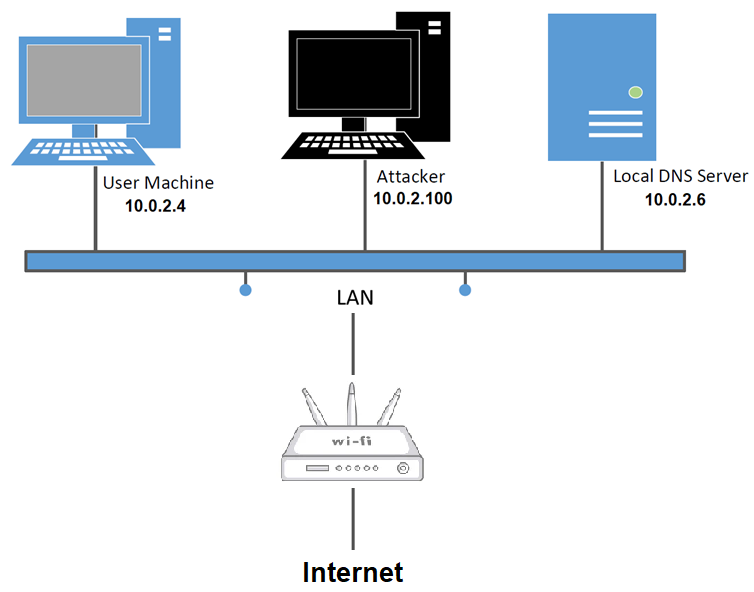
**A04 – Remote DNS Attack Lab**

כתובות IP לכל מחשב

|  |  |  |
| --- | --- | --- |
| Name | IP | MAC |
| Attacker | 10.0.2.100 | - |
| Client | 10.0.2.4 | - |
| DNS Server | 10.0.2.6 | - |

**2. Lab Environment Setup Tasks**

התמונה מטה מייצגת את מבנה הרשת

Task 1: Configure the User VM

* מבוא:
  + תיאור

במשימה זו נרצה להגדיר בCLIENT מי השרת DNS שאליו הוא יפנה את הבקשות ויקבל ממנו את התשובות

* + מטרה

להגדיר שרת DNS לוקאלי שאליו הCLIENT ישלח שאילתות DNS ויקבל מענה לשאילתות האלו.

* + תוצאה מצופה

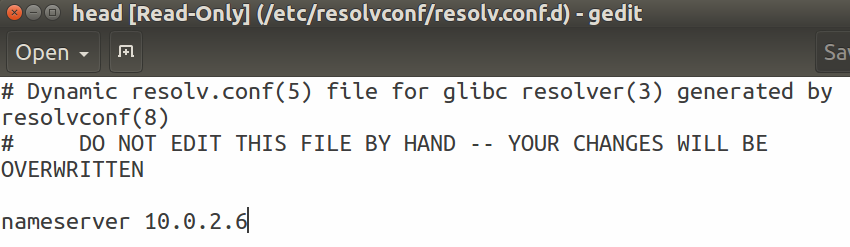
כל בקשות הDNS של הclient ישלחו אל מחשב הביניים שמשמש כ local dns וכל התשובות לשאילתות האלו ישלחו מהlocal dns אל הclient, כלומר הlocal dns יתקשר עם הרשת החיצונית ויעביר את המידע לclient.

* ביצוע המשימה:

תחילה נרצה להגדיר במחשב הCLIENT ששרת הDNS העיקרי שלו יהיה מחשב השרת IP 10.0.2.6 לכן נבצע עריכה לקובץ /etc/resolvconf/resolv.conf.d/head ונוסיף את השורה הבאה:

Nameserver 10.0.2.6

צילום ממחשב הCLIENT



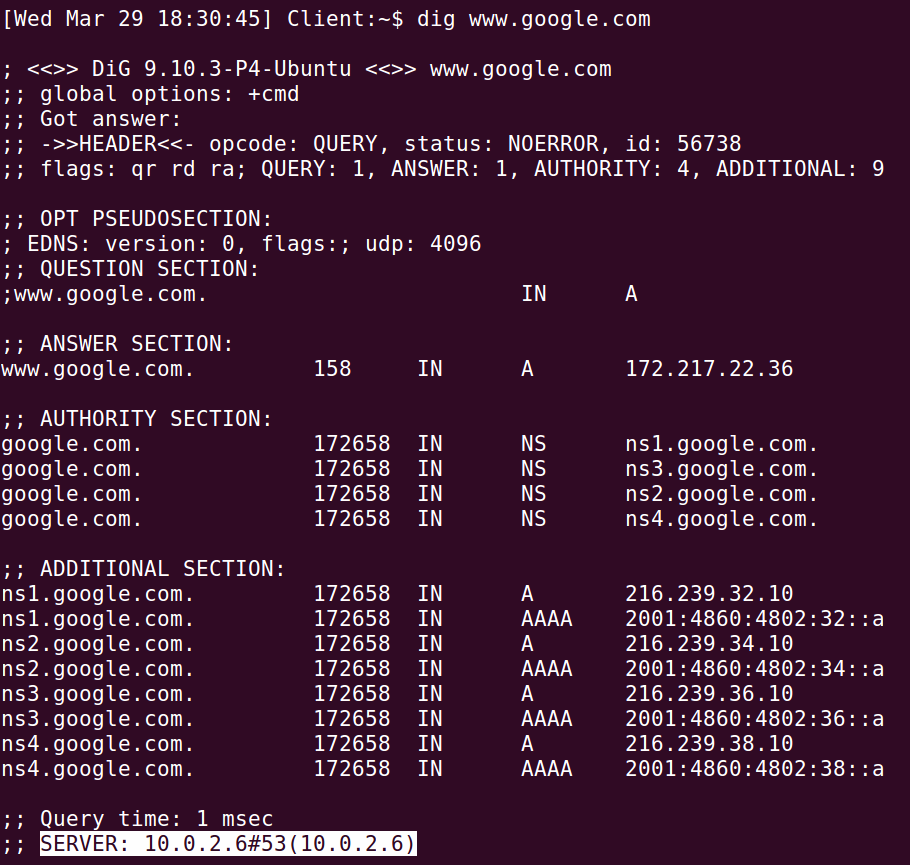
כעת נרשום את הפקודה הבאה בטרמינל:

sudo resolvconf -u

לצורך החלת השינויים בקובץ

כעת נרצה לבדוק במחשב הCLIENT שאכן שרת הDNS דרכו הוא מעביר בקשות הוא השרת שלנו 10.0.2.6

נבצע זאת על ידי הפקודה dig [www.google.com](http://www.google.com)



ניתן לראות בשורה התחתונה המסומנת שאכן הסרבר דרכו עברה הבקשה הוא הסרבר שלנו 10.0.2.6

SERVER מציין את השרת שענה לבקשה שלנו

53 מציין את הפורט שבו השרת מאזין לבקשות שלנו

* סיכום המשימה

הצלחנו לבצע את המשימה, ניתן לראות שהגדרנו את מחשב 10.0.2.6 בתור local dns, וחיברנו את הclient ל10.0.2.6 כך שכל שאילתות הDNS יעברו דרכו

הראינו בעזרת הפקודה DIG שאכן כל השאילתות DNS נשלחות לlocal dns שהגדרנו שהוא 10.0.2.6.

גילינו כיצד להגדיר באופן סטטי שרת DNS שיעביר את שאילתות הDNS של המחשב.

התוצאות התאימו למצופה מאחר והPACKETS של שאילתות הDNS הועברו דרך ה LOCAL DNS שהגדרנו.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 2: Configure the Local DNS Server (the Server VM)

* מבוא:
  + תיאור

במשימה זו נרצה להגדיר את שרת הDNS כך שכל הפניות לZONE מסוים יופנו לתוקף ולקנפג את הגדרות השרת הלוקאלי

* + מטרה

להגדיר שרת DNS לוקאלי שיתווך בין הclient לרשת החיצונית, ולדאוג שכל הבקשות לZONE מסוים יופנו לתוקף

* + תוצאה מצופה

שמירת ההגדרות בצורה נכונה כפי שהתבקשנו

* ביצוע המשימה:

**את השלבים הבאים נבצע במחשב השרת 10.0.2.6**

* Step 1: Remove the example.com Zone

לא רלוונטי מאחר ושחזרנו את מצב המחשבים לברירת מחדל.

* Step 2: Set up a forward zone

כדי לבצע זאת נכנס לקובץ בנתיב /etc/bind/named.conf.local ונרצה להגדיר את הZONE הבא:

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

לאחר ההגדרה הזו, כל ההפניות לדומיין attacker32.com יופנו אל התוקף בכתובת 10.0.2.100 ולא ישלחו שאילתות לבירור כתובת הIP האמיתית של הדומיין מאחר והוא כבר רשום בזיכרון.

