**A02 - IP/ICMP Attacks Lab**

כתובות IP לכל מחשב

|  |  |  |
| --- | --- | --- |
| Name | IP | MAC |
| Attacker | 10.0.2.100 | 08:00:27:36:00:22 |
| Client | 10.0.2.4 | - |
| Server | 10.0.2.6 | 08:00:27:7D:E7:9A |

**Task 1: IP Fragmentation**

* מבוא:
  + תיאור

במשימה זו נרצה לבצע חלוקת פקט למספר חלקים ולבצע בדיקות שונות כגון כגון שליחת פקטה גדולה מהגודל המקסימלי, חפיפה והתקפת DoS

תמונה שמכילה תרשים

התיאור נוצר באופן אוטומטי

* + מטרה

להכיר תכונות נוספות של פקטת IP וכיצד ניתן לחלק אותן וכיצד ניתן לבצע התקפות באמצעותן

* + תוצאה מצופה

נראה כיצד מבצעים התקפות עם פקטות מחולקות.

Task 1.a: Conducting IP Fragmentation

* ביצוע המשימה:

לצורך המשימה נגדיר את מחשב CLIENT ומחשב ATTACKER שמחוברים באותו הרשת.

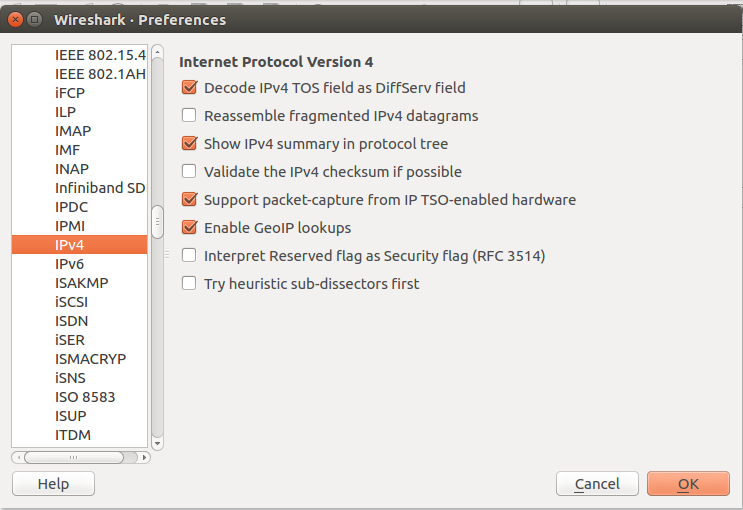
נאזין בשרת לפורט 9090 בתקשורת NETCAT בפרוטוקול UDP בעזרת הפקודה הבאה: nc -lu 9090

nc = אמצעי לתקשורת בין מחשבים בלינוקס או יוניקס (NETCAT)

-l = האזנה

U = פרוטוקול UDP

לפני שנשלח את הפקט ב3 חלקים, נבטל בהגדרות הWIRESHARK ממחשב התוקף את אופציית Reassemble fragmented IPv4 field



ביטלנו כדי שנוכל לראות את חלקי הפקטות ולא רק את הפקטה השלמה מאחר שהוא מאחד את הפקטות ורק לאחר מכן מציג אותה.

הרצנו את הקוד הבא ליצירת הפקט וחלוקה ל3 fragments:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הfragment הראשון מכיל את הheader של ההודעה, גודלו, פרוטוקול, מספר מזהה, אופסט שווה 0 כי הוא החתיכה הראשונה של המידע, דגלון שיהיו פקטות נוספות כאשר לא אכפת לנו מגילוי שגיאות ולכן chksum שווה 0.

תמונה שמכילה תרשים

התיאור נוצר באופן אוטומטי

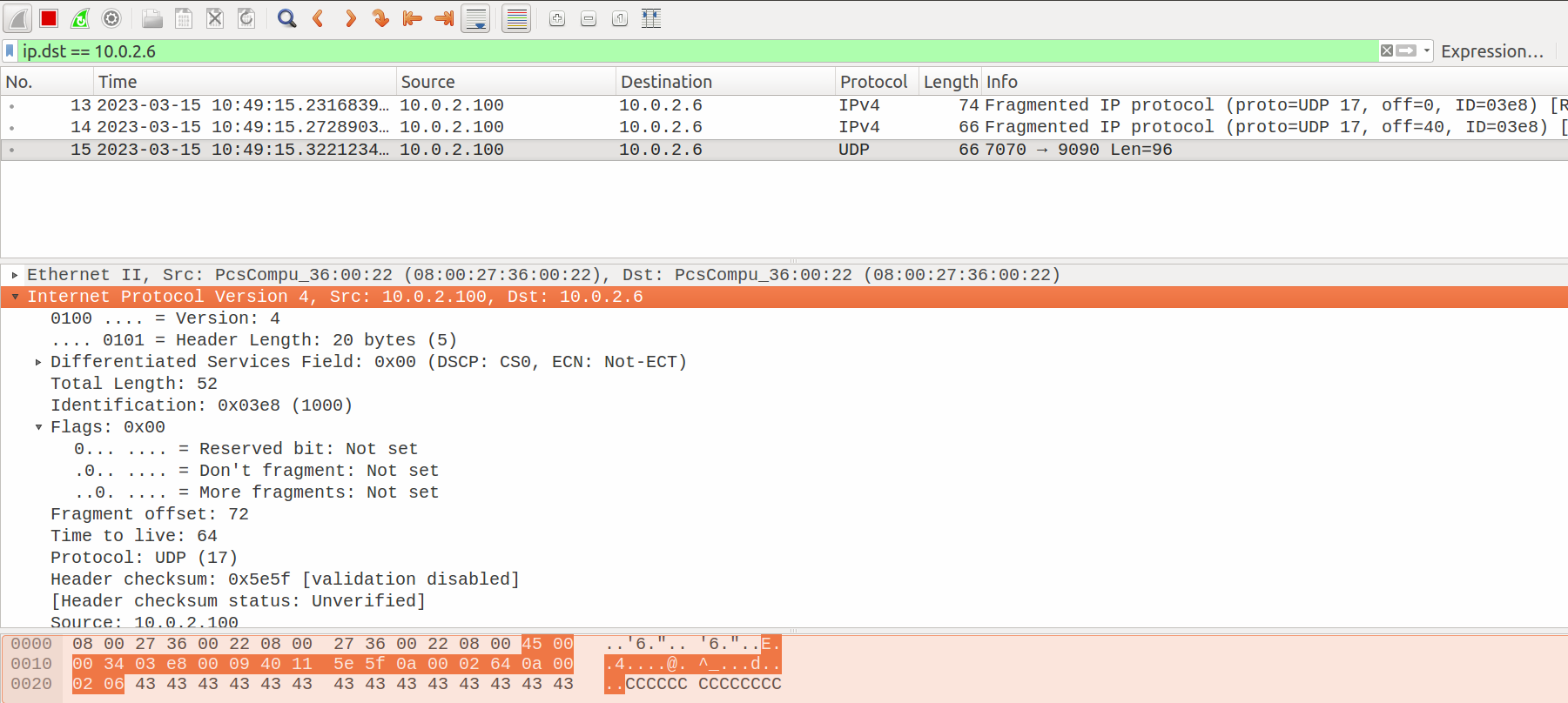
בfragment השני האופסט יתחיל מהתא אחרי האחרון של המידע מהחתיכה הראשונה כלומר 5, הגדרנו שהפקט ישלח בפרוטוקול UDP ע"י id.proto = 17.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

בfragment השלישי האופסט יתחיל מהתא אחרי האחרון של המידע מהחתיכה השנייה כלומר 9, הגדרנו שהפקט ישלח בפרוטוקול UDP ע"י id.proto = 17 ושינינו את הדגלון שיציין שהוא האחרון.

