -a
syn_flood_output.txt
```

מה זה tcpdump?

tcpdump הוא כלי שורת פקודה ב־ Linux(ובמערכות יוניקס אחרות), שמאפשר ליירט ולנתח תעבורת רשת העוברת דרך ממשקי הרשת של המחשב שלך.

הוא משתמש ב־ libpcap ספרייה שמספקת גישה ישירה לחבילות רשת ברמת ה־ link layer(שכבת קו), כלומר לפני שהן מגיעות ליישומים או נבדקות ע"י מערכת ההפעלה.

הסבר על הדגלים:

- -tttt - מציג timestamp מדויק כולל תאריך, שעה, שניות ומילישניות.
- -i any - מאזין לכל הממשקים.
- tee -a - כותב גם לקובץ וגם למסך. (שנוכל לראות בעצמנו תוך כדי התקיפה)

להלן דוגמא מהקובץ שמראה את תוצאות ההתקפה:

```
2025-04-07 20:00:54.693945 enp0s5 In  IP 1.2.3.4.33224 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693945 enp0s5 In  IP 1.2.3.4.13132 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693945 enp0s5 In  IP 1.2.3.4.40995 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693945 enp0s5 In  IP 1.2.3.4.59734 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693945 enp0s5 In  IP 1.2.3.4.17606 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.17590 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.3626 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.48461 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.44598 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.62930 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.47053 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.48438 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.693972 enp0s5 In  IP 1.2.3.4.35717 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694027 enp0s5 In  IP 1.2.3.4.51765 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694027 enp0s5 In  IP 1.2.3.4.55478 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694053 enp0s5 In  IP 1.2.3.4.21880 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694053 enp0s5 In  IP 1.2.3.4.4599 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694146 enp0s5 In  IP 1.2.3.4.16916 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694197 enp0s5 In  IP 1.2.3.4.39047 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694211 enp0s5 In  IP 1.2.3.4.35406 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694211 enp0s5 In  IP 1.2.3.4.32946 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694211 enp0s5 In  IP 1.2.3.4.30442 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.47968 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.40301 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.26519 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.40970 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.32779 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.29705 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.35687 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694225 enp0s5 In  IP 1.2.3.4.18167 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694242 enp0s5 In  IP 1.2.3.4.1856 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694242 enp0s5 In  IP 1.2.3.4.2145 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
2025-04-07 20:00:54.694242 enp0s5 In  IP 1.2.3.4.30330 > apache2-ubuntu.shared.http: Flags [S], seq 0:992, win 5840, length 992: HTTP
```

Figure 4 הפלט מציג רצף של חבילות SYN שנשלחו בקצב גבוה לשרת HTTP ממקור זהה (עם IP 1.2.3.4 אשר הגדרנו בתור IP מקור בהתקפה, פלט זה הוא מתוכנית C), ללא השלמת Handshake.

5. ניתוח השוואתי

נרצה לבצע השוואה בין התוצאות שקיבלנו בשימוש ב- python לבין התוצאות בשימוש ב C:

שפת C - העניקה לנו שליטה ברמה נמוכה מאוד על בניית חבילות ה־TCP באמצעות raw sockets יכולנו להרכיב את כותרות ה־IP וה־TCP באופן ידני ולשלוט בכל שדה בחבילה. היתרון המרכזי בגישה זו הוא ביצועים גבוהים מאוד –זמן השליחה של כל חבילה היה קצר במיוחד, מה שאפשר שליחה של מאות אלפי חבילות בזמן קצר יחסית. בנוסף, השימוש הישיר בפרוטוקול אפשר לנו להבין לעומק כיצד חבילות נבנות ומתקבלות ברשת.

עוד יתרון חשוב הוא תהליך הקומפילציה המוקדמת של הקוד: תוכנית ה־Cמתורגמת מראש לקוד מכונה (binary) ומריצה את ההוראות ישירות, ללא צורך בinterpreter בזמן ריצה. כתוצאה מכך, זמן הביצוע של הקוד הוא מינימלי, מה שמתאים במיוחד לתרחישים שבהם דרושה יעילות מרבית.

עם זאת, שפת C דורשת ניהול מדוקדק של זיכרון ומבנה נתונים ,דבר שהופך את הקוד למורכב יותר, רגיש לטעויות, ודורש זמן פיתוח ודיבוג ארוכים יותר. בניית חבילה דורשת כתיבה ידנית של ביטים והקפדה על כללי פרוטוקול TCP/IP, מה שמגדיל את פוטנציאל השגיאות.