* Step 3: Configure a few options





נרצה להגדיר את שרת ה- BIND 9 וליצור נתיב לזריקת הCACHE של השרת

נכנס לפי ההוראות במעבדה לקובץ בנתיב: /etc/bind/named.conf.options.

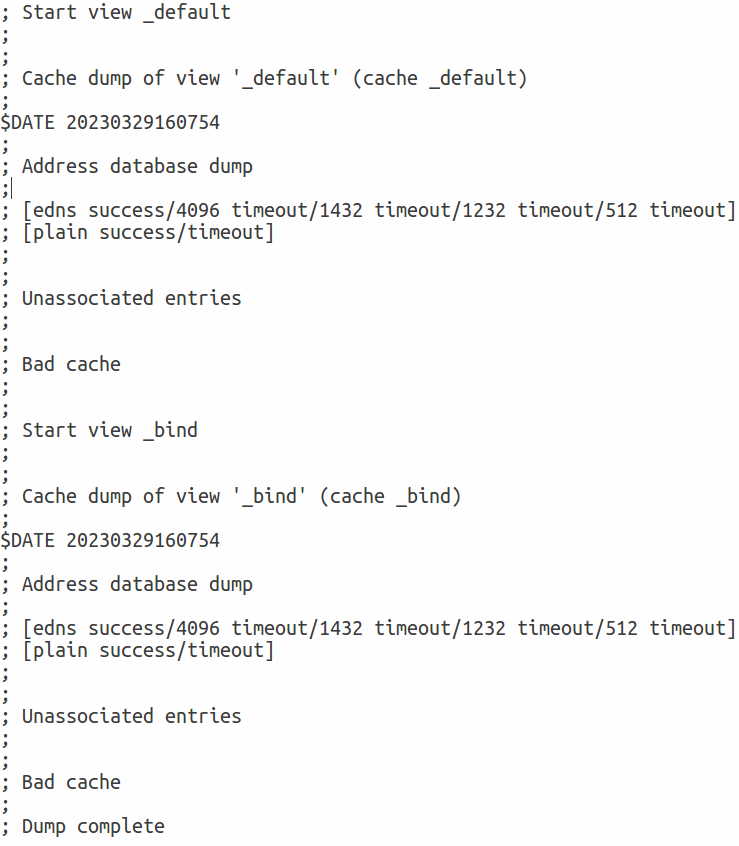
ונוסיף את השורה הבאה:

dump-file "/var/cache/bind/dump.db";

כעת נרצה ליצא את הCACHE של שרת הDNS לקובץ הdump המיועד שלנו על ידי הפקודה הבאה בטרמינל:

sudo rndc dumpdb -cache

וקיבלנו את התוצאה הבאה:



ניתן לראות שהCACHE נזרק לקובץ המיועד שהגדרנו

כעת נרצה לכבות את ההגנה של שרת הDNS מפני SPOOFING ATTACKS, נעשה זאת על ידי הוספת השורה הבאה:

dnssec-enable no;

בקובץ named.conf.options

לאחר מכן, נרצה להגדיר פורט מקור קבוע אליו נפנה לבקשות DNS Query ונבצע באמצעות השורה הבאה:

query-source port 33333

בקובץ named.conf.options

לבסוף, נרצה לרסט כדי להחיל את כל השינויים שביצענו בהגדרות של שרת הDNS עם השורה הבאה:

sudo service bind9 restart

* סיכום המשימה

הצלחנו לבצע את המשימה.

תחילה יצרנו ZONE להעברת כל שאילות הקשורות לדומיין pinhamiga.com אל כתובת הIP של התוקף.

שנית, ביצענו שינויים להגדרות השרת BIND9 כמו זריקת הcache לקובץ שנבחר, כיבוי dnssec, הגדרת פורט מקור קבוע

ולבסוף נרסט את השרת כדי להחיל את כל השינויים.

כל ההגדרות נשמרו באופן תקין.

במשימה זו למדנו כיצד לבצע הפניית שאילתות באמצעות forward בקובץ הגדרות השרת של bind9.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 3: Configure the Attacker VM

* מבוא:
  + תיאור

במשימה זו נרצה להגדיר ZONE חדש כלומר כתובת חדשה שאנחנו ניצור והנתונים שלה יאוחסנו על השרת DNS הלוקאלי

* + מטרה

להגדיר Zone עם כתובות IP שנבחר וכתובת לכל HOSTNAME שנבחר ולשמור את הנתונים על הlocal dns

* + תוצאה מצופה

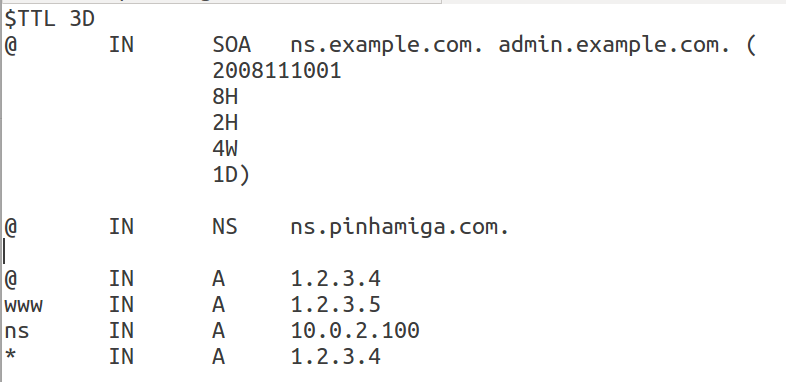
כאשר נבצע dig לכתובת שיצרנו נקבל את כל הנתונים שהגדרנו בZONE על הכתובת.

* ביצוע המשימה:
* Step 1: Download the attacker32.com.zone and example.com.zone files from the lab’s website

הורדנו את הקבצים המשויכים מהאתר.

* ep 2: Modify these files accordingly based on students’ actual network setup (e.g., some IP addresses need to be changed)

ביצענו את השינויים הנדרשים

 תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטי

שינינו את שם הדומיין attacker32.com ל pinhamiga.com

ואת הכתובות IP שהיו לכתובת 10.0.2.100 כדי שכל הhost names שיעברו דרך התוקף.

* Step 3: Copy these two files to the /etc/bind folder

העתקנו את 2 הקבצים אל הנתיב שצוין.

* Step 4: Add the following entries to /etc/bind/named.conf.local:

zone " pinhamiga.com"

{

type master;

file "/etc/bind/ pinhamiga.com.zone";

};

zone "example.com"

{

type master;

file "/etc/bind/example.com.zone";

};

הוספנו בתוקף הפנייה עבור קבצי הקינפוג של הzones אותם הורדנו מseed labs.

* Step 5: Restart the DNS server

לסיום ביצענו ריסטרט לשרת הbind9 כדי להחיל את השינויים.

* סיכום המשימה

הצלחנו לבצע את המשימה.

הגדרנו את מחשב התוקף לפי ההנחיות שקיבלנו.

תחילה נדרשנו להוריד 2 קבצים המכילים הגדרות נוספות עבור הzones בהם נשתמש במעבדה זו.

לאחר מכן, שינינו את ההגדרות כך שיתאימו לתצורת הרשת שלנו והדומיין שעליו אנו נבצע את ההתקפה.

הוספנו את הקבצים אל תיקיית הbind ולבסוף ריסטנו את השרת כדי להחיל את השינויים.

לא נתקלנו בקשיים במשימה זו.

Task 4: Testing the Setup

* מבוא:
  + תיאור

בדיקת הגדרות הקונפיגורציה.

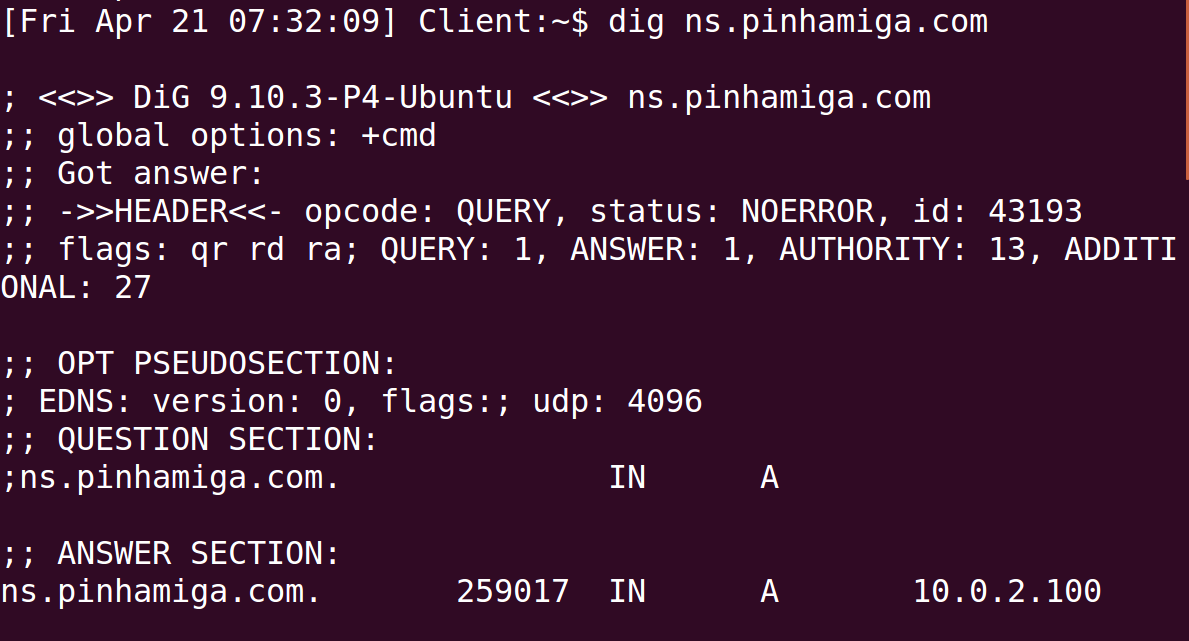
* + מטרה

במשימה זו נרצה לוודא שההגדרות שביצענו נכונות ושהDNS Poisoning בוצע בהצלחה.

