

תרחיש 3 : בניית מתקפת DDOS

1. לוגיסטיקה :

שם מגישים : רון אמסלם
פאד' נוג'ידאת

2. תהליך בניית המתקפה :

במתקפה זו ביצענו סימולציה של מתקפת SYN Flood בה התקוף שולח כמות עצומה של בקשות SYN לשרת במטרה להעיסס עליו ולגרום לו לא להיות זמין. במקביל, שלחנו פינגים מהמונייטור אל השרת כדי לבדוק האם זמן התגובה של השרת נפגעים במהלך המתקפה.

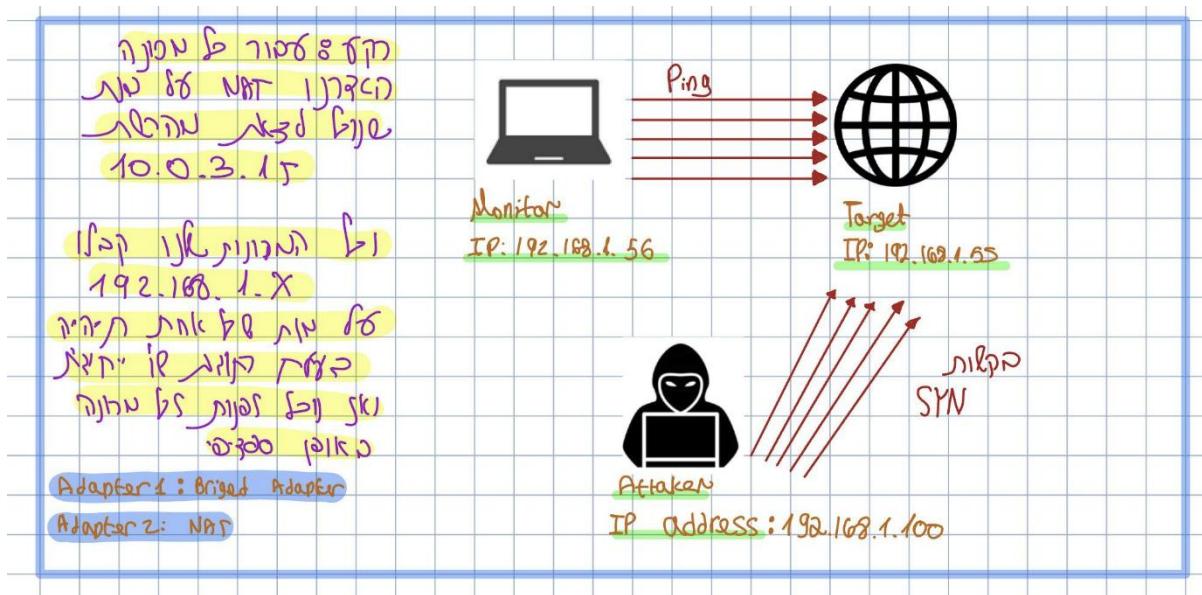
ראשית נדון בבניית המכונות שלנו

Attacker.1

Monitor.2

Target .3

להלן תרחיש שמתאר את המתרחש במתקפה שלנו הכלל כתובות IP עברו כל אחת מהמכונות.



במתקפה, אנחנו בודקים את ההשפעה על קצב שרת web , apache2 , web2 מה שבעצם קורה זה שנחננו שולחים הודעות SYN מהתקוף אל שרת web ויש לנו Monitor שמאזין ושולח פינג אל השרת שלנו, ונחננו בעצם בודקים את העיכוב בזמן בゲעת הping אל המונייטור ובכך מודדים את ההשפעה על השרת.

בתוך זה התעסכנו בלקוד שתי תוכניות אחת בשפת פיתון והשנייה בשפת C וחלק ממטרת התרחיש היא להשוות בין שתי התוכנית בפורמטים שונים כגון : סטיטית תקן , ממוצע , זמן RTT ולהציג את ההבדלים בין שימוש בשפת C לשימוש בשפת פיתון . כמו כן התבקשנו לשמר את כל המידע בקובצים על מנת שלאחר מכן נושאו בין התוצאות שקיבלו על ידי יצירת גרפים .

בסביבת העבודה השתמשנו באoxic virtual והתקנו שתי מכונות שונות על מנת למנוע תקלות טכניות

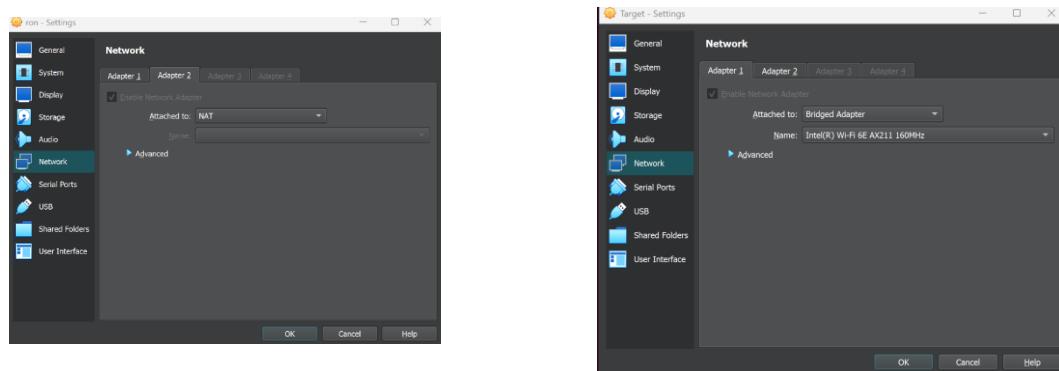


עבור המכונה השלישית השתמשנו בה מה שהיא מותקן כבר אובנטו 10.24.