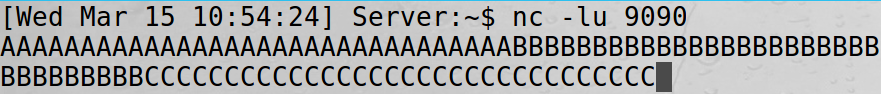
לאחר שליחת הפקט שיצרנו קיבלנו את התוצאה הבאה:





ניתן לראות שהצלחנו את המשימה וקיבלנו את שלושת הfragments כאשר ניתן לראות גם את הנתונים שהגדרנו בwireshark.

כמו כן, קיבלנו את הפלט של הDATA ששלחנו בשרת בפורט שהאזנו לו.



שאלות שעלו לנו בזמן הביצוע:

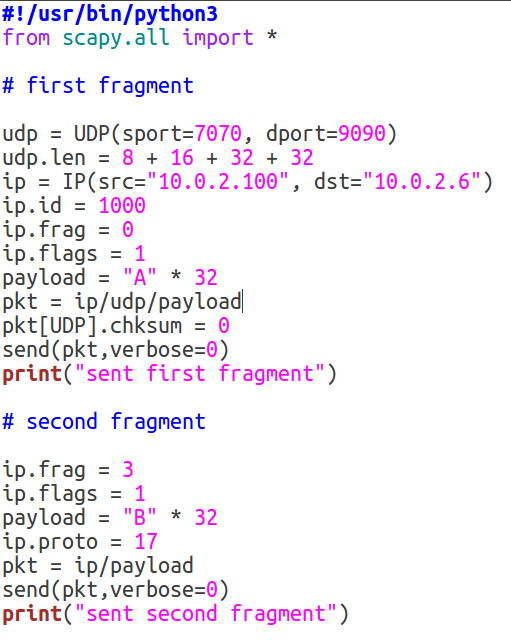
* מה יקרה אם נשלח payload ריק – לא נקבל פלט בשרת
* איך חישבנו את הoffset – חילקנו כל פקטה לבתים לדוגמה הפקט הראשון 40 ביטים שזה 5 בתים אז הoffset יהיה מ0 עד 4 וכך הלאה.

Task 1.b: IP Fragments with Overlapping Contents

* ביצוע המשימה:

המשכנו את הקוד מהסעיף הקודם כך שיתאים למקרים הבאים שנדרשנו לבדוק וביצענו את השלבים להאזנה בשרת מהסעיף הקודם:

* מקרה 1: נרצה לבדוק מה יקרה אם k בתים של fragment השני יחפוף עם k בתים של הfragment הראשון (בחרנו k=2).

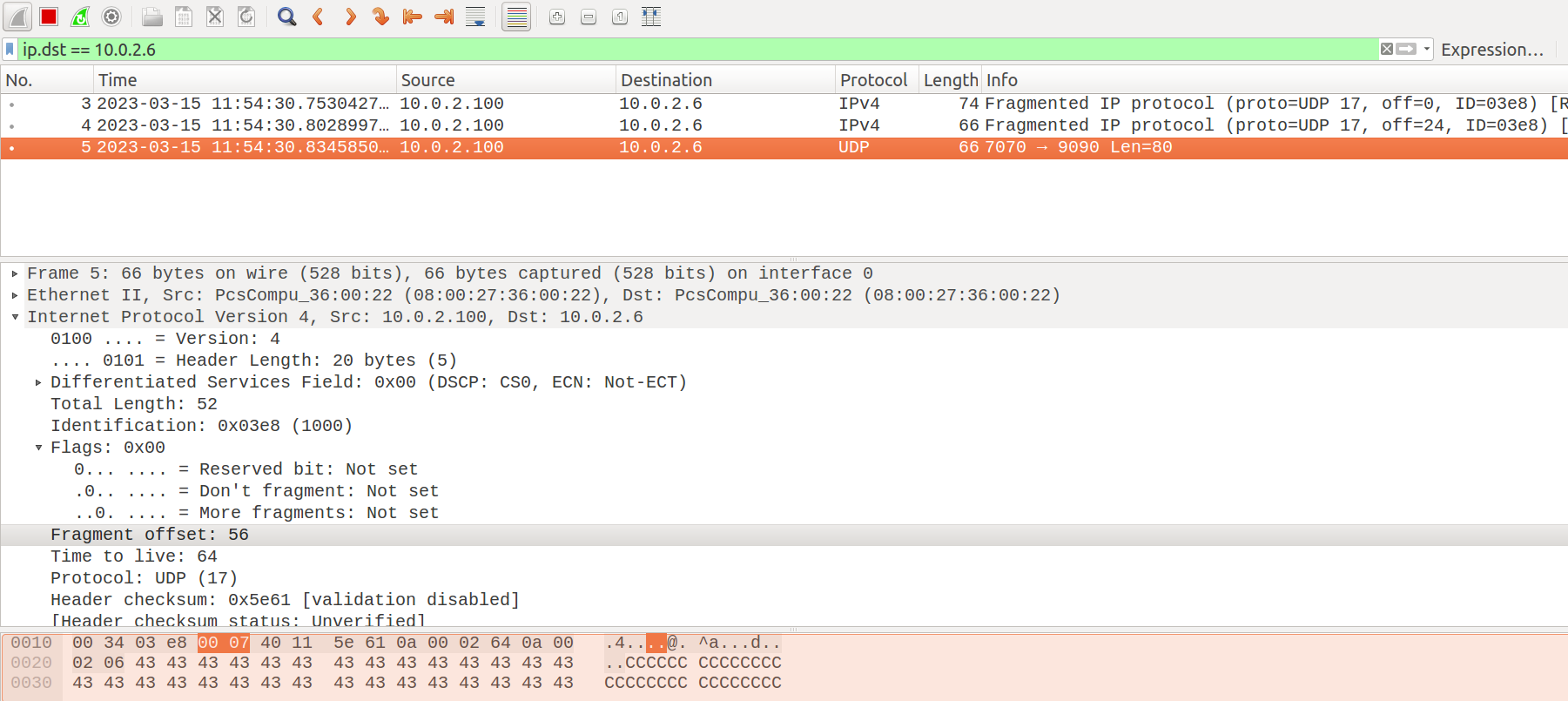
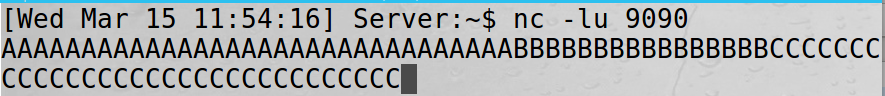


מאחר וקיימים 2 בתים חופפים וכל בית שווה 8 ביטים נדרשנו להוריד מגודל הפקט הכללי 16 ביטים (8+16+32+32) ואת הoffset של הfragment השני שיתחיל מהבית השלישי. ועדכנו גם את הfragment השלישי שימשיך מהסוף של השני, כלומר offset שווה 7.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שלחנו את הפקט המעודכן ובדקנו בwireshark שאכן הoffset של הfragments התעדכן

  ניתן לראות שהצלחנו לבצע חפיפה מפני שכמות הB שהתקבלה היא 16 ולא 32 כמו ששלחנו כי A כבר היה כתוב בזיכרון.