* + תוצאה מצופה

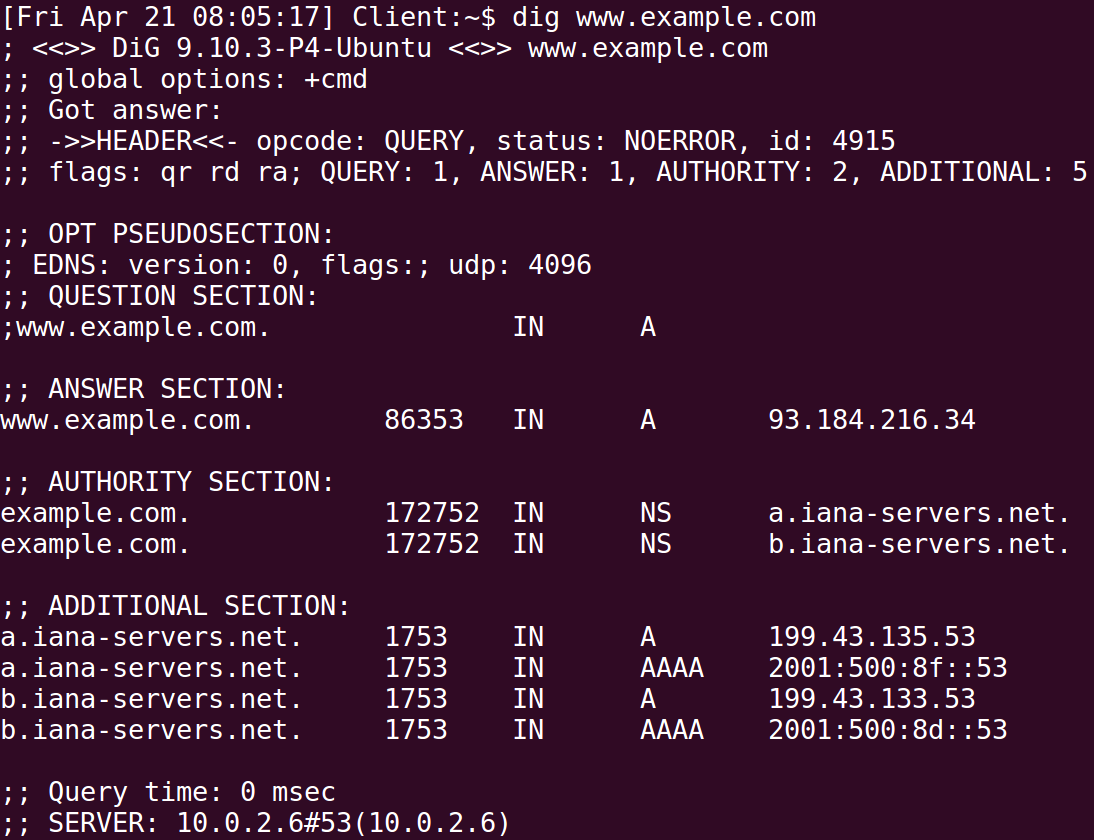
קבלת הIP של התוקף כאשר נשלח dig לדומיינים pinhamiga ו example דרך הלקוח.

* ביצוע המשימה:

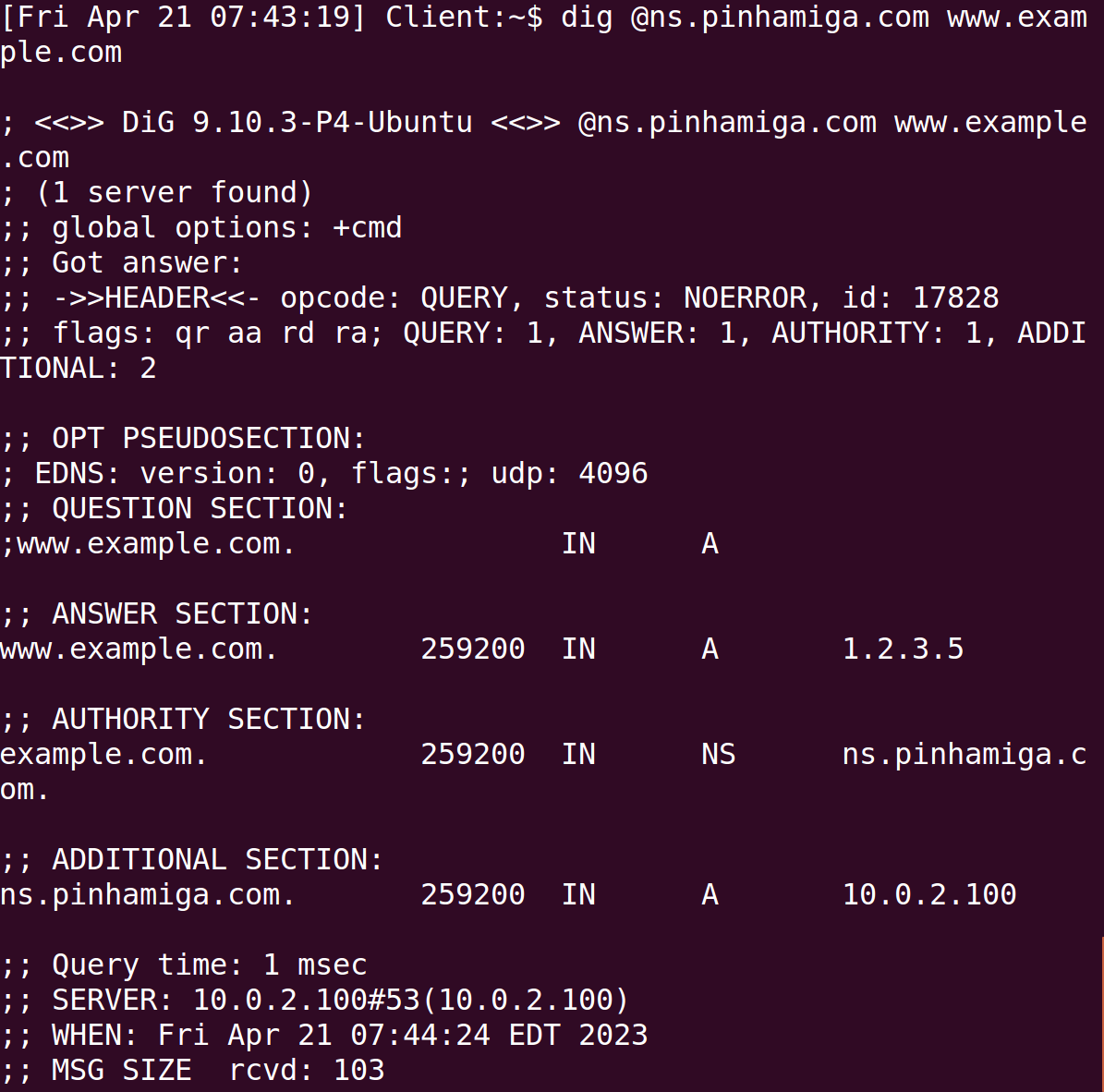
תחילה נבדוק בעת ביצוע פקודת dig על הדומיין של התוקף שנקבל את הIP של התוקף

ניתן לראות כי אכן הגדרנו נכון וקיבלנו את הIP של התוקף 10.0.2.100.

בבדיקה השנייה התבקשנו לבצע 2 פעולות dig.

הראשונה היא dig על [www.example.com](http://www.example.com) אשר יפנה אל השרת DNS הרשמי שלו ונוודא שהתשובה מגיעה מהשרת DNS המקומי שלנו.

ניתן לראות שהעברנו את הבקשה דרך הIP של שרת הDNS המקומי שלנו וקיבלנו את הIP המקורי של [www.example.com](http://www.example.com)

הבדיקה השנייה היא לבצע dig על [www.example.com](http://www.example.com) אשר יפנה אל השרת DNS המקומי דרך הדומיין pinhamiga שמוגדר עם הIP של התוקף ונוודא שהIP שהתקבל הוא הIP של התוקף.