על מנת ליצור סביבת עבודה מספקת מה שעשינו זה הגדרנו עבור כל מכונה את ה Network באופן הבא .
השתמשנו בNAT על מנת שנוכל לתקשר עם האינטרנט החיצוני (נדרש על מנת לגשת לשרת ה web שלו .)

במהלך המתקפה אנחנו משתמשים בשליחת חבילת SYN מצד התקוף , ובעצם אנחנו מנצלים את מגנון פתיחת הקשר של פרוטוקול TCP (שלב הראשון הוא שליחת SYN) , מהסיבה TCPsheesh הוא פרוטוקול אמין אז הוא מגיב לחיבורו , והשרת מקצה משאבים להתחברות הלוקו , אבל אנחנו לא ממשיכים בהתחברות לשרת ולכך החיבורים הנוצרים הם חיבור פתוח , אנחנו מבצעים את זה מיליון פעמים כך שבסוף דבר אנחנו גורמים לעומס רב על השרת ול��פוקד איטי ואף לקוי של השרת עבור לקוחות חדשים או לקוחות שכבר נמצאים בשירות .

השתמשנו גם ב Bridged על מנת שכל שלושת המכונות יקבלו IP ייחודי ב כדי שנוכל להבדיל ביניהם ויכלו לתקשר אחת עם השנייה
כמו שמצוין במטה :



תיקוף :

```
# netstat -an | grep brd
root@ron-VirtualBox:~$ ip a
1: ens3: <NOQUEUE,NOFCS> brd 0: flags=4163<UP,BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:1e:00:00 brd 00:00:00:00:00:00
        txqueuelen 1000 scope host brd 00:00:00:00:00:00
            valid_lifetimes forever preferred_lifetimes forever
            link-layer-peer brd 00:00:00:00:00:00
            queueingdisc fq_codel
2: ens3: <NOQUEUE,NOFCS> brd 0: flags=4163<UP,BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:1e:00:00 brd 00:00:00:00:00:00
        txqueuelen 1000 scope global noprefroute ens3
            valid_lifetimes 240s preferred_lifetimes 240s
            link-layer-peer brd 00:00:00:00:00:00
            queueingdisc fq_codel
3: veth-ron: <NOQUEUE,NOFCS> brd 0: flags=4163<UP,BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:77:53:b0:00 brd ff:ff:ff:ff:ff:ff
        txqueuelen 1000 scope global dynamic noprefroute veth-ron
            valid_lifetimes 60875ms preferred_lifetimes 60875ms
            link-layer-peer brd 00:00:00:00:00:00
            queueingdisc fq_codel
root@ron-VirtualBox:~$
```

מוניטור:

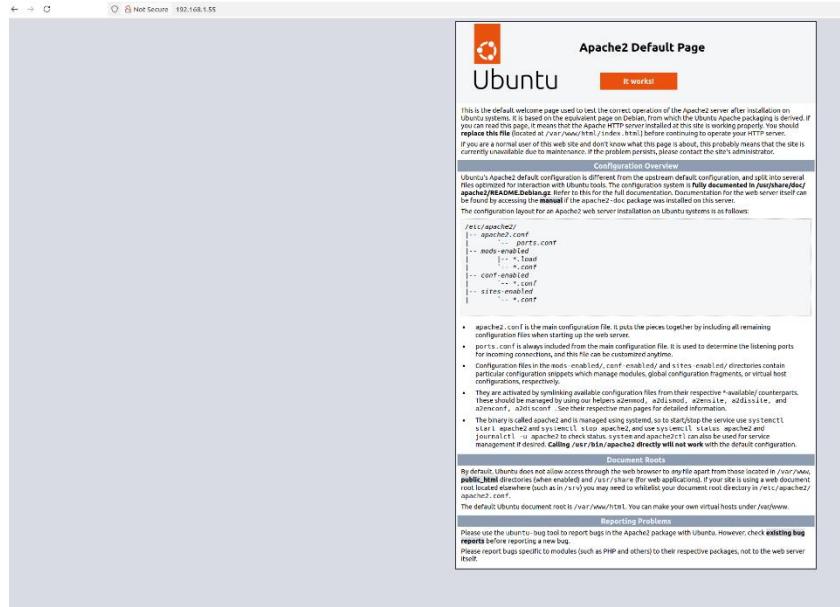
: Ty

```
bosboxes$osboxes:~$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback brd 00:00:00:00:00:00 state UNKNOWN  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: enp6s3: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 00:00:27:60:06:bd brd ff:ff:ff:ff:ff:ff  
    inet 192.168.1.55/24 brd 192.168.1.255 scope global dynamic noprefixroute enp6s3  
        valid_lft 303sec preferred_lft 303sec  
    inet6 fe80::5e0:27ff:fe60:6bd/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
3: enp6s4: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 00:00:27:60:06:bf brd ff:ff:ff:ff:ff:ff
```

כמו כן הפעילנו את השירות של apache 2 במכונת היעד שלנו:

```
osboxes@osboxes:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
  Active: active (running) since Thu 2025-04-10 10:09:33 EDT; 6h ago
    Invocation: 1/a3565d2c324cb18b334b1c3182940c
      Docs: https://httpd.apache.org/docs/2.4/
 Process: 1348 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
```

ו�ידאנו שבאמת יש גישה לשרת ה-Web שלנו:



במהלך המעבדה הרצינו מת肯定ת NY פעמיים – פעם אחת באמצעות קוד בשפת C ופעם נוספת באמצעות קוד בפייטון. בכל אחת מההרצאות מדדנו את זמני השילחה של בקשות ה- NYSAות זמני התגובה של הרשת, בעזרת פינגים שנשלחו מהמניטור.