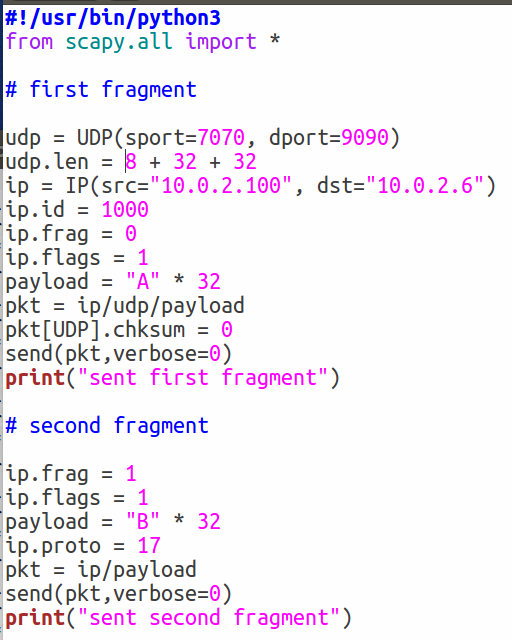


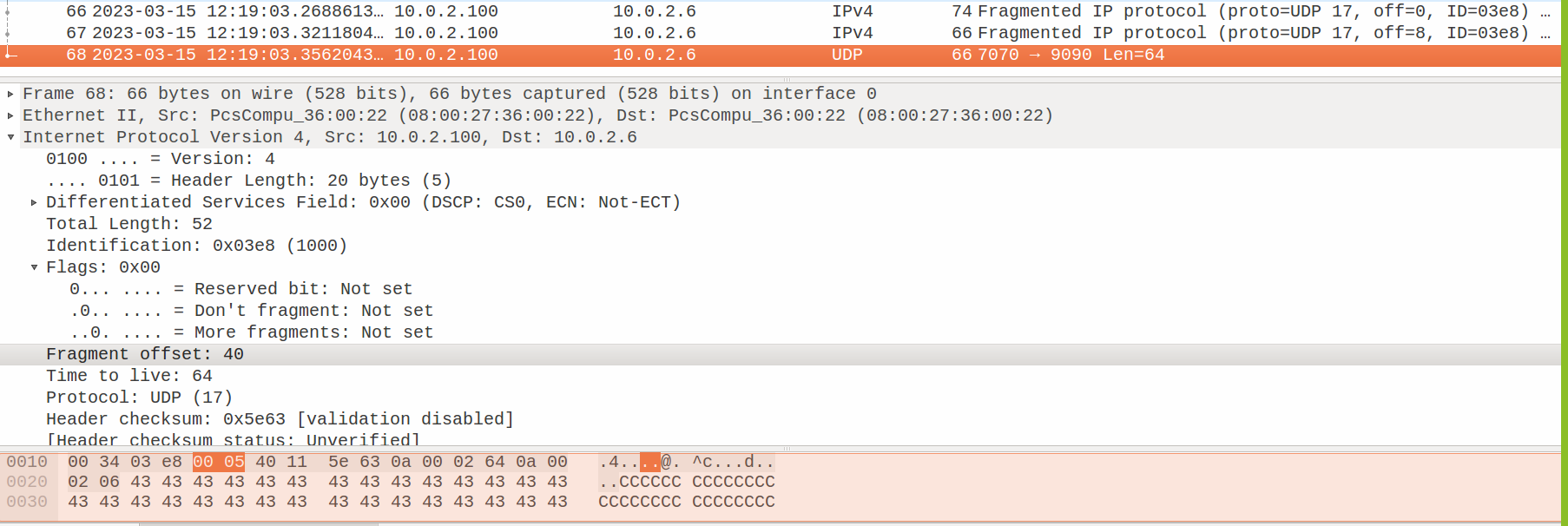
שאלנו את עצמנו למה B לא נכתב במקום A אבל אחרי חקירה ממושכת גילינו שאופן הפעולה של מקבל הפקט, הוא שהמידע הראשון שנכתב בזיכרון (A) לפי האופסט נשמר ואם מגיע פקט נוסף שמבקש לדרוס את הנתונים באותו המיקום הוא לא יתקבל, אלא נקבל את המידע בבית הראשון לאחר הבתים החופפים ולכן B ירשם רק בסוף הפקט הראשון שכתב A.

אין חשיבות לסדר הפקטות שמגיעים מפני שהם מורכבים בסוף ומסודרים לפי האופסט.

* מקרה 2: נרצה לבדוק מה יקרה אם 2 fragments יהיו חופפים לגמרי.

ביצענו את ההתאמות הנדרשות כדי שfragment 2 יהיה חופף כולו על fragment 1 בכך ששמנו לו offset שווה 1 (שזה המקום 0 בDATA) כמו כן הפקט השלישי יתחיל מאופסט 5 אחרי fragments 1+2 ועדכנו את גודל הפקט בהתאם מפני שאנחנו מצפים ל32 בתים חופפים  
שמנו לב שאם לא נעדכן את הגודל נקבל שגיאה בWIRESHARK גודל לא פקט לא מתאים.

 תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי ניתן לראות בWIRESHARK שקיבלנו את שלושת הפקטות עם האופסטים המעודכנים שלהם כך שfragment 1+2 חופפים ונצפה לפלט של 64 בתים ללא האותיות B. תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

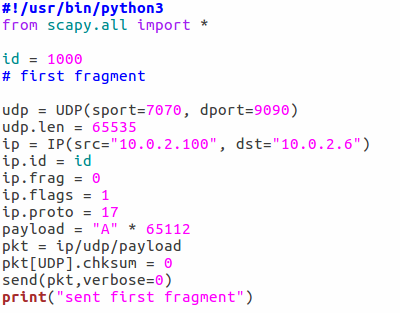
ניתן לראות שהצלחנו במשימה מאחר והשרת קיבל את הנתונים ללא המידע של הfragment השני (אין B).

Task 1.c: Sending a Super-Large Packet

* ביצוע המשימה:

במשימה נדרשנו לשלוח פקט שגדול מהמקסימום האפשרי שהוא 2 בחזקת 16 כלומר 65536 ביטים.

שינינו את הגודל של הפקט ואת האופסט המתאים בכל fragment ובכל פקט שלחנו כמות גדולה של DATA כדי לעבור את המקסימום.

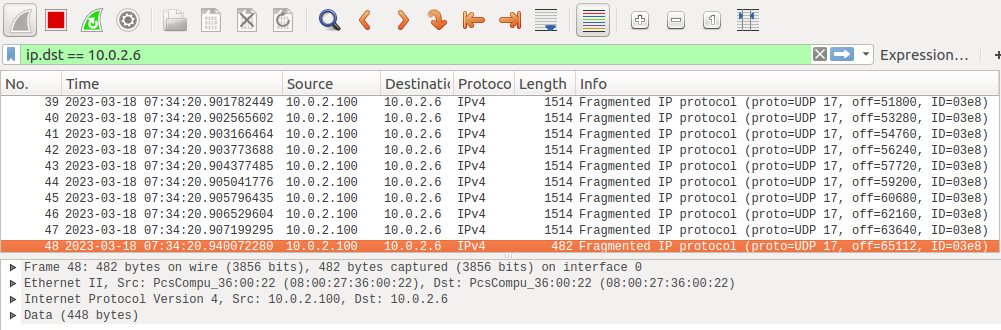


תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שמנו בפקט הראשון נתונים בגודל של 65112 בתים ובפקט השני שמנו 440 בתים כך שבסך הכל אנחנו שולחים יותר מהמקסימום המותר בנוסף להדרים שמצטרפים לfragments.

ניתן לראות בWIRESHARK שעברנו את הכמות המותרת והפקט פוצלה ליותר fragments וקיבלנו הודעה שעברנו את המקסימום האפשרי.





השרת לא מציג כלום מפני שגודל פקטת הUDP גדולה מהמקסימום ובגלל השגיאה לא נקבל פלט ולכן הצלחנו במשימה ליצור פקט ענק.

כשבנינו את הפקט יצאנו מנקודת הנחה שכמות הfragments יהיו לפי כמה שחילקנו אותם אך בפועל קיבלנו הרבה יותר fragments.

