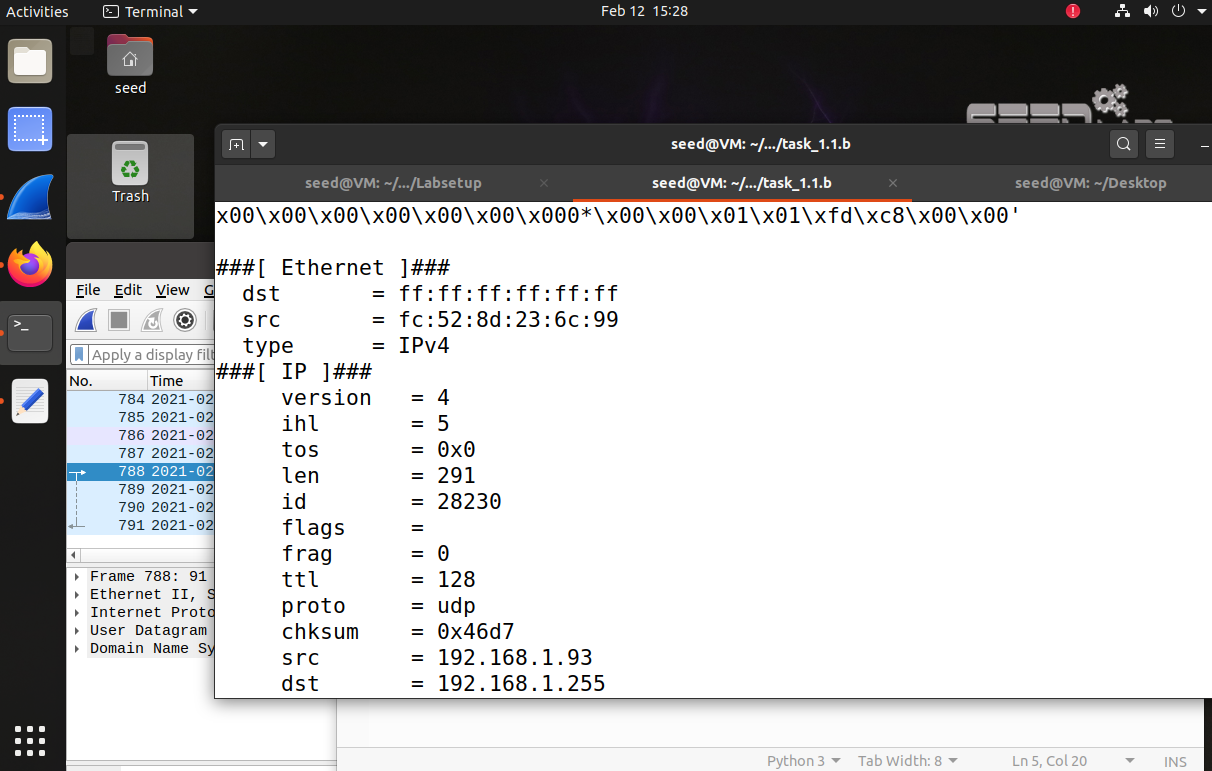
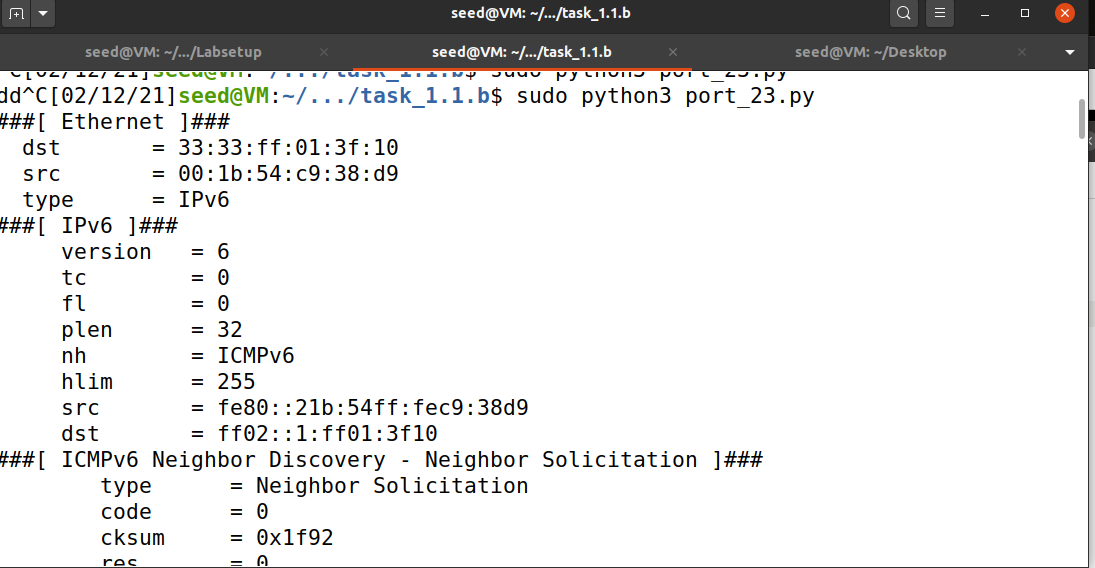
מעבדה Sniffing and Spoofing