ניתן לראות כי שרת הDNS המקומי שלנו באמת מעביר את הבקשה אל השרת המרוחק של התוקף ומביא לנו את הIP הזדוני שהגדרנו עבור האתר.

@ - מנתב את הבקשה דרך שרת הDNS שנבקש

* סיכום המשימה

הצלחנו לבצע את המשימה.

הגדרנו נכונה את כלל הדרישות לטובת הקונפיגורציה של המכונות.

ראינו שקיבלנו עבור שרת הDNS של pinhamiga את הIP של התוקף ועבור האתר [www.example.com](http://www.example.com) קיבלנו את הIP של התוקף כאשר העברנו את הבקשה דרך השרת שלו.

גילינו כיצד לנתב בקשות דרך שרת ספציפי (בעזרת התו @).

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 4: Construct DNS request

* מבוא:
  + תיאור

במשימה זו נרצה לגרום לשרת הDNS המקומי לשלוח בקשה לבירור כתובת IP של דומיין.

* + מטרה

להמציא כתובת שבוודאות לא נמצאת בזיכרון cache עבור דומיין מסוים כך שנגרום לשרת DNS המקומי לשלוח בקשת DNS QUERY.

* + תוצאה מצופה

שרת הDNS המקומי ישלח את הבקשה לשרת DNS AUTHORITY של אותו הדומיין.

* ביצוע המשימה:

נשלח פקט שתגרום לשרת הDNS המקומי לשלוח בקשה לDNS חיצוני לגבי דומיין כלשהו.

לצורך שליחת הפקט ניעזר בקוד הבא:

#!/usr/bin/python3

from scapy.all import \*

Qdsec = DNSQR(qname='dsfgsdfg.example.com')

dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0,

arcount=0, qd=Qdsec)

ip = IP(dst='10.0.2.6', src='10.0.2.100')

udp = UDP(dport=53, sport=33333, chksum=0)

request = ip/udp/dns

send(request)

האזנו לפקט ששלחנו בעזרת wireshark

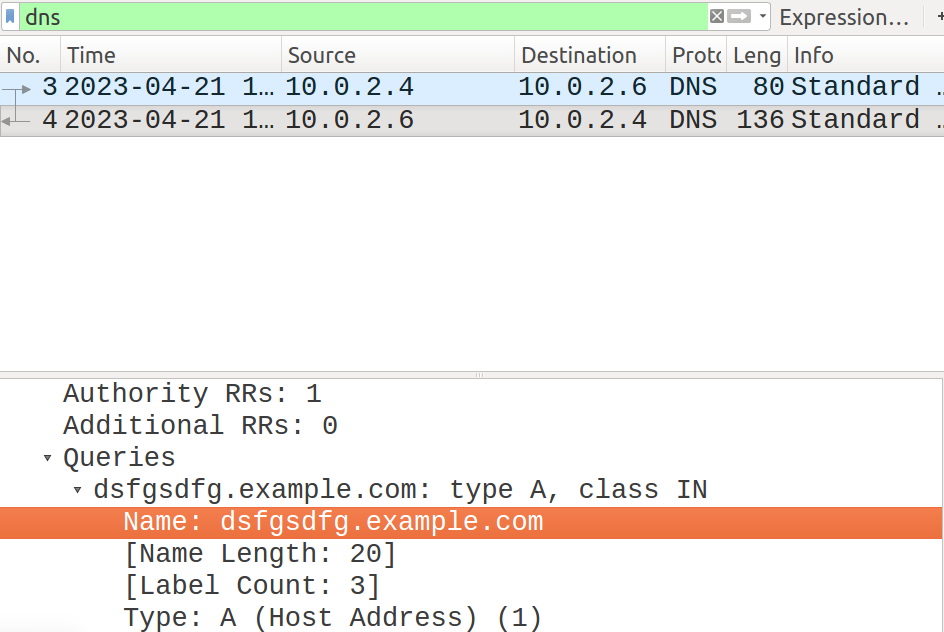
תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי



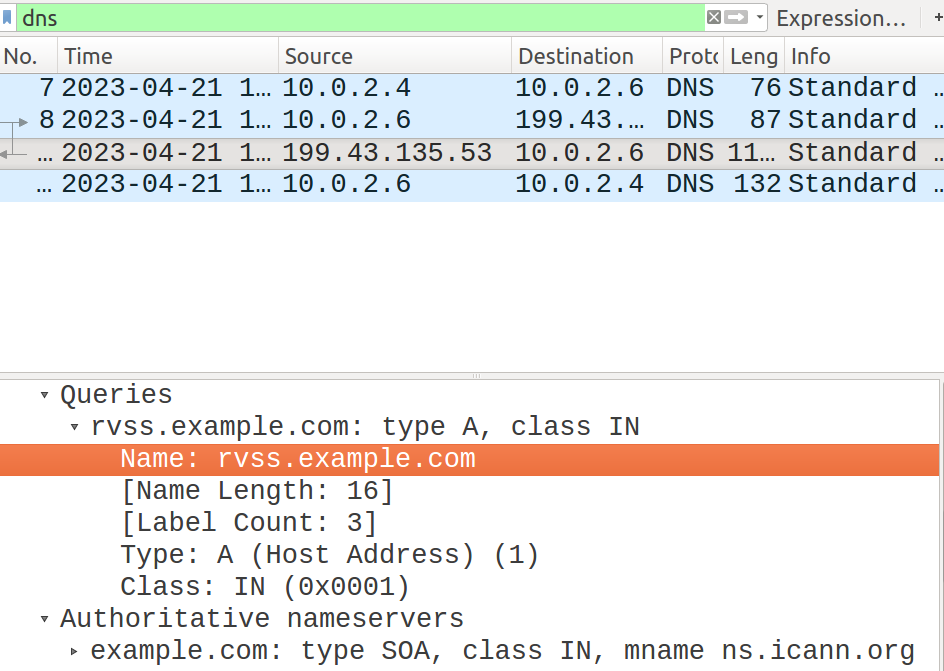
ניתן לראות שהצלחנו מאחר ושרת הDNS המקומי שלח בקשה לבירור הכתובת של האתר אותו רצינו לDNS חיצוני.

בנוסף, ביצענו בדיקה וזייפנו פקט מהלקוח אל שרת הDNS המקומי:



רואים כי השרת לא פונה שוב אל השרת המרוחק אלא נותן תשובה באופן מידי מפני שהכתובת כבר שמורה בcache.

עם זאת, רצינו לוודא שאנחנו גם מצליחים לגרום לשרת DNS המקומי לבצע בדיקת DNS QUERY נוספים:



כפי שניתן לראות, אנו אכן מצליחים לגרום לשרת לשלוח בקשות DNS לשרת מרוחק.

* סיכום משימה

הצלחנו לבצע את המשימה.

שלחנו פקט שתיצור DNS QUERY עבור כתובת דומיין רנדומלית שבחרנו אל השרת DNS המקומי והראינו שהשרת DNS המקומי מעביר את הבקשה אל שרת DNS מרוחק.

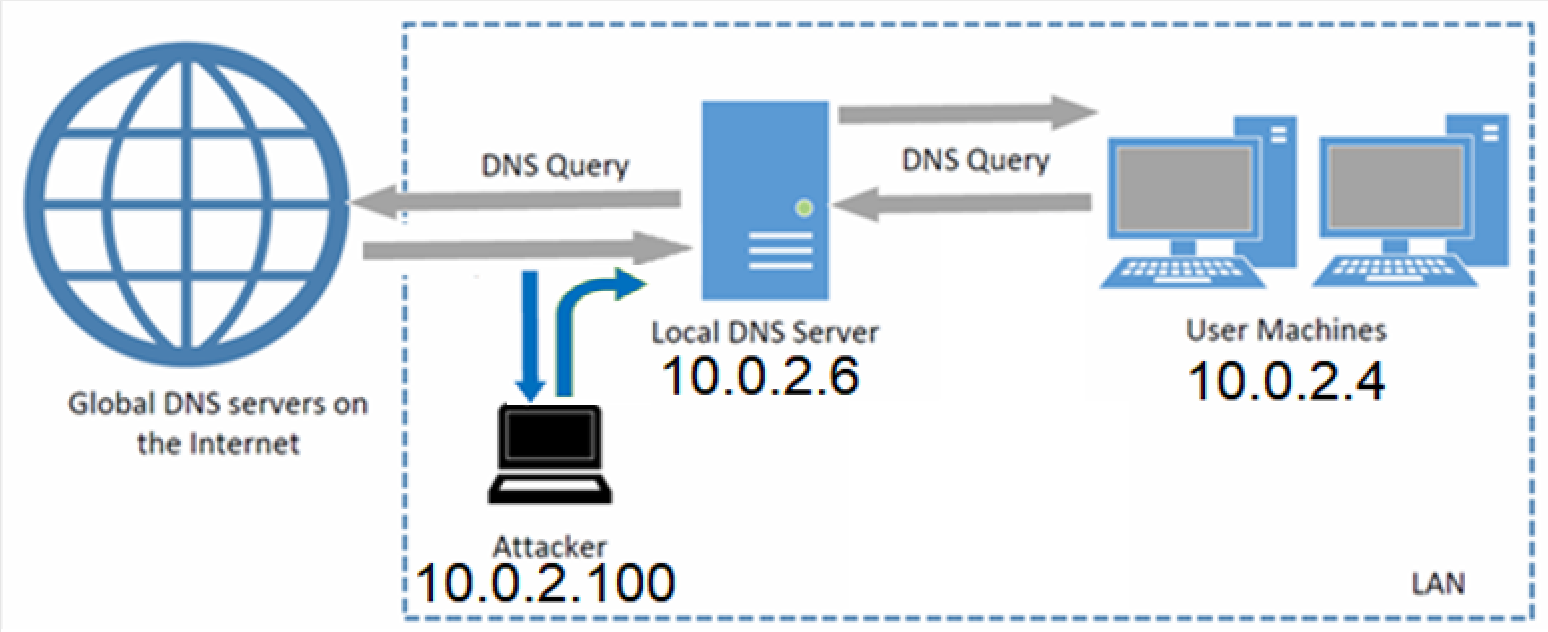
גילינו כיצד ניתן לגרום לשרת הDNS המקומי לשלוח בקשת DNS למרות שיש לו את המידע על אותו הדומיין כבר בזיכרון הCACHE.