גילינו שיש ערך שנקרא Maximum Transfer Unit שתלוי בהגדרות הרשת וברוב המקרים עומד על 1500 בתים ולכן הפקט חולק להרבה יותר fragments באופן אוטומטי עוד לפני שנשלח.

ספרנו את כמות המידע שעבר וראינו שכל פרגמנט שחולק מהפרגמנט הראשון היה בגודל 1514 בתים ונשלחו 44 חלקים כאלו סה"כ 66616 בתים ועוד הפרגמנט האחרון בגודל 482 בתים כך שעברנו את המקסימום האפשרי.

Task 1.d: Sending Incomplete IP Packet

* ביצוע המשימה:

נרצה במשימה זו לבצע התקפת DOS באמצעות שליחת מלא פקטות אשר מחולקות לfragments כאשר אנחנו מחסירים fragment אחד ודבר זה גורם לעומס על זיכרון השרת ופגיעה בביצועים שלו.

כדי לבצע זאת אנחנו נרצה לשלוח בלולאה פקטות רבות אבל את הfragment השני של כל פקט לא נשלח כדי שהשרת ישמור בזיכרון את הפקטות ויחכה לfragment החסר עד שהTTL ייגמר ויזרוק את הפקט וישלח הודעת ICMP TTL כדי שנדע שהפקט נזרק.

התחלנו בכך ששלחנו פקטות רבות וניסינו להגדיל את גודל הDATA בכל פקט כדי שנוכל לראות שינוי בזיכרון של השרת.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

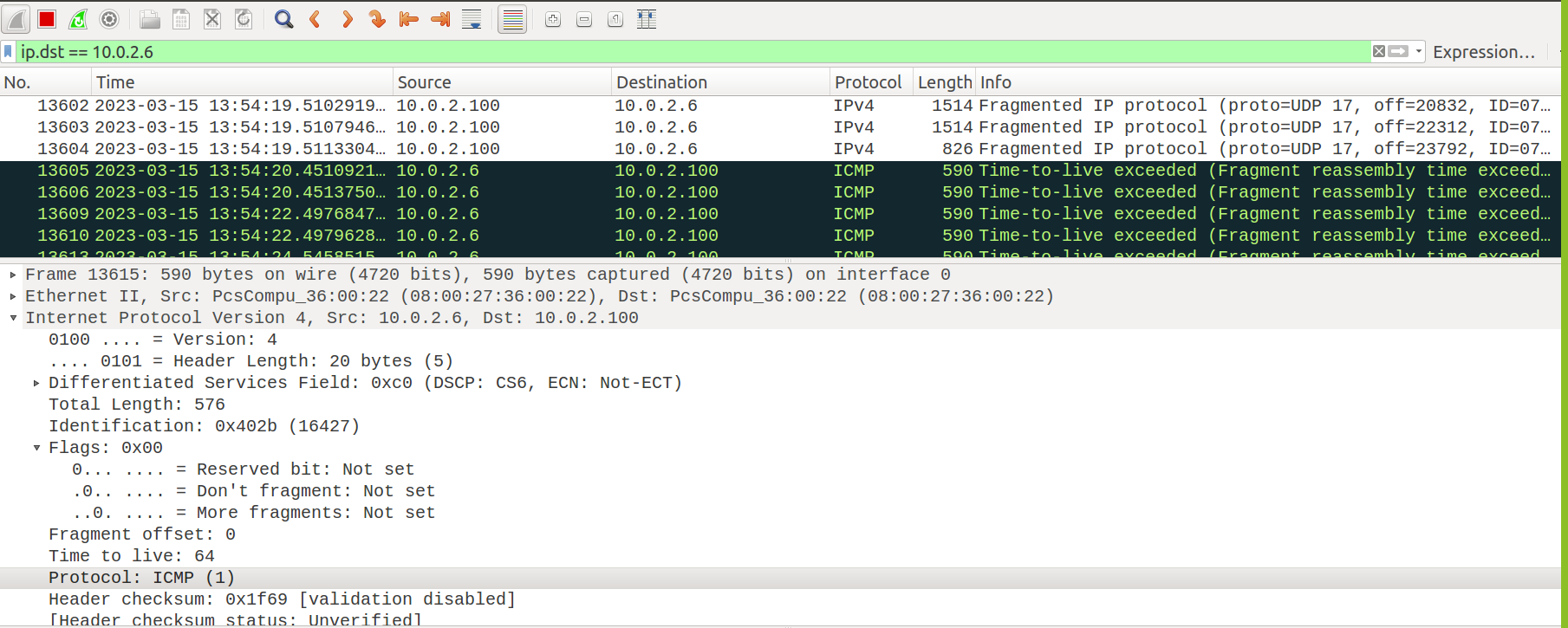


תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



לאחר שליחת הפקטות באמצעות הקוד שכתבנו ניתן לראות בwireshark שנשלחו פקטות רבות וגם ICMP TTL לאחר זמן מה מאחר והן נזרקו.



ניתן לראות בזיכרון השרת שהוא מתחיל להתמלא

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



מכאן רואים שהצלחנו במשימה מפני שהזיכרון הפנוי יורד לאט ואם היינו ממשיכים עם ההתקפה ומגדילים את גודל המידע שמועבר בכל פקט היינו עדים לשינוי קיצוני יותר בזיכרון הפנוי ולכן הוכחנו שניתן לבצע התקפת DoS באמצעות IP Fragmentation (1.8 -> 1.7)

* סיכום המשימה

הצלחנו לבצע את תתי המשימות במשימה 1 שהיוותה מכלול של .

החלק הראשון היה לבצע IP fragmentation ולחלק פקט ל3 חלקים ולשלוח אותה לשרת.

בחלק זה ראינו שהשרת קיבל את המידע המועבר והמידע הוצג בטרמינל השרת וכך הוכחנו שהצלחנו בביצוע המשימה.

בסעיף השני היינו צריכים לבצע חפיפה בין 2 fragments כאשר פעם אחת חלק מהמידע חופף ובחלק השני כל המידע חופף.

ראינו שהצלחנו לבצע את המשימה מאחר שבחפיפת החלקים המידע של הfragment השני נדרס.

בסעיף השלישי, היינו צריכים לשלוח פקט גדול מהמקסימום האפשרי ולראות אם נצליח לקבל את ההודעה בשרת. ראינו שהפקט מחולק באופן שונה ממה שאנחנו מגדירים בגלל חריגה מהMTU והמידע לא מוצג בשרת כי בהרכבה של כל החלקים המחשב מבין שחרגנו מהגודל המקסימלי לפקט ולא מקבל אותו.

בסעיף הרביעי, היינו צריכים לשלוח פקט מחולק כאשר אחד מהחלקים לא יישלח כך שהשרת אשר מקבל את החלקים ישמור אותם בזיכרון וימתין לחלק החסר לזמן מוגבל, דבר אשר יגרום לפגיעה בביצועי השרת ונקרא בעברית DoS.

הצלחנו בסעיף הרביעי לבצע העמסה לזיכרון של השרת והראינו זאת ע"י כך שהזיכרון הפנוי בשרת הלך וירד עם קבלת פקטות נוספות עם חלק חסר בכל פקט.

למדנו כיצד נדרש לחשב offset לכל fragment כדי שהפקט יישלח בצורה תקינה. מה הגודל המקסימלי של הודעת IP.

גילינו על מצב הדגלון שאחראי על חלוקת הפקט.