שפת Python-

בשונה מ־C השימוש ב־ Python עם ספריית Scapy סיפק לנו ממשק פשוט, קריא ונוח ליצירת חבילות רשת .יכולנו לבנות חבילת SYN בפקודה אחת, לשנות שדות בקלות, ולשלוח חבילות בקצב מתון יחסית. היתרון המרכזי הוא מהירות פיתוח גבוהה ,נגישות קלה, ותיעוד עשיר ברשת. זהו פתרון מצוין לניסויים, בדיקות והדגמות מהירות.

עם זאת Python ,סובלת מ־חולשה בביצועים –הקוד מתבצע שורה־שורה בזמן ריצה על ידי ה- interpreterמה שגורם ל־ overhead בכל שלב בהרצת התוכנית. דבר זה משפיע ישירות על מהירות שליחת החבילות ומקטין את היעילות של המתקפה.

למרות ש־Scapy מאוד גמישה, היא אינה מספקת שליטה מלאה כמו raw sockets ב־C ולא תמיד מאפשרת ביצוע ברמת ביטים נמוכה. יחד עם זאת, לפיתוח מהיר וללימוד – מדובר בכלי נוח, חזק ומספק.

בנוסף להשוואה התיאורית בין שפות התכנות, בוצעה גם השוואה מספרית של ביצועי המתקפה וההשפעה על השרת:

מדד	Python 10K	Python Regular	C
כמות חבילות SYN	1,000,000	1,000,000	1,000,000
משך שליחה כולל	558.391 שניות	10.2 שעות	7.393 שניות
זמן ממוצע פר חבילה (שניות)	0.557147 שניות	0.036 שניות	0.00000491 שניות
סטיית תקן זמן פר חבילה (שניות)	0.000558 (זמן ממוצע בין כל K10 חבילות)	1.183	0.00000449
RTT ממוצע (שניות)	0.276	1.17	0.00539
סטיית תקן RTT(שניות)	0.294	0.441	0.01658 (16.58ms)

מסקנות מספריות:

1. **קצב שליחה :** קוד ה־ C היה מהיר הרבה יותר – שלח מיליון חבילות תוך פחות מ־8 שניות, לעומת כ־9 דקות בפייתון בגרסת K10 ובגרסה הרגילה ההתקפה (או קצב השליחה) לקחה זמן של כ-10 שעות.
2. **השפעה על זמני תגובה :** למרות ש־ C הייתה מהירה, רק Timeout אחד נמדד – בגלל שהשרת קרס מהר מדי כדי לתעד יותר. בפייתון בגרסת הK10, קצב השליחה האיטי איפשר לתעד יותר מקרים של אי־תגובה. ובגרסה האיטרטיבית (פקט אחר פקט) **לא** היה ניתן לראות עומס על השרת עד כדי TIMEOUT.

3. **פיזור התגובות (סטיית תקן) :** גבוהה יותר בפייתון, מה שמעיד על אי־יציבות רשתית.

4. **אבחון קריסה :** Timeout בודד ב־ Cמעיד על קריסה מהירה של השרת שנצפתה רק פעם אחת בשל מהירות ההתקפה. בפייתון בגרסת K10 הקריסה נמשכה לאורך זמן, ונמדדו 8 timeouts , ובפייתון הגרסה האיטרטיבית לא גרם לאף קריסה, מה שניתן להסיק מכך שקצב שליחה איטי = הפשעה מזערית על מצב השרת.

סיכום כללי

במהלך פרויקט זה ביצענו הדמיה של מתקפת SYN Flood בסביבה וירטואלית מבוקרת, תוך שימוש בשתי שפות תכנות C – ו־ Python – במטרה לבדוק את השפעתן על שרת אמיתי ולנתח את ביצועי המתקפה מבחינת קצב שליחה, עומס על השרת ויכולת ניטור.

העבודה כללה:

- פיתוח של סקריפטים שתוקפים את השרת באמצעות חבילות SYN.
- שימוש בספריית Scapy בשפת Python לצורך פיתוח מהיר ונגיש.
- הרצת פינגים ממכונת ניטור חיצונית למדידת תגובת השרת (RTT) במהלך ההתקפה.

תובנות עיקריות:

- שפת C אפשרה שליחה מהירה במיוחד של מיליון חבילות תוך פחות מ־8 שניות. כתוצאה מכך, נמדד רק Timeout אחד, אך גם הוא מעיד על כך שהשרת קרס בעקבות עומס המתקפה.
- שפת Python גרסת הK10 שלחה את אותן החבילות בפרק זמן של כ־558 שניות, מה שאפשר לזהות 8 פינגים שנכשלו (Timeout) ו־ RTT ממוצע גבוה משמעותית, דבר המלמד על האטה כללית וירידה בזמינות השרת. לעומת גרסה האיטרטיבית שלקחה כ-10 שעות מה שמדגיש עד כמה פיתון אינו נועד לתקיפות איטרטיבית מסוג זה.(בשביל DOS)
- ההשוואה הכמותית בין השפות מדגישה את הפער בין ביצועים לפשטות תכנותית C :מספקת שליטה מדויקת וביצועים מרביים, אך דורשת זמן פיתוח ודיבוג גבוהים Python .לעומתה נוחה וגמישה, אך מוגבלת מבחינת יעילות הריצה.