שאלה A1.1 : כאשר הרצנו את התוכנה ללא הרשאות ROOT התקבלה שגיאת operation not permitted שהגיעה מקובץ socket.py . הסיבה לכך היא שSCAPY צריכה ליצור raw sockets כדי להריץ את התוכנה ודבר זה אפשרי רק עם הרשאות גבוהות (ROOT)

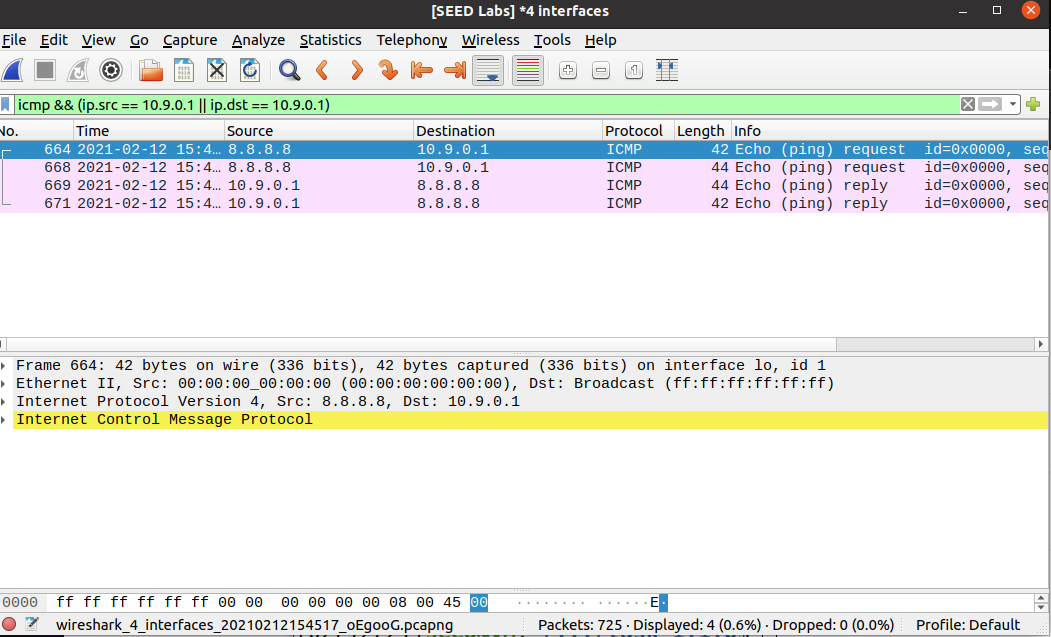
שאלה 1.1B : מצורפים 3 קבצי פייתון עם הסינונים המבוקשים ב SYNTAX של Berkeley Packet Filters





שאלה 1.2 :