אנו מנהים שיכולה להיות הבדל בביטויים בין שתי המשפטות – כאשר C עשויה להיות מהירה כי היא עיליה יותר מהסיבה שהיא יותר קרובה לשפט מכונה, ופיתון איטית יותר כי יש לה את תהליך התרגום שלוקח הרבה יותר זמן MC.

מתקפת SYN באמצעות קידוד בשפט C

בצד התוקף יצרנו קובץ בסיסומת C בשם attack1.c בצד הצלוי שוחח ערכינו וכתבונו קוד אשר שלוח בקשנות NYX לשרתת לנודרש. עשינו 100 איטרציות כך שבכל איטרציה נשלחות 10,000 פאקטות .

במיושם התוקף מתחזה לכתובת 192.168.1.2 ושולח הודעות ממנה.

בצד המוניטור יצרנו קובץ בסיסת C בשם `ping4.c` בעזרת הכלי `snano` ערכנו וכתבנו קוד אשר שולח בקשות `ping` לשרת כל 5 שניות.

קמפלנו את הקבצים שלנו בעזרת gcc

```
osboxes@osboxes:~$ nano ping4.c
osboxes@osboxes:~$ gcc ping4.c -o ping4 -lm
```

כמו כן התקן במכונה של ה Wireshark Target על מנת לבדוק את התעבורה במהלך המתקפה.

כאן ניתן לראות את הבקשות NYSE מצד התוקף אל כיוון השרת:

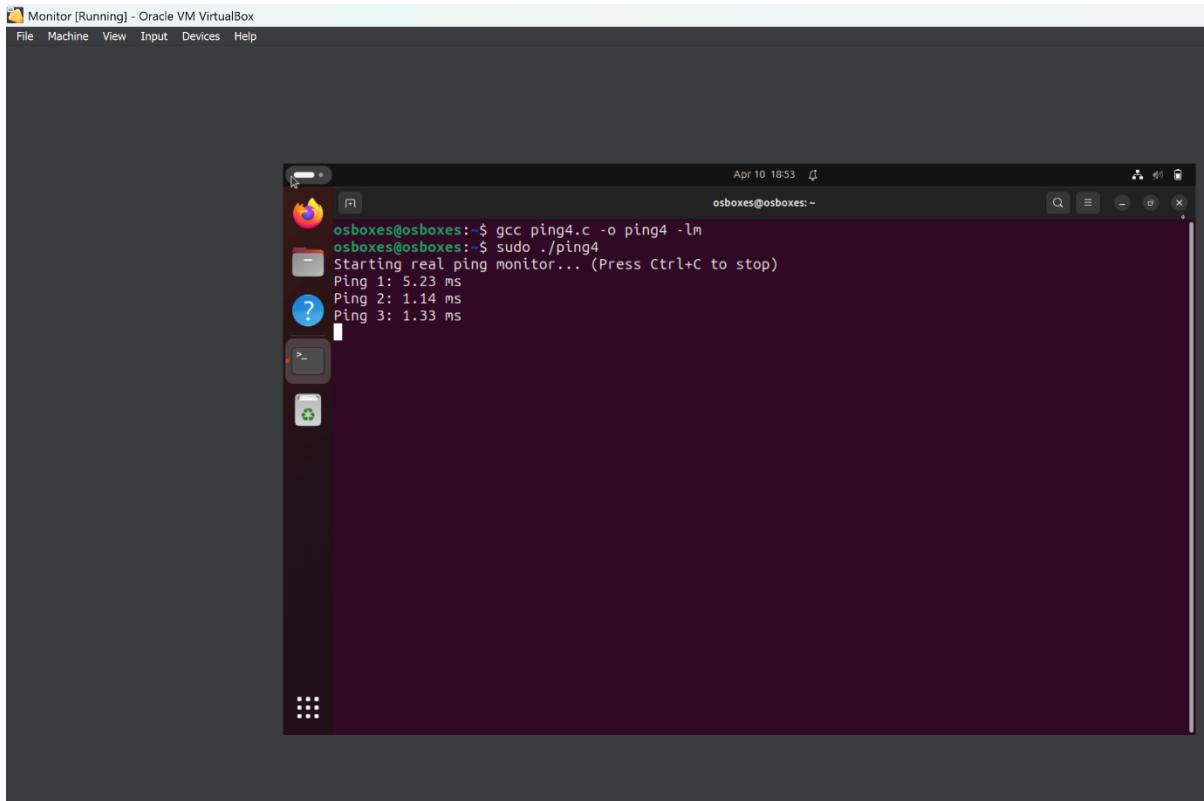
The Wireshark interface displays a list of network captures. The current capture is titled "enp0s3". The packet list shows a series of TCP retransmissions from source IP 192.168.1.2 to destination IP 192.168.1.55 on port 69. The protocol column indicates TCP, and the status column shows "Retransmission" for most of the listed packets. The "Time" column shows the timestamp for each packet. The "Source" and "Destination" columns show the respective IP addresses. The "Protocol" column is TCP. The "Length" and "Info" columns provide detailed information about the packet structure. The "Details" and "Bytes" panes are visible on the right side of the interface.