ראינו שהצלחנו לבצע את ההרעלה בכך שבפעם השנייה לא נשלחה בקשה לשרת מרוחק אלא נשלפה מהזיכרון.

לבסוף, ביצענו את הבדיקה עבור כתובת נוספת בדומיין וראינו שגרמנו לבקשת DNS לשרת DNS חיצוני בהצלחה.

התוצאות התאימו למצופה מאחר והבקשה ששלחנו לשרת המקומי נשלחה ממנו הלאה לשרת חיצוני למרות שהמידע היה שמור לו בזיכרון עבור אותו הדומיין.

Task 5: Spoof DNS Replies

* מבוא:
  + תיאור

במשימה זו נרצה לבצע זיוף מענה לשאילתת בקשת DNS שתענה מהכתובת של השרת DNS המקורי עם הכתובת המזויפת שנכניס לה.

* + מטרה

ליצור פקט מזויף שיענה לשאילתת DNS עם הNS המקורי כאשר הIP שלו יהיה הIP של התוקף וגם הכתובת של האתר תהיה עם הIP של התוקף.

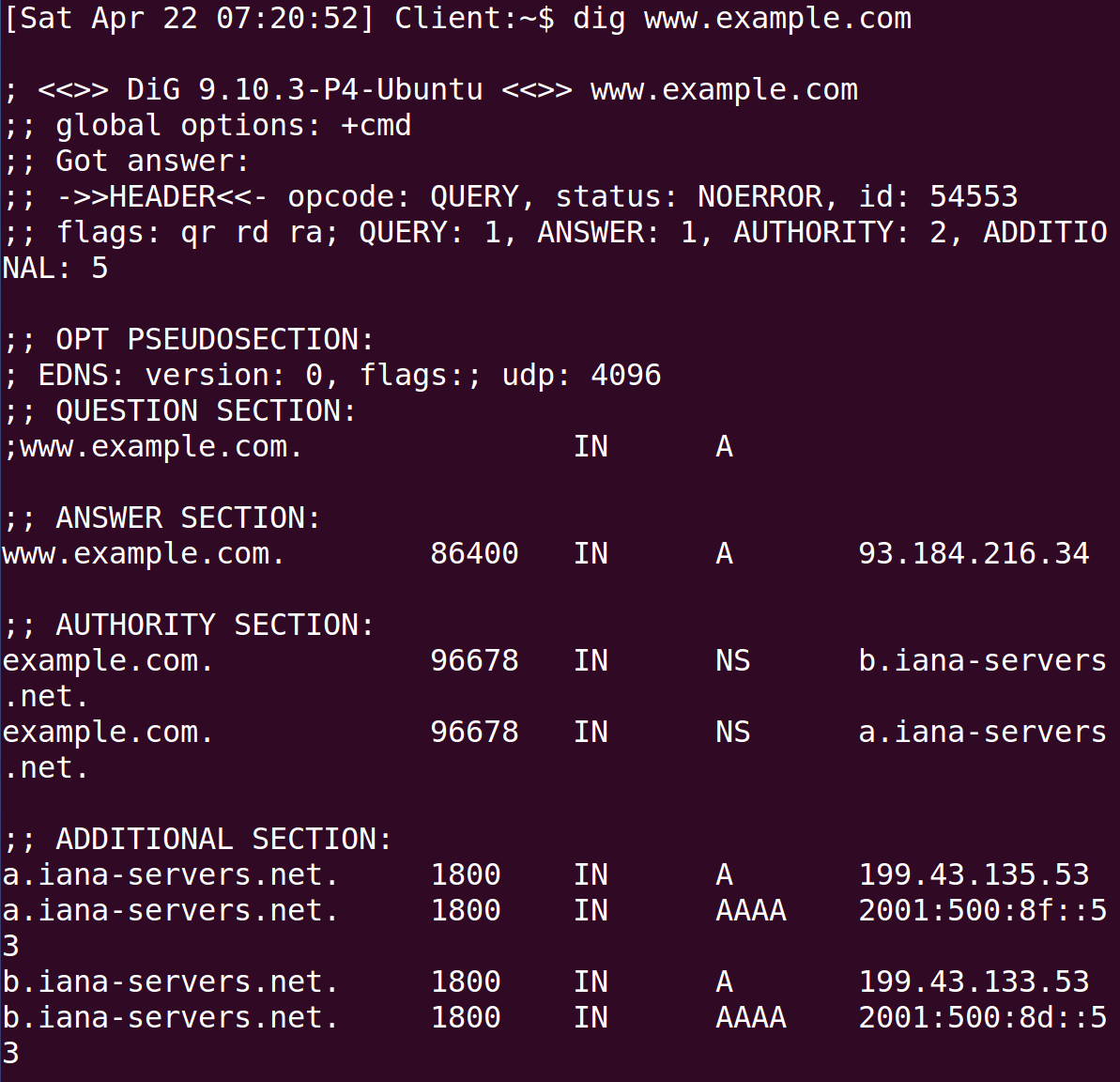
* + תוצאה מצופה

לראות בwireshark שהפקט הועבר עם המידע הרלוונטי.

* ביצוע המשימה:

נשלח פקט מזויף למענה לDNS QUERY כאשר נשלח אותו בשם ה authority server ונעדכן את הכתובות IP המתקבלות עבור ה host name הנדרש ועבור ה ns שיהיו הכתובת IP של התוקף.

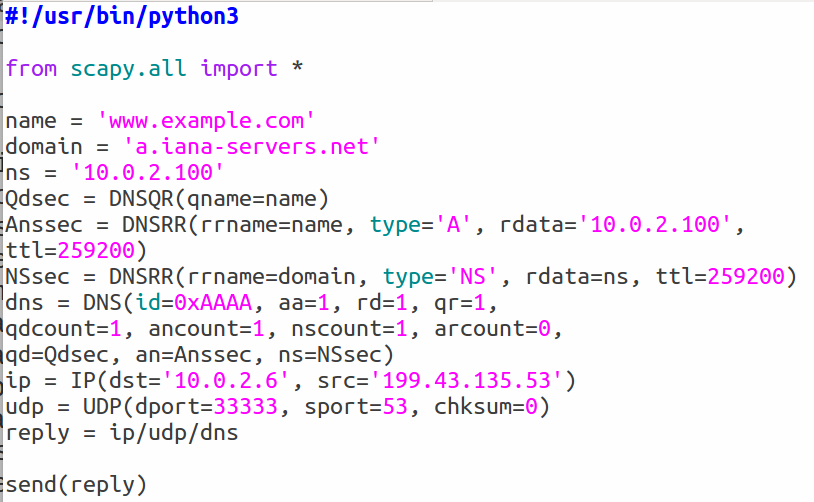
לפני שנבצע את המשימה נרצה לדעת מה השרת authority המקורי ומה הIP שלו עבור כתובת האתר [www.example.com](http://www.example.com)





בתמונה ניתן לראות את הפרטים שנצטרך לטובת בניית הפקט.

נשתמש בקטע הקוד הבא:



בקוד ניתן לראות כי הגדרנו את השם של האתר עליו נבצע את המתקפה, נגדיר את שם הדומיין שאמור לתת את התשובה על כתובת האתר ומה הIP אליו נרצה להפנות בקשות עתידיות.