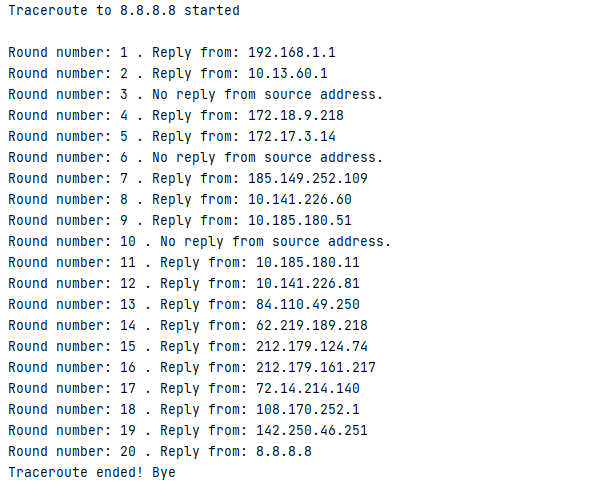
כפי שניתן לראות בצילום המסך מטה ובקובץ הWIRESHARK שנקרא icmp\_spoof.pcap ב2 השורות הראשונות כתובת המוצא זיופה בהצלחה לכתובת 8.8.8.8 שהוא כתובת הDNS הפתוח של חברת גוגל בחבילת ICMP echo request הידועה גם בשם PING . ב2 השורות האחרונות ניתן לראות את התשובה מהכתובת 10.9.0.1 בחבילת ICMP echo replay.



שאלה 1.3 :

תוכנה המדמה traceroute מצורף בקובץ traceroute.py

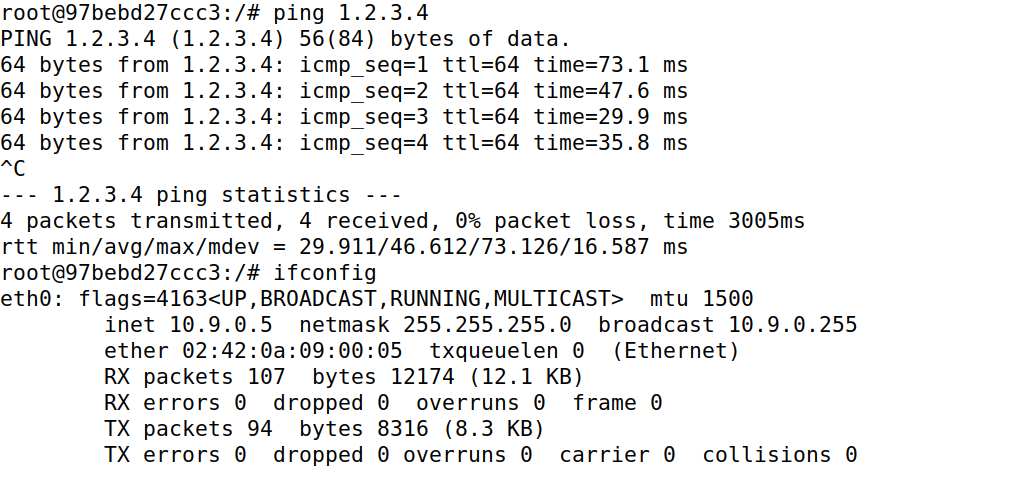
פלט לדוגמה מהתוכנה הבודק את המרחק ב TTL ל שרת הDNS של גוגל 8.8.8.8 :



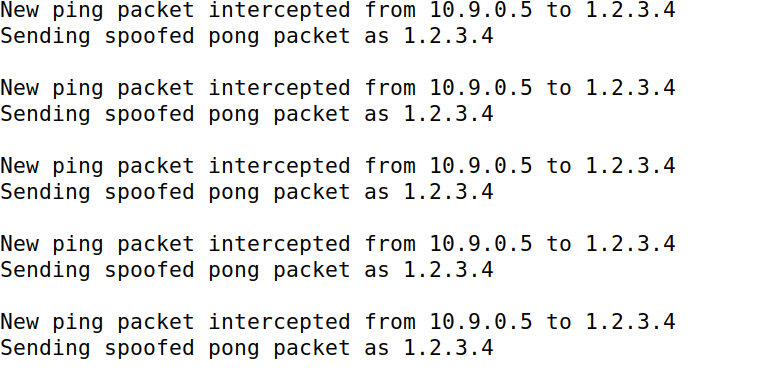
שאלה 1.4 :

כדי לזייף תשובות לפקודת ICMP ECHO REQUEST (שנקראת גם PING) , עלינו לשנות את כתובת המוצא והיעד של הבקשה כדי שיראו כאילו הגיעו מהיעד המקורי של הפקטה. לאחר מכן יש לשנות את סוג פקטת ה ICMP לסוג 0 , שהוא הקוד לתשובה לגיטימית (ICMP ECHO RESPONSE ) לבקשה שנשלחה.