בצד המוניטור כתבו קוד אשר ישלח בקשות PING לשרת על מנת שנוכל למדוד את העומס

ip.addr == 192.168.1.56						
lo.	Time	Source	Destination	Protocol	Length	info
+	8337 2.020932146	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=4096/16, ttl=64 (r)
-	8340 2.020954457	192.168.1.55	192.168.1.55	ICMP	62	Echo (ping) reply id=0xe01a, seq=4096/16, ttl=64 (r)
+	25495 7.028855473	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=4352/17, ttl=64 (r)
-	25497 7.028873133	192.168.1.55	192.168.1.56	ICMP	62	Echo (ping) reply id=0xe01a, seq=4352/17, ttl=64 (r)
+	35044 12.035096579	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=4608/18, ttl=64 (r)
-	35045 12.035121698	192.168.1.55	192.168.1.56	ICMP	62	Echo (ping) reply id=0xe01a, seq=4608/18, ttl=64 (r)
+	44752 17.040150124	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=4864/19, ttl=64 (r)
-	44756 17.040173969	192.168.1.55	192.168.1.56	ICMP	62	Echo (ping) reply id=0xe01a, seq=4864/19, ttl=64 (r)
+	54527 22.045425736	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=5120/20, ttl=64 (r)
-	54528 22.045443912	192.168.1.55	192.168.1.56	ICMP	62	Echo (ping) reply id=0xe01a, seq=5120/20, ttl=64 (r)
+	64825 27.058342264	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=5376/21, ttl=64 (r)
-	64833 27.058367726	192.168.1.55	192.168.1.56	ICMP	62	Echo (ping) reply id=0xe01a, seq=5376/21, ttl=64 (r)
+	75058 32.063252726	192.168.1.56	192.168.1.55	ICMP	62	Echo (ping) request id=0xe01a, seq=5632/22, ttl=64 (r)
-	75059 32.063277439	192.168.1.55	192.168.1.56	ICMP	62	Echo (ping) reply id=0xe01a, seq=5632/22, ttl=64 (r)

(השעות בין המכוונות שונות)

התחלנו הראשית לשЛОח ping מהמוניטור שלנו אל השרת ב-10.4.25.18:53 בשעה



ואת המתקפה התחלנו מיד אחרי שנשלחו כמה פינגים בשעה 10.4.25 17:43

```
ron@ron-VirtualBox:~/Desktop$ sudo ./attack1
Starting attack...
Sent 10000 packets
Sent 20000 packets
Sent 30000 packets
Sent 40000 packets
Sent 50000 packets
```

המתקפה הסתיימה אחרי 5 דקות ו 34 שניות

ונוצרו לנו קבצים הבאים

מהמניטור : pings_results_c.txt בפורמט הבא :

```
1 Ping 1: 1.361000 ms
2 Ping 2: 1.237000 ms
3 Ping 3: 1.328000 ms
4 Ping 4: 0.935000 ms
5 Ping 5: 0.869000 ms
6 Ping 6: 2.863000 ms
7 Ping 7: 4.213000 ms
8 Ping 8: 1.070000 ms
9 Ping 9: 0.784000 ms
10 Ping 10: 0.786000 ms
11 Ping 11: 0.764000 ms
12 Ping 12: 0.838000 ms
13 Ping 13: 1.258000 ms
14 Ping 14: 0.378000 ms
15 Ping 15: 0.842000 ms
16 Ping 16: 6.475000 ms
17 Ping 17: 6.921000 ms
18 Ping 18: 1.325000 ms
19 Ping 19: 0.870000 ms
20 Ping 20: 3.325000 ms
21 Ping 21: 0.692000 ms
```

ובסוף ממוצע וטיטית תקן :

```

77 Ping 77: 1.696000 ms
78 Ping 78: 0.997000 ms
79
80 Average RTT: 1.640167 ms
81 Standard Deviation: 1.472180 ms
82

```

מהתוקף shus בפורמט הבא :

```

Packet 1: 0.134221 ms
Packet 2: 0.040636 ms
Packet 3: 0.030584 ms
Packet 4: 0.028601 ms
Packet 5: 0.028188 ms
Packet 6: 0.025639 ms
Packet 7: 0.029348 ms
Packet 8: 0.028133 ms
Packet 9: 0.027669 ms
Packet 10: 0.028441 ms
Packet 11: 0.030434 ms
Packet 12: 0.149167 ms
Packet 13: 0.071998 ms
Packet 14: 0.045793 ms
Packet 15: 0.029007 ms
Packet 16: 0.026618 ms
Packet 17: 0.026536 ms
Packet 18: 0.025767 ms
Packet 19: 0.025657 ms
Packet 20: 0.028334 ms
Packet 21: 0.027216 ms

```

בסוף זמן הכלול, ממוצע, וסטיית תקן

```

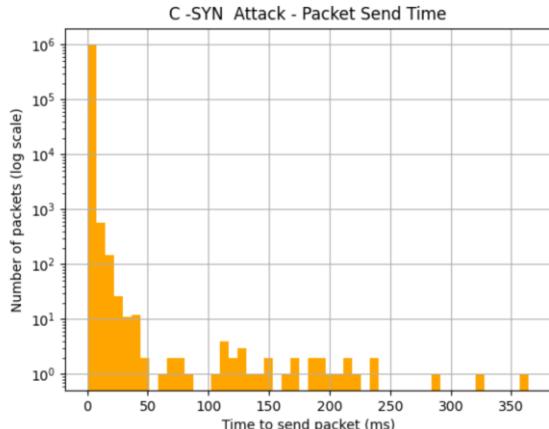
Total time: 373.05 seconds
Average time: 0.372858 ms
Standard Deviation: 1.183974 ms

```

מהנתונים הסטטיסטיים עולה כי הזמן הממוצע לשילוח חבילה NYU עומד על כ- 0.37 מילישניות בלבד, דבר המעיד על קצב שליחה גבוה מאוד. עם זאת, סטיית התקן שעמדה על כ- 1.18 מילישניות מצביעה על כך שלא כל הבקשות נשלחו באותו הקצב – חלקן לקחו משמעותית יותר זמן. פיזור זה מתבטא גם בגרף ההיסטוגרמה, בו ניתן לראות בקשות בודדות שהתעכבו.