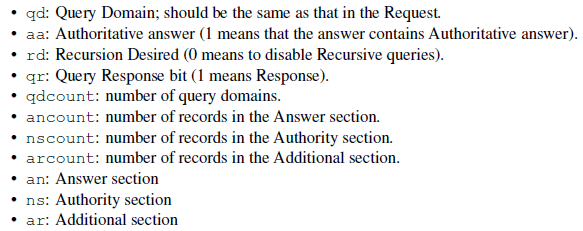
בסקטור התשובה הגדרנו את הIP של התוקף עבור כתובת האתר.

בסקטור הNS הגדרנו את הדומיין המקורי אבל הוא מוגדר עם הIP של התוקף.

באובייקט הDNS הגדרנו aa=1 שמציין שיועבר גם מידע על הauthority, rd=1 עבור תשובה רקורסיבית, qr=1 עבור שליחת מענה לשאילתת DNS, qdcount=1 עבור מספר הדומיינים שנשאלים עליהם, ancount=1 כמות הרשומות ביחידת התשובה, nscount=1 כמות הרשומות ביחידת הauthority, arcount=0 כמות הרשומות ביחידת הadditional.

לאחר מכן השלמנו את בניית הפקט בין שרת הDNS המקומי אל השרת המרוחק ושלחנו את הפקט.

פירוט קצר על יצירת dns packet:



ביצירת הפקט הcheck sum שווה 0 כדי לתת למערכת הפעלה לחשב אותה באופן אוטומטי

לאחר שליחת הפקט, ראינו בwireshark שאכן נשלח פקט מהשרת DNS המרוחק אל השרת המקומי.

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי



ניתן לראות שהצלחנו ואכן קיבלנו תשובה מהשרת המרוחק עם הנתונים המזויפים אותם הזנו וסימנו בתמונה באדום.

* סיכום המשימה

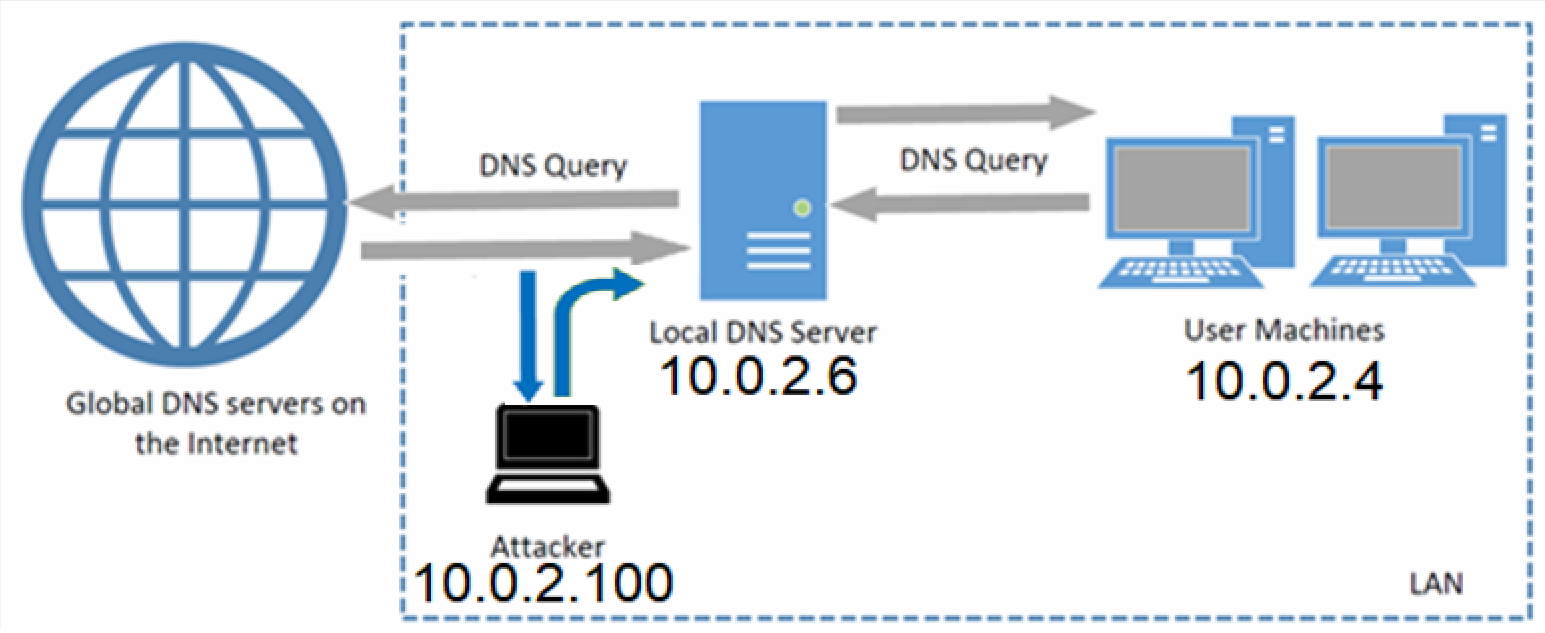
הצלחנו לבצע את המשימה.

שלחנו פקט מהתוקף בשם השרת המרוחק אל השרת המקומי עם הנתונים המזויפים עבור כתובת האתר המבוקש ועבור ns האחראי על אותה הכתובת.

הוכחנו שהפקט נשלח עם הנתונים המזויפים ע"י כך שהסנפנו אותה על גבי הרשת בעזרת הwireshark.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 6: Launch the Kaminsky Attack

* מבוא:
  + תיאור

נבצע מתקפה שמרעילה את זיכרון הcache של הDNS המקומי

* + מטרה

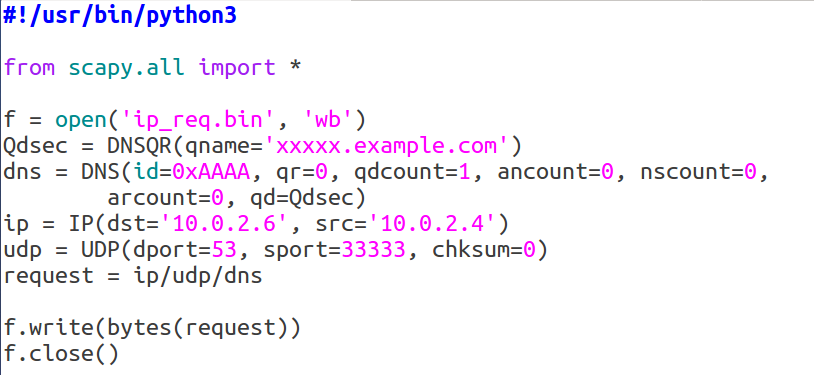
לספק מענה מהיר לשאילתת DNS QUERY אשר נשלח מהDNS המקומי לפני שהDNS החיצוני מספיק לענות לו.

* + תוצאה מצופה

זיכרון הcache של הDNS המקומי יורעל ובכל בקשת DNS QUERY עבור הדומיין שנבחר, נקבל את המידע המזויף שהשתלנו לו בזיכרון.

* ביצוע המשימה:

עדכנו את קטעי הקוד הקודמים כך שנוכל לשמור אותם בקובץ ולהריץ בצורה היברידית עם קוד בשפת C כדי שנוכל לשלוח פקטות באופן מהיר.



קטע הקוד מעלה, אחראי על ייצוא בקשת הDNS לקובץ.

לאחר מכן נרצה גם לעדכן את הקוד לתשובת הDNS

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

בקטע הקוד מעלה, אחראי על ייצוא תשובת הDNS לקובץ.

לאחר הרצת קטעי הקוד, נרצה לוודא שהקבצים נוצרו בהצלחה:

תמונה שמכילה טקסט, צילום מסך, גופן, תרשים

התיאור נוצר באופן אוטומטי

ניתן לראות שנוצרו 2 קבצים חדשים : ip\_req.bin, ip\_resp.bin בהצלחה.

לאחר מכן נרצה ליצור קובץ בשפת C שיריץ את קבצי הפייתון שהכנו מראש:

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

עד כאן, קובץ הקוד בC המלא.