כפי שניתן לראות בתמונות הבאות, בכל אחד מכתובות היעד שנשלח אליהם PING מה DCOKER , הVM HOST ענה בהצלחה בפקטה שזויפה ע"י התוכנה sniff\_and\_spoof.py .



הפקטה שנשלחה לכתובת שלא קיימת 1.2.3.4 מה DOCKER



התשובה שנשלחה מ sniff\_and\_spoof

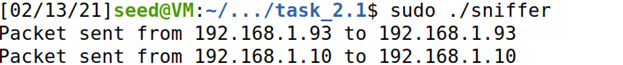
1.2.3.4 נשלחה מהDOCKER ל default gateway שהוא הVM עצמו , ונשלחה חזרה מVM

8.8.8.8 נשלחה מהDOCKER ל default gateway שהוא הVM עצמו , ונשלחה חזרה מVM

10.9.0.99 נשלחה מהDOCKER ב BROADCAST לרשת שבו היא נמצאת מכיוון שמדובר על כתובת שבאותה SUBNET כמו של הIP של הDOCKER שממנו נשלחה הבקשה (10.9.0.5) . מכיוון שלא הגיעה לdefault gateway לא יכלנו לענות ולזייף את התשובה מתוך ה VM (אבל ישנן דרכים אחרים לעשות זאת) . ומכיוון ש 10.9.0.9 היא כתובת שאינה קיימת בSUBNET לא נשלחה תשובה חזרה ל DOCKER .

שאלה 2.1.A:

הקוד מצורף בתיקיה לשאלה זו. צילום מסך להוכחה שתוכנת sniffer התקמפלה ורצה בהצלחה:



**הסבר קצר על התוכנה:**

בשלב הראשון אנו קוראים לפונקציה מתוך ספרית PACP שנקראית pcap\_open\_live על מנת לפתוח את כרטיס הרשת להסנפה. יש לציין את שם כרטיס הרשת כפי שמופיע ב ifconfig .

לאחר מכן קוראים ל pcap\_compile ו pcap\_setfilter על מנת להגדיר פילטרים להסנפה, השתמשתי בהגדרות הדיפולטיות שהופיעו במדריך , להסניף רק ICMP.

בשלב הבא אנו קרואים ל pcap\_loop על מנת להתחיל את ההסנפה לייב מכרטיס הרשת ומגדירים את פונקציית got\_packet כפונקציית ה CALLBACK שתיקרא על כל פקטה שתיורט. מכיוון שהגדרנו את פרמטר CNT כ -1 , ההסנפה תתבצע בצורה אינסופית (או עד אשר התוכנה תיעצר באופן יזום על ידי המשתמש).

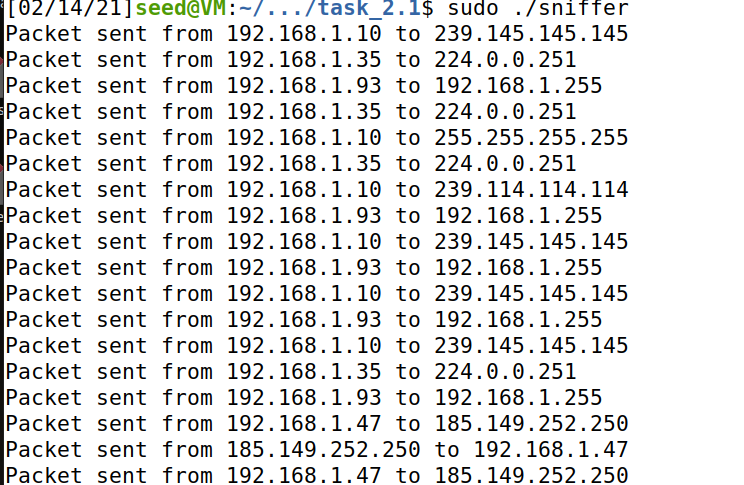
כל פעם ש got\_packet תיקרא לאחר שפקטה מסוימת התאימה לתנאים והפילטרים שהגדרנו , אנו נחלץ את חלק הethernt וחלק הIP מהפקטה שיורטה ונדפיס את כתובת המוצא והיעד.