גרפים :

גרף ראשון המבטא את - זמן שליחה של בקשות SYN



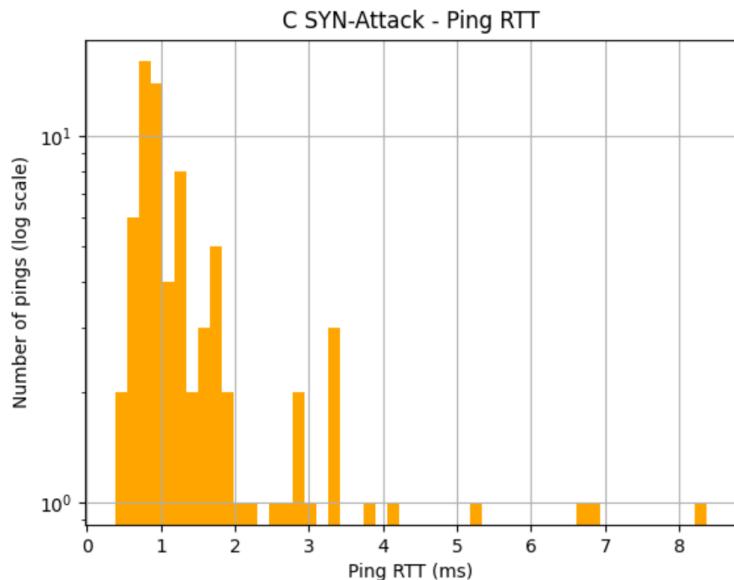
הגרף מציג כמה זמן לשלוח כל חבילה SYN מהתוקף אל השרת.

בציר X הזמן לשילוח חבילה אחת (ב밀ישניות).

בציר Y מספר הפקודות שנשלחו באותו הזמן זהה (בסקללה לוגריתמית).

התפלגות מראה שרוב החבילות נשלחו מהר, עם شيئا גודל סביבה 0–5 ms . אבל יש גם צב אורך של חבילות שנשלחו בצורה איטית יותר – חלק אפילו מעל 300 ms זה נראה קרה בגלל עומס נקודתי במערכת, שגורם לעיכובים קטנים.

גרף שני המבטא את זמן תגובה של השרת (RTT)



הגרף מציג את הזמן RTT של הפינגים ששלח המוניטור במהלך התקיפה.

בציר X זמן תגובה של פינג – במלישניות.

בציר Y כמה פינגים התקבלו עם זמן זהה (בסקללה לוגריתמית).

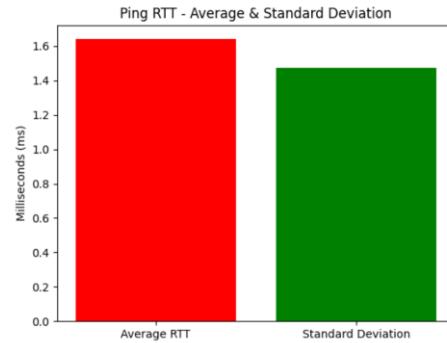
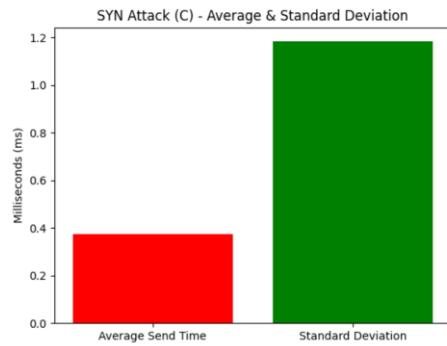
רוב הפינגים התקבלו בזמן RTT נמוך לעומת השרת מגיב מהר.

חלק קטן מהפינגים לקח להם יותר זמן.

אפשר לראות שההתפלגות מאוד מרכזת בין 0.5 ל-2.5 מילישניות, עם שיא סביר 1 מילישניות לעומת המוחלט של הפינגים קיבלו תגובה ממש מהירה. יש כמה ערכים בודדים שמייצגים גם ל-47 עד 8 מילישניות אבל הם נדירים ולא משפיעים הרבה. בסה"כ נראה שהשתתת הגיב בצורה מאוד יציבה ומהירה לאורך כל התקופה.

מהגרפים אפשר לראות שרוב החבילות נשלחו ממש מהר, כמעט כולם תוך פחות מ-5 מילישניות. גם הפינגים שקיבלו מהמונייטור בזמן ההתקפה נשארו עם זמן תגובה נמוך יותר – כמעט כולם בין 0.5 ל-2.5 ms זה אומר שהעומס שהתקפה יצרה על השרת לא השפיע כמעט בכלל על יכולת שלו לענות לפינגים. למרות שלחנו מיליון חבילות, נראה שהשליחה הייתה מספק יציבה ומהירה כדי לא להעמש יותר מדי על המערכת.

במהר לנוטונים הסטטיסטיים :



הגרף הראשון ממחיש את הטעות בין הזמן הממוצע לשילוח חבילה SYN לבין סטיית התקן של זמן השילחה.

ניתן לראות שהממוצע נמוך מאד, אך סטיית התקן גבוהה יחסית כ-1.18, מה שמעיד על פיזור רחב בזמן השילחה. ככלומר, בזמן שרוב הביקשות נשלחו במהירות, חלק קטן מהן התעכב.