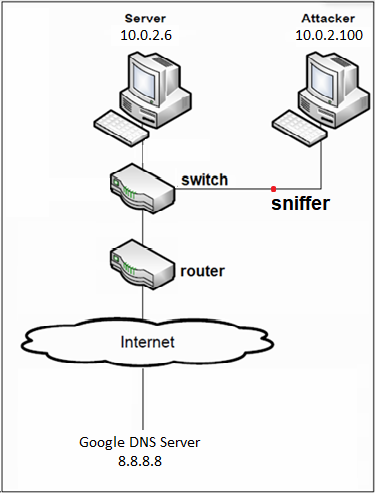
התוצאות לא תמיד התאימו למצופה לדוגמה בסעיף 2 לא ידענו אם A יידרס על ידי B או ההפך ובסעיף 3 לא ידענו איך לזהות הצלחה של שליחת פקט גדול מהמקסימום.

התמודדנו עם הבעיות בכך שנעזרנו בסטודנטים אחרים בקורס ובגוגל וגילינו מה התוצאות המצופות ורשמנו את ההסברים לתוצאות האלה בתתי סעיפים מעלה.

Task 2: ICMP Redirect Attack

* מבוא

במשימה נרצה לבצע MITM באמצעות ICMP REDIRECT כך שכל התעבורה בין הקורבן לשרת תעבור דרך התוקף ונשמש כראוטר לקורבן



* מטרה

ביצוע התקפת MITM כך שכל התעבורה של הקורבן ליעד מסוים ברשת יעברו דרך התוקף

* תוצאה מצופה

שנראה במסלול הניתוב של הקורבן שינוי של הראוטר דרכו הוא מעביר את התעבורה מ10.0.2.1 לכתובת 10.0.2.100

* ביצוע המשימה

תחילה נבצע בדיקה במחשב השרת דרך מי עוברות הפאקטות ליעד 8.8.8.8

ראינו שהפאקטות עוברות דרך ראוטר 10.0.2.1

כעת נריץ את הקוד שיעדכן את נתיב העברת הפאקטות דרך המחשב התוקף 10.0.2.100תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הראוטר 10.0.2.1 שולח לקורבן פאקט עם הוראה לעדכן את טבלת הניתוב מאחר שיש דרך יעילה יותר להגיע לשרת 8.8.8.8 דרך ראוטר 10.0.2.100 שהוא התוקף.

יש מספר קודים לביצוע ICMP REDIRECT:

קוד 0 - Redirect for host

מעדכן את השולח שכתובת היעד השתנתה

קוד 1 - Redirect for a network

מעדכן את השולח על ניתוב טוב יותר להעביר מידע דרכו

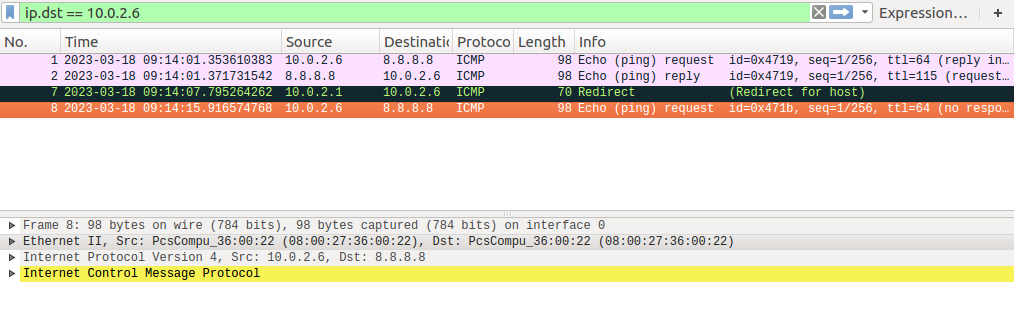
קוד 2 - Redirect for a type of service and network

מעדכן את השולח על מסלול טוב יותר עבור שירותים מסוימים

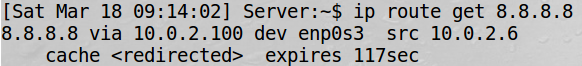
קוד 3 - Redirect for a type of service and host

מעדכן את השולח על מסלול טוב יותר עבור שירותים מסוימים הנשלחים למארח

לאחר שליחת הפאקט שבנינו, נבדוק בWIRESHARK שנקבל הודעת ניתוב מחדש:

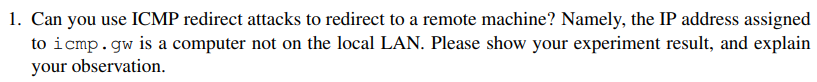


כעת נבדוק את מסלול הניתוב



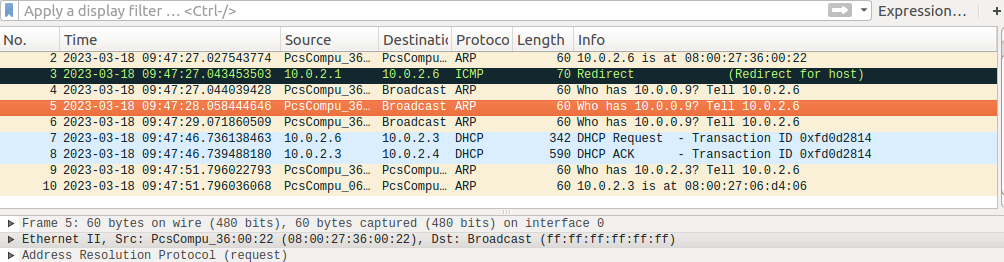
רואים שהצלחנו במשימה ומסלול הניתוב שונה בהצלחה.

שאלה 1:



לא, ICMP redirect attack לא יכולה לשנות את מסלול הניתוב למכונה מחוץ לרשת הפנימית מפני שההודעה מכילה הודעה למארח שקיים מסלול טוב יותר להגיע ליעד ברשת הפנימית.

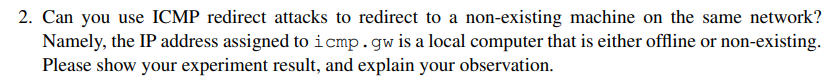
בדקנו את זה וראינו בwireshark שנשלחת הודעת arp request כדי לבדוק מי זה 10.0.0.9 ומאחר והוא לא באותה הרשת הפנימית הוא לא מזהה אותו



ולכן לא מבצע את העדכון ומתעלם מהicmp redirect



שאלה 2:



לא ניתן לבצע ניתוב מחדש אל מחשב שנמצא ברשת כאשר הוא כבוי או לא קיים. הבקשה תיזרק ולא תתבצע שום פעולה ומסלול הניתוב יישאר כפי שהיה.

* סיכום המשימה

הצלחנו לבצע את המשימה בכך שהראנו כיצד מסלול הניתוב אצל הקורבן השתנה וגילינו על שיטה חדשה לבצע מתקפת MITM.

התוצאה התאימה למצופה מאחר שמסלול הניתוב השתנה כפי שרצינו.