**מדוע צריך הרשאות ROOT כדי להריץ את הסניפר?**

ללא הרשאות ROOT , התוכנה מקבל segmentation fault וקורסת. לאחר הוספת תנאי IF לפונקציית pacp\_open\_live ניתן לראות שהשגיאה קורית כאשר התוכנה מורצת ללא הרשאות גבוהות והHANDLE שחוזר מהופנקציה הוא NULL . הסיבה לכך שהתוכנה צריכה הרשאות ROOT היא מאותה סיבה שהיינו צריכים אותם שהשתמשנו ב SCAPY – על מנת ליירט raw packets מכרטיס הרשת אנו צריכים הרשאות אלה, מכיוון שמדובר בתהליך שיכול להיות מסוכן למשתמש.

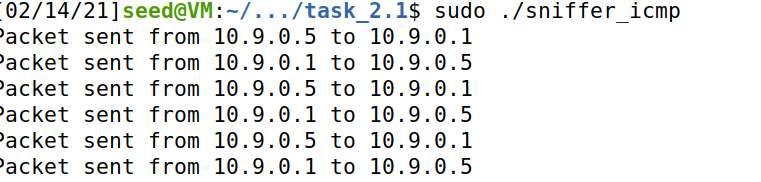
**הפעלת promiscuous mode**

כאשר מצב זה מופעל, ניתן להסניף פקטות שלא בהיכרח יועדו לכתובת של כרטיס הרשת שלנו אבל כן עברו דרכו בדרך אל היעד ובחזרה ממנו. פרמטר promisc שהוא הפרמטר השלישי בפונקציית pacp\_open\_live הוא בוליאני שמגדיר האם ההסנפה תתבצע במצב מופעל או כבוי. אם משתנה זה הוא 1 , promiscuous mode מופעל ונוכל להסניף את כל הפקטות שעברו אצלנו ואם הוא 0 המצב כבוי. בתמונה למעלה ניתן לראות את הסניפר כאשר המצב כבוי, ובתמונה למטה ניתן לראות את הסניפר כאשר המצב מופעל.

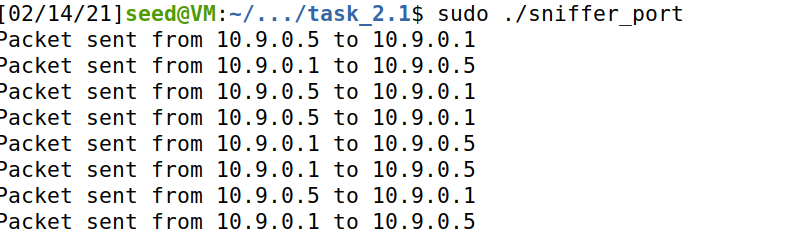


שאלה 2.1.B:

הפעלת פילטר ICMP על שני HOSTS

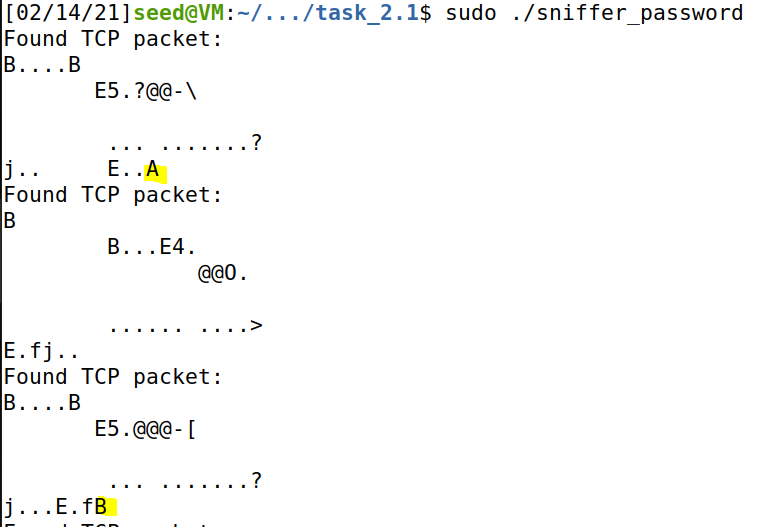
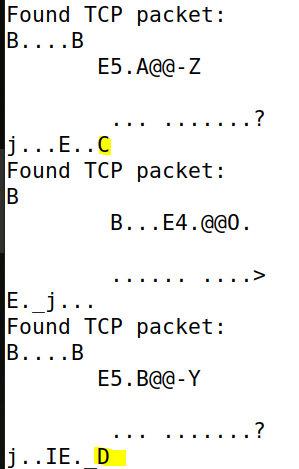