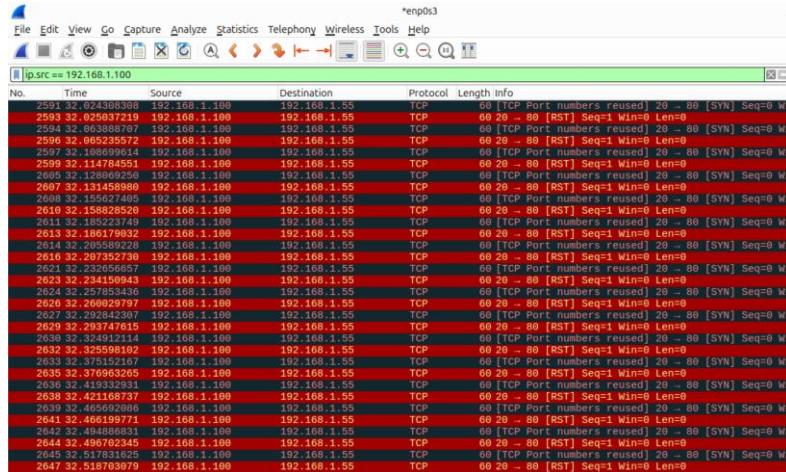
בגרף של הפינגים – הממוצע יחסית נמוך, אבל סטיית התקן כמעט זהה אליו. זה מראה שהתקפה לא השביבה את השרת, אבל כן הקצת השפעה לא יציבה על התגובה לפינגים.

מתקפת SYN באמצעות פיתון

בצד התקוף יצרנו קובץ בסיסו מוקד בשם attack5.py בצד הלקוח הcli שוחח ערכנו וכתבנו קוד אשר שלוח בקשות SYN לשרת כנדרש עשוינו 100 איטרציות כך שבכל איטרציה נשלחות 10,000 פאקטות .

בצד המוניטור יצרנו קובץ בסיסו מוקד בשם c6ping.py בצד הלקוח הcli שוחח ערכנו וכתבנו קוד אשר שלוח בקשות ping לשרת כל 5 שניות .

ניתן לראות בWireshark הנמצא במכונה של היעד , עקב בקשות התקוף :



כפי שניתן לראות, כמעט כל החבילות הן בקשות TCP SYN ב프וטוקול TCP הנשלחות אל השרת , ניתן להבין מהו שמתיקיימת מתקפת SYN Flood שבה התקוף שלוח כמותות גדולות של בקשות פתיחת חיבור מוביל להשלים את התהילהן.

ובקשות המוניטור:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x043f, seq=1/256, ttl=64 (rep)
8	5.034890340	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0xbabc1, seq=1/256, ttl=64 (rep)
13	15.550219173	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x3993, seq=1/256, ttl=64 (rep)
15	15.559576546	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x65d0, seq=1/256, ttl=64 (rep)
22	20.589838697	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x4565, seq=1/256, ttl=64 (rep)
36	25.615206645	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0xc299, seq=1/256, ttl=64 (rep)
42	30.637559362	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0xe1b2, seq=1/256, ttl=64 (rep)
45	35.653425361	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0xd5af, seq=1/256, ttl=64 (rep)
49	40.691636706	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0xf8de, seq=1/256, ttl=64 (rep)
53	45.714781772	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x18c9, seq=1/256, ttl=64 (rep)
57	50.729402254	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x8947, seq=1/256, ttl=64 (rep)
62	55.863183977	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x05cf, seq=1/256, ttl=64 (rep)
67	60.912841870	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x6a59, seq=1/256, ttl=64 (rep)
71	65.931945244	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x9811, seq=1/256, ttl=64 (rep)
73	70.943928140	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x1772, seq=1/256, ttl=64 (rep)
76	76.191852064	192.168.1.56	192.168.1.55	ICMP	98	Echo (ping) request id=0x60a2, seq=1/256, ttl=64 (rep)

קודם כל התחלונו לשלוח קצר פינגיים ממוכנת המוניטור בשעה 10:20 ב-10.4.25

```
osboxes@osboxes:~$ sudo python3 ping6.py
Ping 1: 1.53 ms
Ping 2: 1.53 ms
```

از התחלונו את המתקפה בשעה 17:22 ב-10.4.25

```
ron [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
ron@ron-VirtualBox:~$ sudo python3 attack5.py
Starting attack...
```

והמתקפה נגמרה ב- 23:22 6 שעות

```
ron [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
ron@ron-VirtualBox:~$ sudo python3 attack5.py
Sent 790000 packets
Sent 800000 packets
Sent 810000 packets
Sent 820000 packets
Sent 830000 packets
Sent 840000 packets
Sent 850000 packets
Sent 860000 packets
Sent 870000 packets
Sent 880000 packets
Sent 890000 packets
Sent 900000 packets
Sent 910000 packets
Sent 920000 packets
Sent 930000 packets
Sent 940000 packets
Sent 950000 packets
Sent 960000 packets
Sent 970000 packets
Sent 980000 packets
Sent 990000 packets
Sent 1000000 packets

Attack completed successfully!
Total time: 21461.79 seconds
Average packet send time: 21.27 ms
Standard Deviation: 27.81 ms
ron@ron-VirtualBox:~$
```