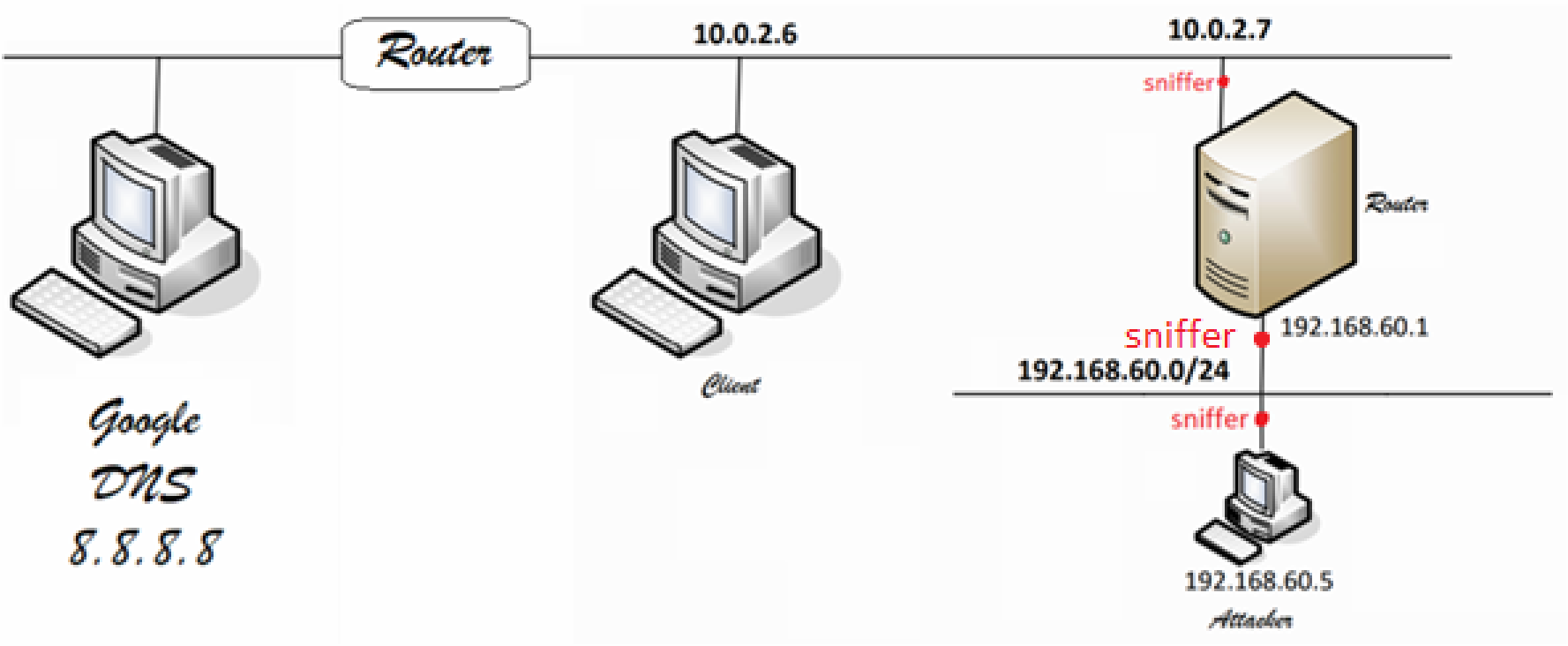
לא נתקלנו בבעיות בזמן ביצוע המשימה.

Task 3: Routing and Reverse Path Filtering

* מבוא

במשימה נרצה להגדיר 2 רשתות שונות כאשר יש מחשב שיוגדר בתור הראוטר שיעביר את הפקטות ביניהם.

נרצה במשימה הזו לבדוק את המנגנון הגנה במערכת ההפעלה של לינוקס להתחקות אחר המתאם רשת שממנו נשלח הפקט בפועל למול מתאם הרשת של מקור הפקט שזייפנו ממנו.



מטרה

לראות אם המנגנון מצליח להגן מפני ip spoofing בכך שהוא מזהה שהפקטות נשלחו ממתאמי רשת שונים.

* תוצאה מצופה

מערכת ההפעלה תבצע בעת קבלת פקט מבט לאחור ולפני שהיא תעביר את הפקט היא תבצע בדיקה האם מסלול החזרה של הפקט יהיה זהה מבחינת מתאמי הרשת שהוא עובר דרכם. אם כן, הפקט תתקבל בכתובת היעד אחרת מערכת ההפעלה תזרוק את הפקט והיעד לא יקבל את הפקט.

במילים אחרות, אם המתאמים שונים במסלול השליחה והחזרה אז המסלול הוא אסינכרוני ומערכת ההפעלה תזרוק את הפקט.

* **בזמן הגדרת הרשת הגדרנו את התוקף ברשת הפנימית ואת הקליינט ברשת NAT וביצענו לפי המעבדה את ההתקפה מהקליינט לתוקף.**

Task 3.a: Network Setup

תחילה התבקשנו להגדיר את הרשת כדי שתיראה כמו בשרטוט מעלה, כלומר מחשב הקליינט משתמש במחשב הראוטר כדי לתקשר עם מחשב התוקף ולהפך כששניהם נמצאים ברשתות שונות.

לאחר מעקב אחר הפעולות שהתבקשנו לבצע לצורך ההגדרה ניתן לראות שהמכונות הוגדרו כנדרש:

תמונה שמכילה טקסט, מכתב

התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט, מכתב

התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט, מכתב

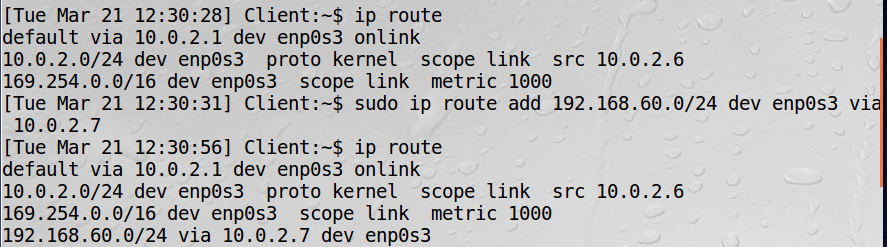
התיאור נוצר באופן אוטומטי



ניתן לראות בתמונות את כתובות הIP והמתאמי רשת שהוגדר לכל מחשב.

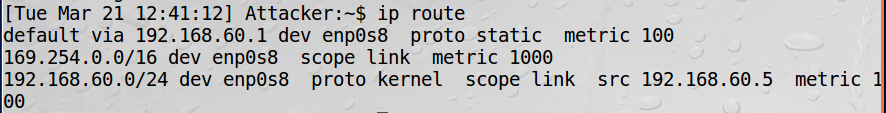
Task 3.b: Routing Setup

במחשב הקליינט 10.0.2.6 הגדרנו ניתוב נוסף דרך 10.0.2.7 שהוא הראוטר בין 2 הרשתות הנפרדות.

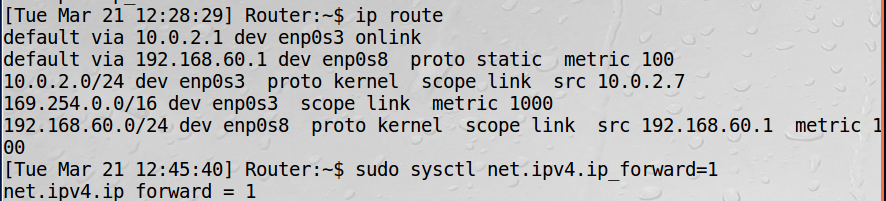




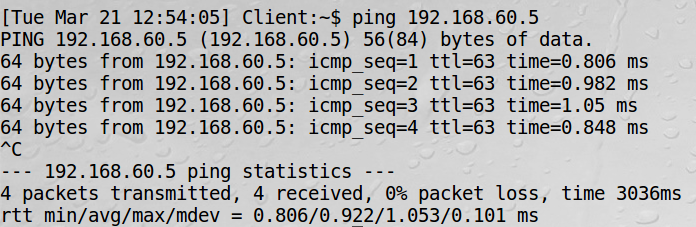
במחשב התוקף 192.168.60.5 לא היינו צריכים להוסיף או למחוק כלום מאחר והוא כבר מנותב דרך הראוטר בכתובת 192.168.60.1



בראוטר שהגדרנו נבצע IP Forwarding בצורה הבאה:



לאחר כל ההגדרות נרצה לוודא שניתן לתקשר בין 2 המחשבים ברשתות השונות בכך שנשלח פינג





ניתן לראות שהפינג עבר בהצלחה מהקליינט לתוקף ולכן הצלחנו בהגדרה של הרשת הנוספת.