הפעלת פילטר TCP עם פורט יעד בין 10-100



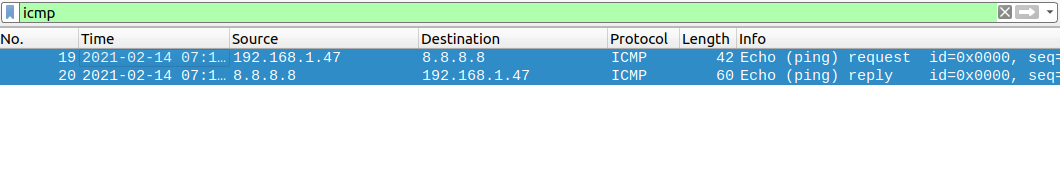
שאלה 2.1.C:

מכיוון ש TELNET שולח את הסיסמה תו אחר תו בפקטות נפרדות, מצורפים 2 צילומי מסך שלכדו את הסיסמה שהכנסתי "ABCD"

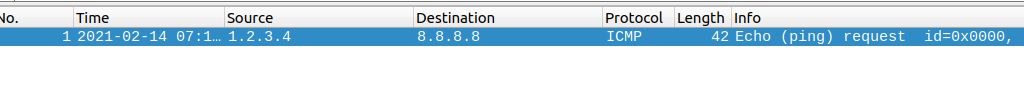
 

שאלה 2.2.A:

צילום מסך של יצירת raw packet של ICMP

 שאלה 2.2.B:

צילום מסך של SPOOF של ICMP שנשלח מכתובת 1.2.3.4



**האם ניתן לשנות את אורך IP PACKET לערך רנדומלי?**

לא. אם נשנה את ערך האורך לערך רנדומלי ולא לאורך האמיתי של הפקטה נקבל שגיאה בעת ריצה בפונקציית sendto

**האם צריך לחשב את ה checksum של ה IP HEADER ?**

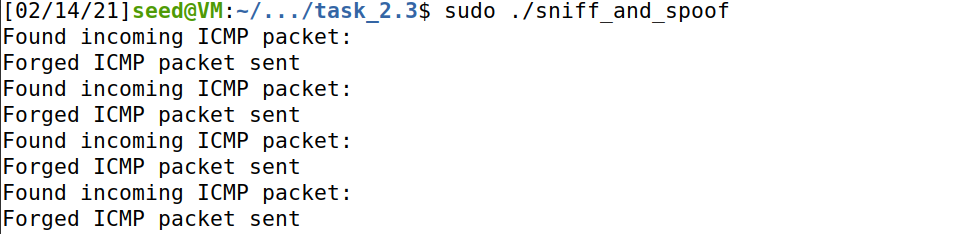
מערכת ההפעלה יודעת לחשב את ה checksum ל IP HEADER אבל כן צריך לחשב בעצמנו לפרוטוקולים אחרים כמו ICMP , כפי שניתן לראות בקוד.

**האם צריך ROOT ?**

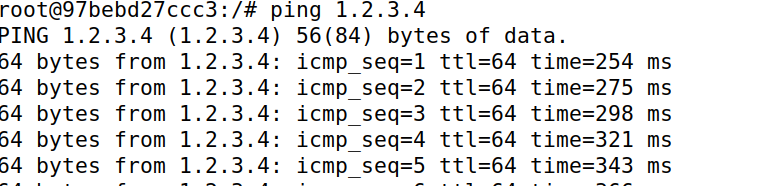
על מנת ליצור RAW PACKETS צריך לרוץ עם הרשאות ROOT בדיוק כמו בתרגילים של ההסנפה מכיוון שמדובר בתהליך העלול להיות מסוכן למשתמש. כאשר מנסים להריץ את התוכנה ללא ROOT היא נכשלת בעת יצירה הSOCKET בפונקציית send\_packet שכתבתי.

שאלה 2.3 :

ניתן לראות בצילומי המסך הבאים את צד התוקף שמאזין ושולח תשובות לפקטות מסוג ICMP שנשלחות מתוך הDCOKER לכתובת שלא קיימת - 1.2.3.4



צד התוקף



פינג שנשלח מהDOCKER ל 1.2.3.4