ונוצרו לנו קבצים הבאים

מהמניטור : pings_results_p.txt בפורמט הבא

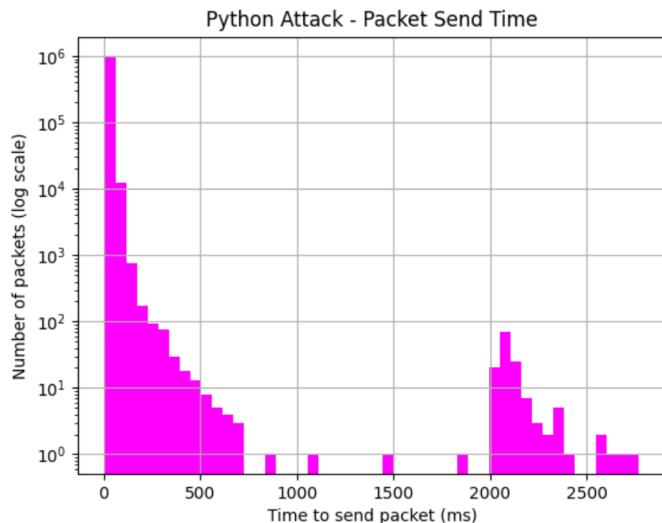
	ובסוף כל
4204	Ping 4250: 1.21 ms
4205	Ping 4251: 0.73 ms
4206	Ping 4252: 0.76 ms
4207	Ping 4253: 1.89 ms
4208	Ping 4254: 1.59 ms
4209	Ping 4255: 1.08 ms
4210	Ping 4256: 1.50 ms
4211	Ping 4257: 0.90 ms
4212	Ping 4258: 0.83 ms
4213	
4214	Average RTT: 1.58 ms
4215	Standard Deviation: 15.50 ms
	Ping 1: 1.53 ms
	Ping 2: 0.82 ms
	Ping 3: 1.12 ms
	Ping 4: 1.41 ms
	Ping 5: 1.94 ms
	Ping 6: 7.60 ms
	Ping 7: 1.43 ms
	Ping 8: 2.41 ms
	Ping 9: 1.77 ms
	Ping 10: 2.51 ms
	Ping 11: 6.10 ms
	Ping 12: 1.19 ms
	Ping 13: 1.81 ms
	Ping 14: 1.68 ms
	Ping 15: 0.86 ms
	Ping 16: 0.58 ms

מהתוקף : `sys_results_y.txt` בפורמט הבא :

Packet 999998: 23.243 ms	Packet 1: 95.724 ms
Packet 999999: 19.546 ms	Packet 2: 35.965 ms
Packet 1000000: 23.069 ms	Packet 3: 29.921 ms
 	Packet 4: 28.024 ms
Total time: 24910.18 seconds	Packet 5: 21.890 ms
Average packet send time: 24.587 ms	Packet 6: 23.803 ms
Standard Deviation: 27.630 ms	Packet 7: 24.096 ms
	Packet 8: 32.305 ms
	Packet 9: 35.687 ms
	Packet 10: 33.245 ms
	Packet 11: 32.421 ms
	Packet 12: 19.638 ms
	Packet 13: 37.049 ms
	Packet 14: 22.022 ms
	Packet 15: 33.478 ms
	Packet 16: 37.713 ms

גרפים:

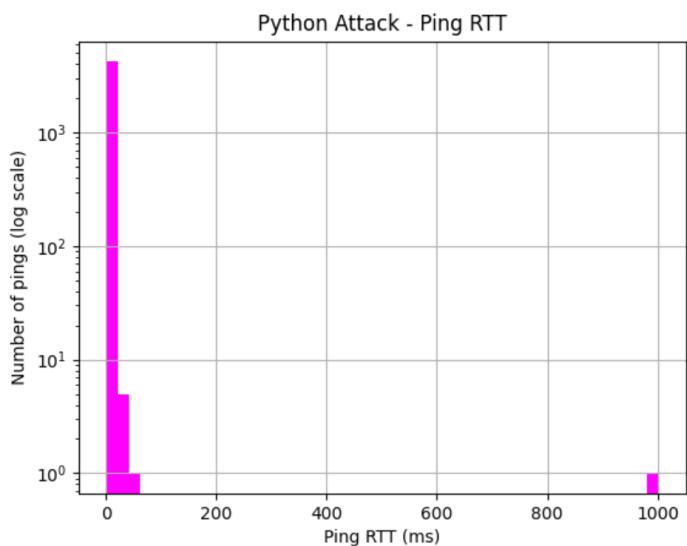
גרף הראשון : קצב שליחת SYN מצד התוקף



הgraf מציג את זמני השילחה של חבילות SYN במהלך המתקפה שבוצעה בפייתון. כל עמודה מייצגת את מספר החבילות שנשלחו באותו זמן מסוים. מרבית הבקשות שנשלחו בתוך פחות מ-100 מילישניות – דבר המעיד על מערכת תקינה בתחילת המתקפה. עם הזמן, ניתן לראות האטה הדרגתית, ולאחר מכן הופעה של קבוצת חבילות בודדות שנשלחו באיחור קיצוני של למעלה מ-2 שניות. הופעה זו מצביעה על עומס שהולך ונבנה במהלך התקוף.

התפלגות ממש רחבה ולא איחודה. רוב החבילות שנשלחו יחסית מהר, אבל ככל שהזמן עבר, לוקח יותר ויותר זמן לשולח חבילה. אפילו יש חבילות שלקח להן יותר מ-2.5 שניות להשלוח זה מראה שהמערכת התחליה טוב, אבל לאט לאט נחשפה לעומס על השרת.

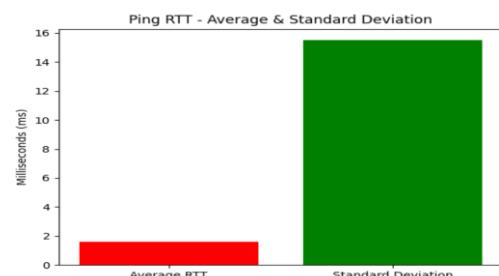
גרף שני : המביטה את זמן תגובה של השרת (RTT)



הגרף מציג את زمنי ה-RTT של פקודות ping לשרת במהלך המתקפה. בציר האופקי מופיע זמן התגובה של כל ping (ב밀ישניות), ובציר האנכי מספר הפינגים שנמדדו בזמן זהה. רוב הפינגים קיבלו תשובה מהירה, בזמן של פחות מ-20 ms אבל רואים גם פינגים בודדים שהגיעו לזמן תגובה של כמעט 1000 ms המסבירו מדוע הרשות המשיך להשיב לפינגים, המתקפה כן השפיעה על הביצועים שלו, כי ברגעים מסוימים נוצר עיכוב משמעותי בתגובה.