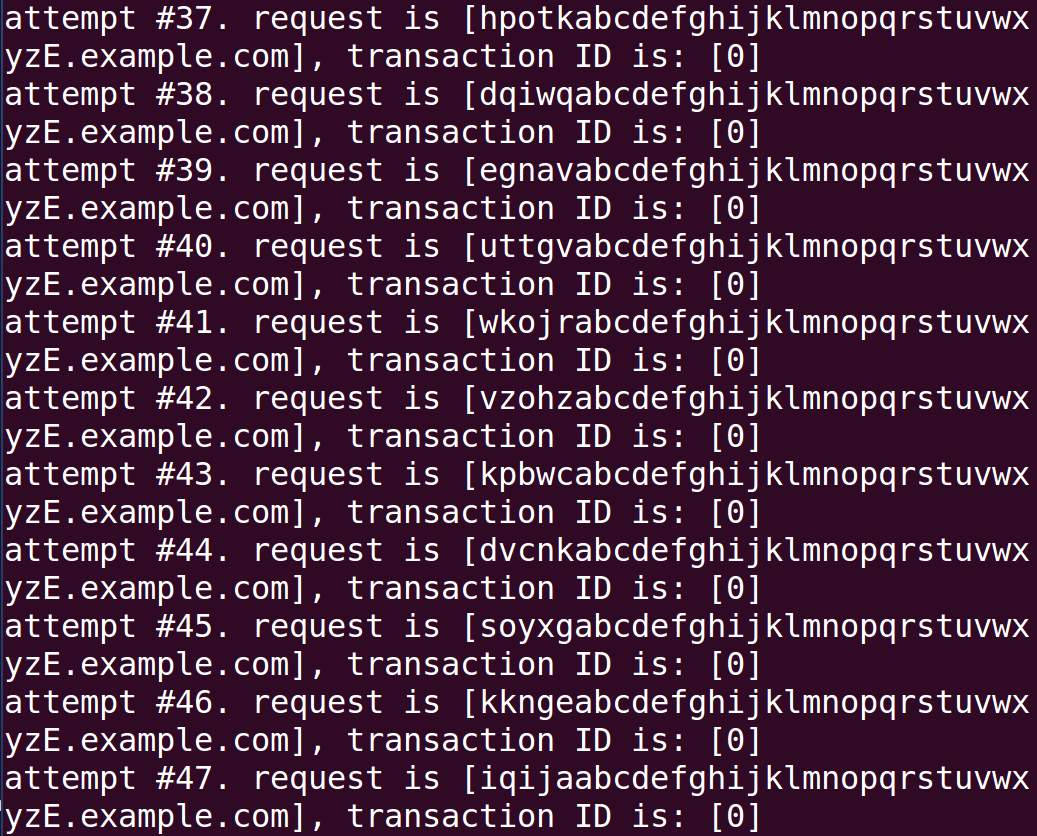
כעת נבצע הרצה לכל קטעי הקוד אחד אחרי השני.

תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

נבצע את ההתקפה הראשית עם הקובץ בC כאשר נשתמש בפרמטר -o לתת כינוי לקובץ הרצה.

כעת נראה על מסך הטרמינל את הבקשות שנשלחות אל שרת הDNS המקומי

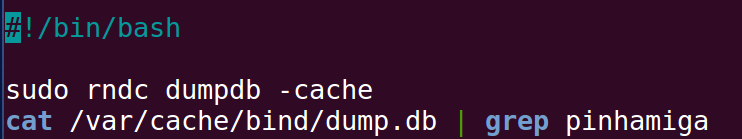


ניתן לראות שהיינו צריכים לבצע מספר רב של ניסיונות הרעלה של הזיכרון.

לאחר מכן, לטובת הבדיקה להצלחת ההתקפה ניצור קובץ bash.



הקובץ מכיל את 2 הפקודות הבאות:



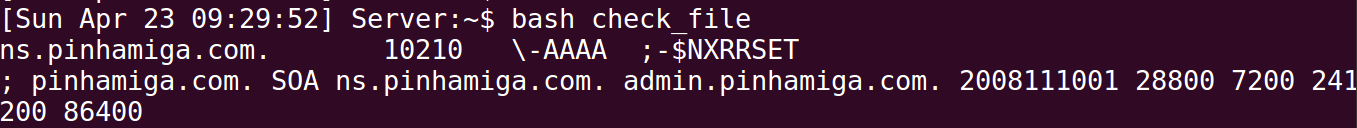
הפקודה הראשונה יוצרת קובץ עם הזיכרון cache

הפקודה השנייה תחתוך מהטקסט בקובץ את השורה הרלוונטית המכילה את המילה pinhamiga.

נריץ את הסקריפט:



נראה האם הצלחנו במשימה:



ניתן לראות כי באמת הצלחנו לשתול מידע מזויף בזיכרון הcache של שרת הDNS המקומי.

* סיכום המשימה

הצלחנו לבצע את המשימה.

במשימה זו יצרנו 3 קטעי קוד עבור מטרות שונות, כאשר שילבנו בצורה היברידית 2 שפות תכנות.

SCAPY לשם הנוחות ושפת C עבור מהירות הביצוע.

בעזרת הקוד שלחנו פקטות מרובות אשר גורמות לDNS המקומי לשלוח שאילתת DNS QUERY ועל ידי כך הקוד שכתבנו יוכל גם לנסות לענות לשאילתות האלו ולהרעיל את זיכרון הcache של הDNS המקומי.

הוכחנו שהצלחנו ע"י כך שהופיע בזיכרון הcache של השרת המקומי שורה המכילה מידע על הדומיין המזויף שהגדרנו pinhamiga.

גילינו כיצד לשלב בין 2 שפות תכנות, והיתרונות והחסרונות של כל שפה.

בנוסף, גילינו שלא פשוט להרעיל זיכרון DNS, שלא על הרשת המקומית שלנו.

התוצאות התאימו למצופה מאחר וראינו שקיים מידע על הדומיין המזויף בזיכרון של השרת המקומי.

נתקלנו בבעיות בזמן ביצוע המשימה כשלא ידענו איך לכתוב בשפת C ולהפוך את הקוד בפייתון לקבצים בינאריים ולשלב אותם בקוד C.

נעזרנו בגוגל כדי לפתור את הבעיה אך בזמן החיפוש הבנו גם שלא בהכרח

נצליח בהתקפה ולא ניתן לדעת כמה ניסיונות נצטרך לבצע כדי שזה יצליח.

Task 7: Result Verification

* מבוא:
  + תיאור

בדיקת הצלחת ההרעלה של הזיכרון של שרת הDNS המקומי

* + מטרה

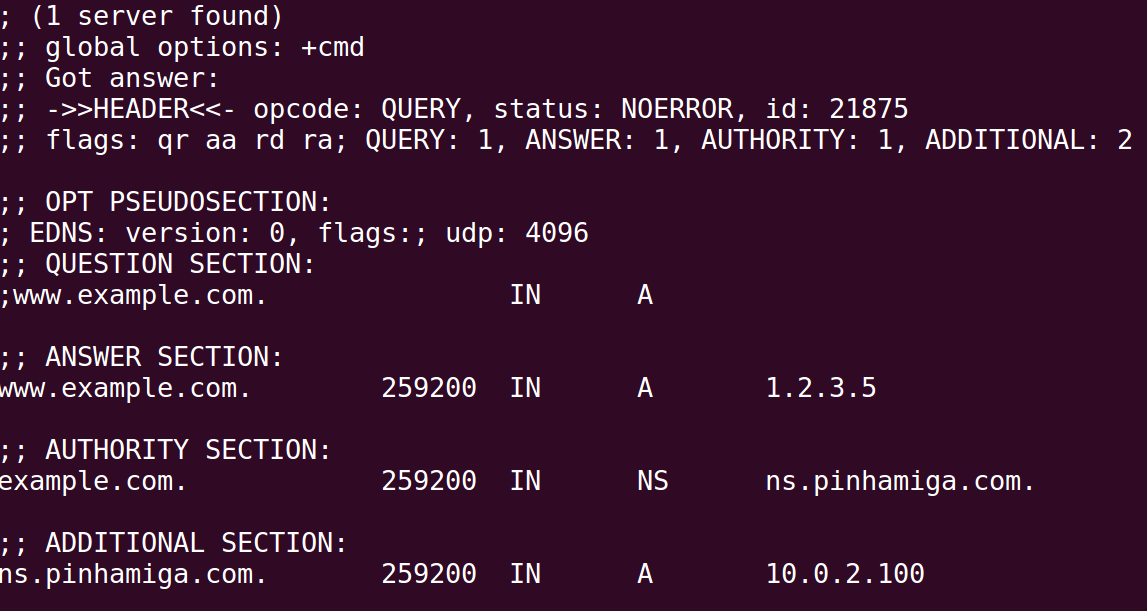
לבדוק שהזיכרון cache של שרת הDNS המקומי הורעל

* + תוצאה מצופה

בעת ביצוע dig [www.example.com](http://www.example.com) שאמור להביא לנו את הכתובת הרשמית של האתר והפרטים הנוספים שלו, נקבל במקום את הפרטים המזויפים אותם השתלנו.