Task 3.c: Reverse Path Filtering

בסעיף זה נרצה לזייף 3 פקטות שונות כאשר בכל פקט נרשום כתובת מקור שונה ונרצה לבדוק האם מערכת ההפעלה מעבירה או זורקת את הפקט.

הרצנו את הקוד הבא אצל הקליינט כאשר כתובת המקור היא כתובת מזוייפת הנמצאת ברשת הNAT של הקליינט וכתובת היעד היא התוקף שנמצא ברשת הפנימית:



#!/usr/bin/python3

from scapy.all import \*

def spoof\_pkt():

a = IP()

a.src = "10.0.2.2" # spoof from NAT network

a.dst = "192.168.60.5"

b = ICMP()

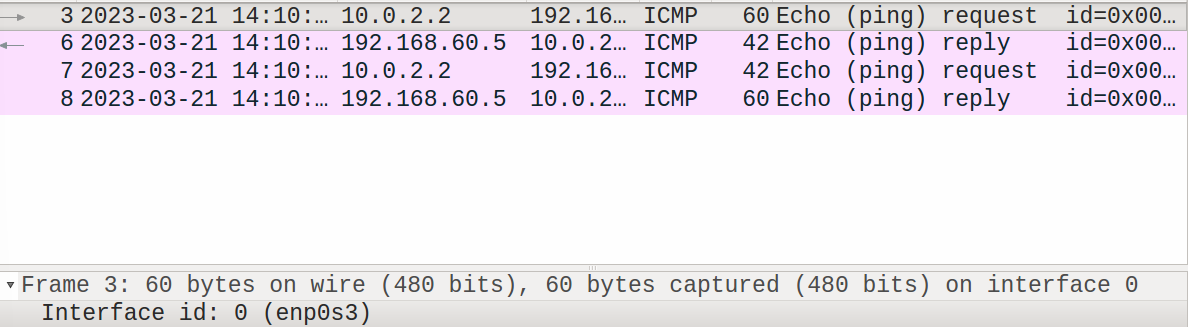
b.type = 8

send(a/b,verbose=0)

spoof\_pkt()

print("sent spoofed packet")

התמונה מהwireshark במחשב הראוטר



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ניתן לראות שהפקט הועבר בהצלחה בין 2 המתאמים מאחר וקיבלנו את אותו הפקט פעמיים, כל פעם ממתאם אחר.

שלחנו פקט מenp0s3 אל enp0s8 וקיבלנו אותו גם מenp0s8 חזרה אל enp0s3 ולכן המסלולים סימטריים.

תמונה מה wireshark של התוקף

תמונה שמכילה טקסט

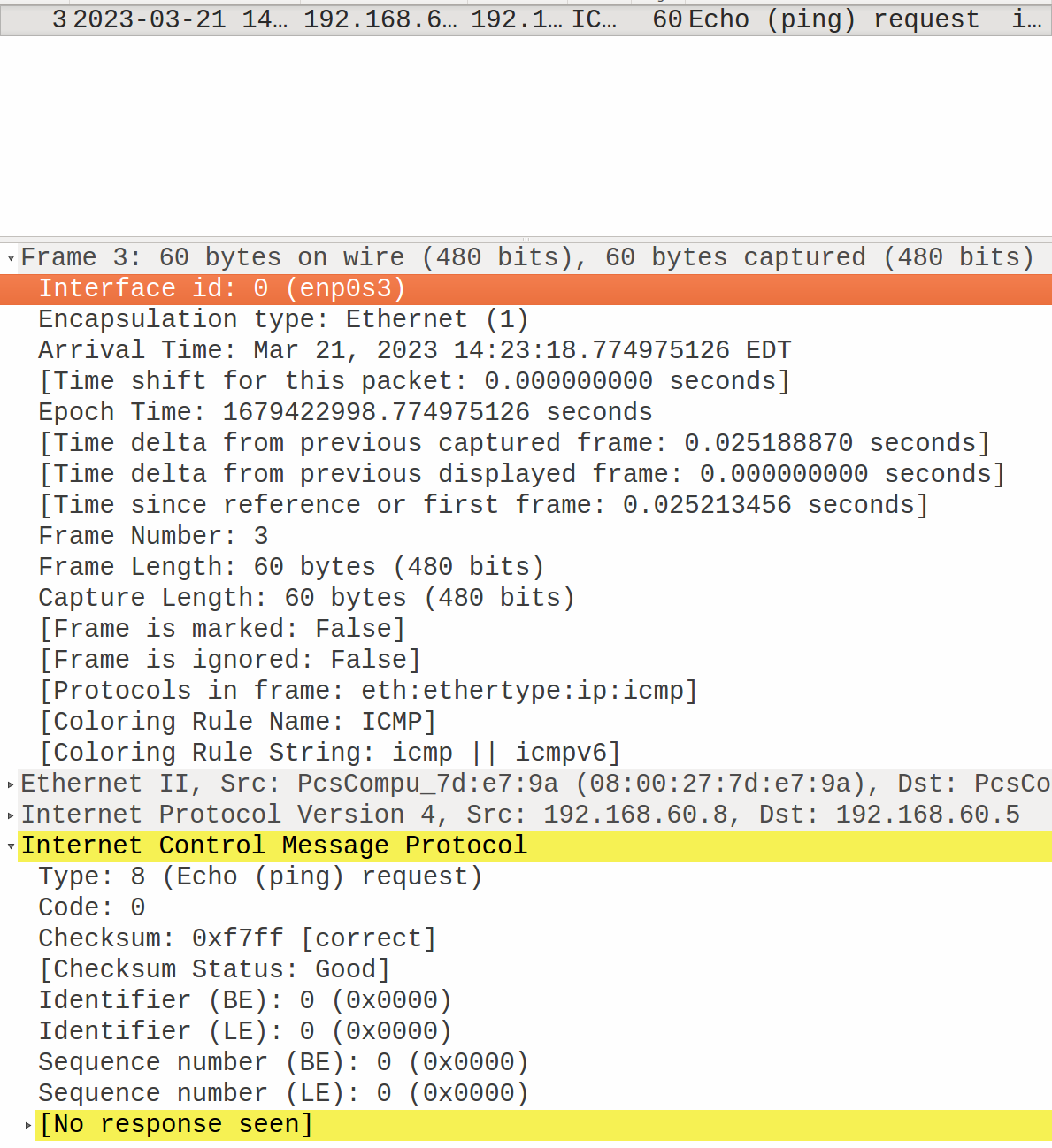
התיאור נוצר באופן אוטומטי

ניתן לראות שהפקט התקבל בהצלחה במחשב התוקף במתאם הרשת הפנימית.

הרצנו את הקוד הקודם אצל הקליינט כאשר כתובת המקור היא כתובת מזוייפת הנמצאת ברשת הפנימית של התוקף וכתובת היעד היא התוקף עם השינוי הבא: 

a.src = "192.168.60.8" # spoof from internal network

התמונה מהwireshark במחשב הראוטר





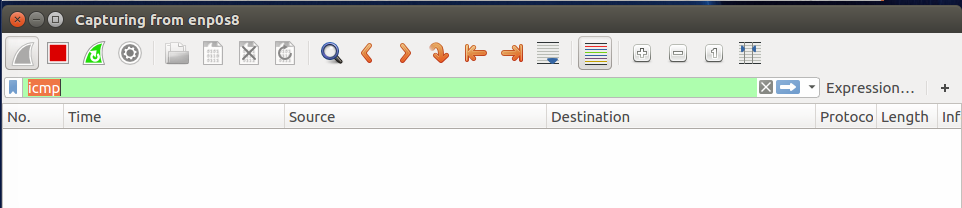
ניתן לראות שהפקט הועבר בהצלחה בין 2 המתאמים מאחר וקיבלנו את אותו הפקט פעמיים, כל פעם ממתאם אחר.