כל זמן שליחת החבילות עלה – בגלל עומס פנימי של הקוד או עומס שנוצר על השרת – כך גם נראה עיכובים בתשומות הפינג מהמניטור. שני הגרפים יחד מראים שמערכת ההתקפה התחילה לעבוד מהר, אבל עם הזמן נכנסה לעומס כבד, מה שגרם גם להאטיה בשילוחה וגם לתגובה איטית מהשרת. זה מעיד על הידדרות בBITSים לאורץ זמן, שנitinן לזמן גם מהשליחה וגם מהתגובה של השרת.

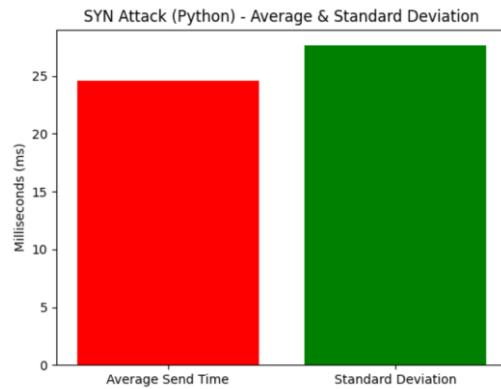
אנחנו יכולים להסתכל על הנתונים הסטטיסטיים שמופיעים בקבצי פלט שלנו להבין כי ניתוח סטטיסטי מהצד של התקוף : מדובר בזמן שליחת ממוצע נמוך יחסית, שਮuid על כך שהסקיריפט מצילח לייצר תעבורת ומהירות גבוהה. עם זאת, סטיית התקן הגבואה מעידה על שונות גדולה בין זמני השילוח כלומר, למرات שרוב החבילות נשלחו מהר, חלון נתקלו בהאטיה. הדבר עשוי לעומס במערכת התקופה או במשക הרשת שלה.



ניתוח סטטיסטי מהצד של המוניטור : מדובר בממוצע RTT נמוך מאוד, אך סטיית התקן גבוהה מאוד ביחס לממוצע. זה אומר שרוב הפינגים הגיעו מהר מאוד, אבל היו מקרים בודדים שבום ה-RTT-עליה מאוד.

הממוצעים הנמוכים יכולים להטעות ולגרום לחושב שלא הייתה השפעה, אבל דווקא סטיית התקן הגבואה מצביעת על חוסר יציבות ברשת – זהה בדיקת SYN Flood מה שמתќפת.

השרת לא קרס, אבל חוות ירידות רגניות בBITSים שמתעדות בגרפים ובסטטיסטיות.



הממצאים הנמורים עלולים להטעות וליצור תחושה שהשרת לא השפיע מהתקפה, אבל סטיות התקן הצביעו מבהירות שכן הייתה **פגיעה ביציבות**. זו בדיקת המטריה של מתקפת – SYN Flood או דואקן להפיל את השרת, אלא לגרום לו להתאמץ, לאבד עקביות, ולהציג לאט מדי פעם. השרת לא קרט, אבל בהחלט סבל מירידות רגניות בביטויים זהה מתועד היבט בגרפים ובמדדים.

השוואה בין C לפיתון כפי שידוע, שפת **C** היא שפת תכנות ברמת Low-Level קרובה מאוד לשפת מכונה, מה שמאפשר לה **LAGSHET YISIROT LIZICRON**, לנחל משאבים בצורה מדויקת, ולבצע פעולות בקצב מהיר במיוחד. לעומת זאת, **PYTHON** היא שפת High-Level שמתמקדת בנוחות קידוד אבל באה על חישוב **BYUTUVIM** ו-**DINHOL YUIL SHL** משאבים.

מההשוואה שנעשתה, אפשר לראות בבירור שהתקפת NYSE שנכתבה ב- C הייתה **YILHA MASHMOUTIT**:

- **זמן השיליחה הממוצע של חבילה** היה נמוך ביותר כ- 0.37 ms והמערכת הצלילה לשלווח מיליאן חבילות בתוך כמה דקות.
- **זמן התגובה של השרת** נשארו ייצבים לאורך כל ההתקפה, ללא עיכובים מורגשים.

לעומת זאת, פיתון:

- **דרשה מעל 24,000 שניות** (6 שעות) לשלווח את אותה כמות של חבילות.
- **הציגה סטיות תקן גבוזות**, שמעידות על **chosor yicvot b'shilicha** ובתגובה.
- **יצרה עומס מORGASH על השרת**, שהתבטא בפינגים עם RTT חריגים – עד כדי 1000 ms

פיתון נוכה כשבוגרים קוד, אבל מהתוצאות שראינו היא לא מתאימה למטרות שדורשות מהירות כמו העמסת שרת בתקנות , בזמן ש- C הצלילה לשלווח מיליון חבילות לפחות מ- 400 שניות, לפיתון לוקח הרבה יותר זמן עם המונע עיכוביים לאורך הדרכ. גם הפינגים מהמניטור חיזקו את זה ב- C הם נשארו ייצבים, ובפיתון ראיינו **KFICZOT LA ZFOVOT**.