* תמונה שמכילה טקסט, גופן, צילום מסך

  התיאור נוצר באופן אוטומטיביצוע המשימה:

תחילה נבצע במחשב הלקוח את הפקודה הבאה:



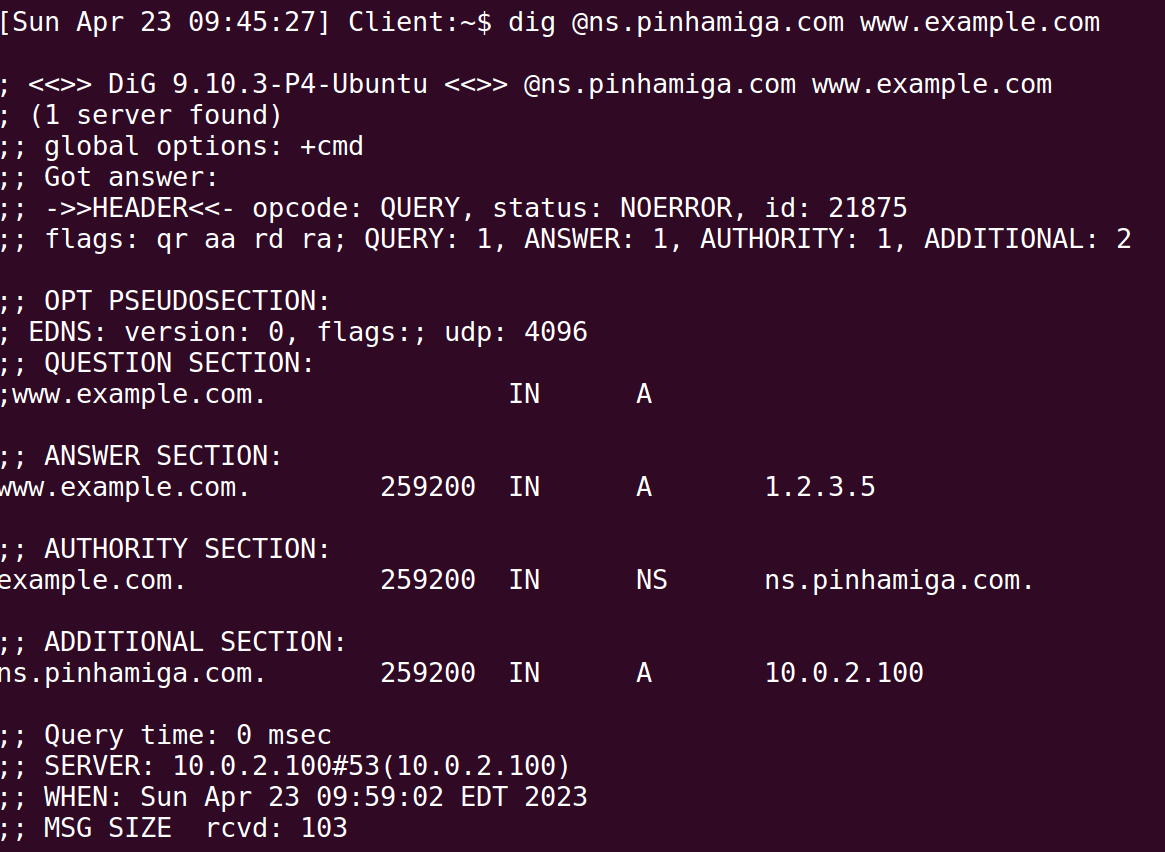
תמונה שמכילה טקסט, גופן, צילום מסך, טיפוגרפיה

התיאור נוצר באופן אוטומטי



ניתן לראות כי קיבלנו את שרת הNS של pinhamiga.com עם הIP של התוקף, הפנינו לIP זדוני בעת כניסה לאתר [www.example.com](http://www.example.com) והשרת שהעביר לנו את התשובה זה שרת הDNS המקומי.

כלומר, הרעלנו את הזיכרון שלו והוא זוכר את הפרטים המזויפים.

ביצענו dig שעובר דרך הNS של pinhamiga:

ניתן לראות שהתשובות שהתקבלו ב2 פקודות הdig הן זהות מה שמוכיח על הצלחתנו בהרעלת זיכרון הcache של שרת הDNS המקומי.סיכום המשימה

הצלחנו לבצע את המשימה.

המידע נשמר בזיכרון הCACHE.

הוכחנו זאת בכך שקיבלנו את אותן התשובות המזויפות בעת פקודת DIG לאתר www.example.com

לא גילינו משהו חדש ולא נתקלנו בבעיות.

התוצאות התאימו למצופה מאחר שבביצוע פקודת הDIG לאתר [www.example.com](http://www.example.com) קיבלנו IP של האתר הזדוני אותו השתלנו.

**סיכום כללי למעבדה**

בתחילת המעבדה נדרשנו להגדיר את מחשב 10.0.2.6 בתור local dns, וחיברנו את הclient ל10.0.2.6 כך שכל שאילתות הDNS יעברו דרכו.

היינו צריכים להראות בעזרת הפקודה DIG שאכן כל השאילתות DNS נשלחות לlocal dns שהגדרנו שהוא 10.0.2.6.

לאחר מכן, יצרנו ZONE להעברת כל שאילות הקשורות לדומיין pinhamiga.com אל כתובת הIP של התוקף.

שנית, ביצענו שינויים להגדרות השרת BIND9 כמו זריקת הcache לקובץ שנבחר, כיבוי dnssec, הגדרת פורט מקור קבוע ולבסוף ריסטנו את השרת כדי להחיל את כל השינויים. בדקנו שכל ההגדרות נשמרו באופן תקין.

בהמשך נדרשנו להוריד 2 קבצים המכילים הגדרות נוספות עבור הzones בהם נשתמש במעבדה זו.

שינינו את ההגדרות כך שיתאימו לתצורת הרשת שלנו והדומיין שעליו אנו נבצע את ההתקפה.

הוספנו את הקבצים אל תיקיית הbind ולבסוף ריסטנו את השרת כדי להחיל את השינויים.

בדקנו שאנחנו מקבלים עבור שרת הDNS של pinhamiga את הIP של התוקף ועבור האתר [www.example.com](http://www.example.com) קיבלנו את הIP של התוקף כאשר העברנו את הבקשה דרך השרת שלו.

יצרנו פקט שישלח DNS QUERY עבור כתובת דומיין רנדומלית שבחרנו אל השרת DNS המקומי והראינו שהשרת DNS המקומי מעביר את הבקשה אל שרת DNS מרוחק.

ראינו שהצלחנו לבצע את ההרעלה בכך שבפעם השנייה לא נשלחה בקשה לשרת מרוחק אלא נשלפה מהזיכרון.

לבסוף, ביצענו את הבדיקה עבור כתובת נוספת בדומיין וראינו שגרמנו לבקשת DNS לשרת DNS חיצוני בהצלחה.

רצינו גם לשלוח פקט מהתוקף בשם השרת המרוחק אל השרת המקומי עם הנתונים המזויפים עבור כתובת האתר המבוקש ועבור ns האחראי על אותה הכתובת.

במשימה האחרונה היינו צריכים לבצע את המתקפה המלאה שנקראת Kaminsky attack. במשימה זו יצרנו 3 קטעי קוד כאשר 2 מהם מהמשימות הקודמות וקטעי הקוד כתובים בשפות שונות.

SCAPY לשם הנוחות ושפת C עבור מהירות הביצוע.

בעזרת הקוד שלחנו פקטות מרובות אשר גורמות לDNS המקומי לשלוח שאילתת DNS QUERY ועל ידי כך הקוד שכתבנו יוכל גם לנסות לענות לשאילתות האלו ולהרעיל את זיכרון הcache של הDNS המקומי.

לסיכום,

ביצענו בהצלחה את כל שלבי המעבדה. הצלחנו לבצע את מתקפת קמינסקי ולראות כיצד מרעילים DNS מרוחק שלא נמצא איתנו ברשת.

**משהו חדשני – Heimdal:**

Heimdal Security מציעה פתרון סינון פקטות DNS המגן מפני התקפות DNS POISONING על ידי חסימת גישה לשרתי DNS זדוניים וסינון תעבורת DNS זדונית.

ניתן ליישם פתרונות אלה באמצעות אלגוריתמים, למידת מכונה, זיהוי תבניות, הגדרת חומרה או בצורה תוכנתית או בשכבת הרשת.

הפתרון של Heimdal פועל על ידי הפניית תעבורת DNS מרשת הארגון לשרתי ה-DNS המאובטחים של Heimdal Security, אשר הוגדרו מראש עם רשימה של שרתי DNS המוכרים כבטוחים.כאשר נשלחת בקשת DNS מרשת הארגון, הבקשה נבדקת תחילה מול רשימת שרתי ה-DNS של Heimdal. אם שרת ה-DNS המבוקש אינו ברשימה, הבקשה נחסמת והמשתמש מופנה לדף אזהרה, דבר אשר מונע מהמשתמש להתחבר באופן לא מודע לשרת DNS זדוני שיכול לשמש לצורך הפניה לאתר פישינג או לתוכן זדוני אחר.

בנוסף לחסימת תעבורת DNS זדונית, פתרון סינון ה-DNS של Heimdal Security מספק גם מודיעין וניטור איומים בזמן אמת, כמו גם דיווח וניתוח מתקדמים כדי לעזור לארגונים לעדכן את עמדות האבטחה שלהם ולזהות במהירות התקפות ומתן יכולת תגובה לכל האיומים הפוטנציאליים.