ניתן לראות שהפקט נשלח ממתאם enp0s3 מרשת הNAT אל הרשת הפנימית במתאם enp0s8.

בעת קבלת הפקט אצל הראוטר מערכת ההפעלה תבדוק דרך איזה מתאמים הפקט עבר ותבדוק אם בהחזרת הפקט אל היעד היא תעבור באותם המתאמים. מאחר והפקט נשלח מהNAT אל הרשת הפנימית ובחזרה שלה היא תעבור מהרשת הפנימית אל הרשת הפנימית (enp0s8 אל enp0s8)

מערכת ההפעלה מזהה שהמסלול אסימטרי וזורקת את הפקט ולכן נראה שאין תגובה לבקשת הICMP שזייפנו.

תמונה מה wireshark של התוקף



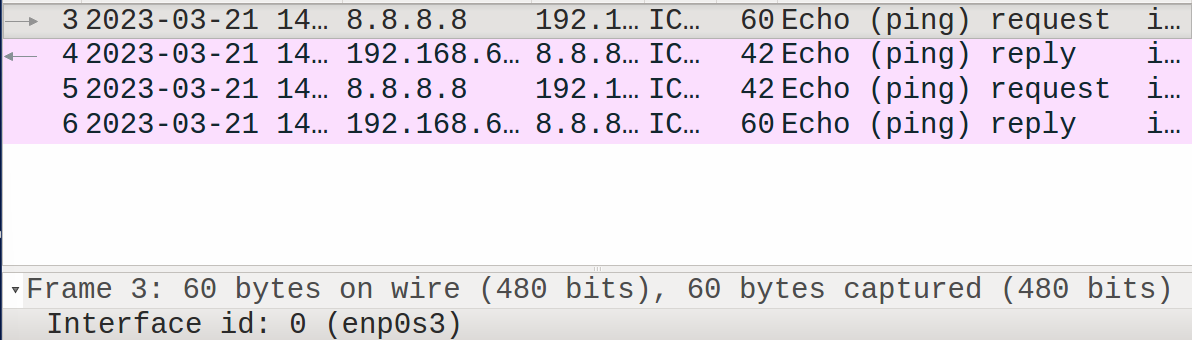


כפי שתיארנו, נראה כי הפקט לא התקבל אצל התוקף.

הרצנו את הקוד הקודם אצל הקליינט כאשר כתובת המקור היא כתובת מזוייפת הנמצאת ברשת האינטרנט (8.8.8.8) וכתובת היעד היא התוקף עם השינוי הבא: 

a.src = "8.8.8.8" # spoof from the internet

התמונה מהwireshark במחשב הראוטר



ניתן לראות כמו במצב הראשון שהפקט התקבל בשני המתאמים מאחר והגישה לאינטרנט היא דרך הקליינט שמחובר אל רשת האינטרנט דרך המתאם שלו, כך שגם שליחת הפקט וגם חזרתו עברו דרך אותם המתאמים, מה שמעיד על מסלול סימטרי.

* ביצענו גם את הכיוון ההפוך (מהרשת הפנימית אל הNAT) והפקט נזרק מאחר ושלחנו לNAT מהרשת הפנימית enp0s8 ודרך החזרה היא דרך רשת הNAT שהיא enp0s3 אל האינטרנט שמקושר גם דרך enp0s3 ולכן המסלולים אסימטרים ולא קיבלנו את הפקט אצל הקליינט.

תמונה מה wireshark של התוקף

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ניתן לראות שהפקט התקבל בהצלחה במחשב התוקף במתאם הרשת הפנימית.

* סיכום המשימה

הצלחנו לבצע את המשימה וראינו את מסלולי הפקטות ודרך איזה מתאמים הם עוברים. הבנו את אופן הפעולה של מערכת ההפעלה כאשר היא מבצעת סינון reverse path filtering לפקטות שמתקבלות אצלה ולפי מסלול השליחה והחזרה יודעת אם המסלול סימטרי או לא ולפיכך לקבל או לזרוק את הפקט.

למדנו שזו שיטה של מערכת ההפעלה להגן מפני ip spoofing ע"י בדיקת כתובת המקור של הפקט המתקבל לקיום מסלול חזרה דרך אותם המתאמים.

ראינו שהתוצאות התאימו למצופה מאחר ופקטות שמסלול השליחה והחזרה שלהם עברו דרך מתאמים שונים (מסלולים אסמיטריים) נזרקו ואם הן עברו דרך מסלול סימטרי הפקטות התקבלו אצל היעד.

חווינו בעיות בהגדרת הרשת כך שלא ידענו מה צריך למחוק ולמי צריך להוסיף מתאם ודרך ניתוב ופתרנו זאת ע"י ניסוי וטעייה ואין כמו עזרה מחברים.

**סיכום כללי למעבדה**

משימות המעבדה כללו פיצול וחפיפה של פקטות, שליחת פקטות גדולות מהגודל המרבי, ביצוע התקפה על השרת (DoS) בעזרת החסרת fragment של פקט, וניתוח מסלולי הפקטות (בדיקת המתאמים דרכם הן עוברות) וסינון פקטות שמסלול החזרה שלהם עובר דרך מתאמים שונים ממסלול השליחה.

נתקלנו בכמה אתגרים בדרך אבל הצלחנו להתגבר עליהם באמצעות ניסוי וטעיה, מחקר וסיוע של סטודנטים בקורס.

צברנו ידע על ניתוב פקטות, זיוף IP ומנגנוני ההגנה של מערכת ההפעלה.

**משהו חדשני:**

לאחר חקירה גילינו שתי דרכים נוספות להתגונן מפני ip spoofing attack

1. Ingress filtering (סינון כניסה) - טכניקה זו כוללת בדיקת אימות שכתובת המקור של הפקט חוקית ושייכת לרשת שממנה הגיעה החבילה. אם כתובת המקור של הפקט אינה חוקית או אינה שייכת לרשת, החבילה נזרקת.
2. Reverse Path Forwarding (RPF) - טכניקה זו כוללת אימות שהפקט מגיע למתאם שהנתב ישתמש בו כדי להעביר את התגובה חזרה לכתובת המקור של הפקט. אם החבילה מגיעה למתאם שלא ישמש להעברת התגובה חזרה, החבילה נזרקת.

כדי לוודא אם RPF מופעל נרשום את הפקודה הבאה:

sysctl net.ipv4.conf.all.rp\_filter

אם התשובה שנקבל היא 1 אז המסנן עובד אחרת נרצה להפעיל אותו באמצעות הפקודה:

sudo sysctl -w net.ipv4.conf.all.rp\_filter=1

בנוסף גילינו שמה שלמדנו בתחילת המעבדה יכול גם לספק דרך לעקוף את מנגנון ההגנה RPF ולבסס התקפה באמצעות פרגמנטציה הכוללות מניפולציה של האופן שבו הפקטות מחולקות ומורכבות מחדש.

לדוגמה, תוקף יכול לשלוח פקטות מקוטעות עם מידע שגוי או חסר, ולגרום למכשיר היעד להרכיב מחדש את החבילות באופן שגוי ולאפשר לתעבורה זדונית לעקוף את המסננים.

אפשרות הגנה מפני התקפה זו: להשתמש בסינון פקטות ובמערכות IDS/IPS שיכולות לזהות ולחסום פקטות שאינן תואמות לסטנדרטים ספציפיים.

בנוסף, ניתן להשתמש במערכות זיהוי ומניעת חדירה (IDS/IPS) כדי לנטר את תעבורת הרשת ולזהות התנהגות חריגה שעלולה להעיד על התקפה, על ידי שילוב אמצעי אבטחה אלה, מנהלי רשת יכולים להגן טוב יותר מפני התקפות מבוססות פרמנטציה ואיומים אחרים.