מסקנות סופיות:

- שפת C מתאימה במיוחד למתקפות בקנה מידה גדול הדורשות שליחה מהירה מאוד של חבילות ברמת פרוטוקול נמוכה.
- שפת Python מתאימה יותר לצורכי ניסוי, הדגמה וניטור, בזכות קלות הפיתוח והקריאות הגבוהה של הקוד.
- קצב השליחה משפיע ישירות על ניתוח תוצאת המתקפה: קוד מהיר מאוד) כמו ב־ C) עלול להסתיר את הסימפטומים של קריסת שרת, בעוד קוד איטי) כמו ב־ Python) מאפשר תיעוד רחב ומעמיק יותר של התגובה.
- מדדי RTT ו־ Timeout שימשו ככלי מדידה חשוב לזיהוי מצבים של עומס או קריסה בפועל, גם מבלי להסתמך על כלים מתקדמים.

באמצעות המעבדה העמקנו בהבנת פרוטוקול TCP, רכיבי חבילת SYN והקשר הישיר בין מבנה התוכנה לבין התנהגות הרשת. הפרויקט מדגים כיצד תכנון נכון של התקפה ובחירה מושכלת של שפה משפיעים על עוצמת ההשפעה על המערכת המותקפת, ועל היכולת לנתח את תוצאות המתקפה בצורה מדויקת.

נספח A – מבנה קבצים של הדוח:

תיקיית code מכילה את קבצי סקריפט לכל מכונה:

apache –

index.html מכיל את תוכן הHTML של מכונת הAPACHE2.

attacker –

ddos.py - מממש סימולציה של מתקפת DDoS מסוג SYN Flood באמצעות שליחה סדרתית של בקשות SYN לכתובת יעד, לצורך הדמיית עומס מלאכותי על השרת ובחינת עמידותו בפני התקפות מסוג זה.

ddos10K.py - מתנהג באותה צורה כמו *ddos.*py, המערכת אוספת 10,000 חבילות בכל איטרציה (מתוך 100 איטרציות) ושולחת את כולן יחד בקריאה אחת לפונקציית send. לאחר כל קבוצה, המחושב ממוצע זמן השליחה לכל חבילה נרשם גם לקובץ טקסט וגם לקובץ CSV, דבר המספק תובנה סטטיסטית מצומצמת ובעלת עלויות נמוכות בהשוואה לשליחה חבילה בודדת בכל פעם.

ddos.c - מממש מתקפת SYN Flood ב־C באמצעות יצירת חיבורי RAW לכתובת יעד, בנייה ידנית של חבילות TCP/IP עם שדות מזויפים, שליחתן במסות גבוהות, ותיעוד זמן שליחה וסטטיסטיקות לניתוח עומסים, לצורך סימולציה ובחינה של עמידות שרתים בפני התקפת DDoS.

attack_graph_anz.py - מבצע ניתוח ויזואלי של תעבורת רשת מסוג SYN Flood באמצעות עיבוד נתוני CSV, תוך הצגת גרף המתאר את קצב שליחת חבילות ה־SYN לאורך זמן, במטרה לזהות דפוסים אופייניים להתקפת DDoS.

monitor –

monitor.py - משמש לניטור ביצועים בזמן אמת של מתקפת DDoS, על ידי ניתוח נתונים (כגון זמן שליחה או קצב תעבורה) ממקורות חיצוניים או מקבצים, לצורך הדמיה, הצגה גרפית או תיעוד של התנהגות המערכת תחת עומס רשת חריג.

Ping_graph_anz.py - מנתח תוצאות של פינגים (ping) מ־ CSVומציג גרפים של זמני תגובה (RTT) לאורך זמן, כדי להמחיש שינויים בביצועי הרשת ולאתר סימנים לפעילות חריגה כמו מתקפת DDoS או עומס יתר.

c_ddos_info ,py_ddos_info –

התקיות מכילות את כלל התוצרים של ניסוי/סימולציית SYN Flood ו-בדיקות PING שבוצעו על גבי מכונת הAPACHE2. היא כוללת נתונים גולמיים, גרפים, ותיעוד טקסטואלי המאפשרים ניתוח מעמיק של תוצאות ההתקפה. תחילית של כל תקייה מתארת את סוג הקוד שהתבצעה, c_... לשפת C ו-py_... לשפת python.