**מטלה 4 - Local DNS Attack Lab**

גיא כץ - 322977265

אברהם דבורה – 315025718

# תוכן עניינים

[Task 2.1: Configure the User VM](#_Task_2.1:_Configure)

[2.1.2 ביצוע המשימה](#_Task_2.1:_Configure)

[2.1.3 סיכום המשימה](#_2.1.3_סיכום_המשימה)

[Task 2.2: Set up a Local DNS Server](#_Task_2.2:_Set)

[2.2.1 מבוא](#_2.2.1_מבוא:)

[2.1.2 ביצוע המשימה](#_2.1.2_ביצוע_המשימה)

[2.1.3 סיכום המשימה](#_2.1.3_סיכום_המשימה)

[Task 2.3: Configure the Attacker VM](#_Task_2.3:_Configure)

[2.3.1 מבוא](#_2.3.1_מבוא:)

[2.3.2 ביצוע המשימה](#_2.3.2_ביצוע_המשימה:)

[2.3.3 סיכום המשימה](#_2.3.3_סיכום_המשימה:)

[Task 2.4: Testing the Setup](#_Task_2.4:_Testing)

[2.4.1 מבוא](#_2.4.1מבוא:)

[2.4.2 ביצוע המשימה](#_2.4.2_ביצוע_המשימה:)

[2.4.3 סיכום המשימה](#_2.4.3_סיכום_המשימה)

[Task 3.4: Construct DNS request](#_Task_3.4:_Construct)

[3.4.1 מבוא](#_3.4.1_מבוא)

[3.4.2 ביצוע המשימה](#_3.4.2_ביצוע_המשימה:)

[3.4.3 סיכום המשימה](#_3.4.3_סיכום_המשימה:)

[Task 3.5: Spoof DNS Replies](#_Task_3.5:_Spoof)

[3.5.1 מבוא](#_3.5.1_מבוא:)

[3.5.2 ביצוע המשימה](#_3.5.2ביצוע_המשימה:)

[3.5.3 סיכום המשימה](#_3.5.3_סיכום_המשימה:)

[Task 3.6 Launch the Kaminsky Attack.](#_Task_3.6_Launch)

[3.6.1 מבוא](#_3.6.1_מבוא:)

[3.6.2 ביצוע המשימה](#_3.6.2_ביצוע_המשימה:)

[3.6.3 סיכום המשימה](#_3.6.3_סיכום_המשימה:)

[Task 3.7: Result Verification](#_Task_3.7:_Result)

[3.7.1 מבוא](#_3.7.1_מבוא:)

[3.7.2 ביצוע המשימה](#_3.7.2_ביצוע_המשימה:)

[3.7.3 סיכום המשימה](#_3.7.3_סיכום_המשימה:)

[סיכום המעבדה](#_סיכום_המעבדה)

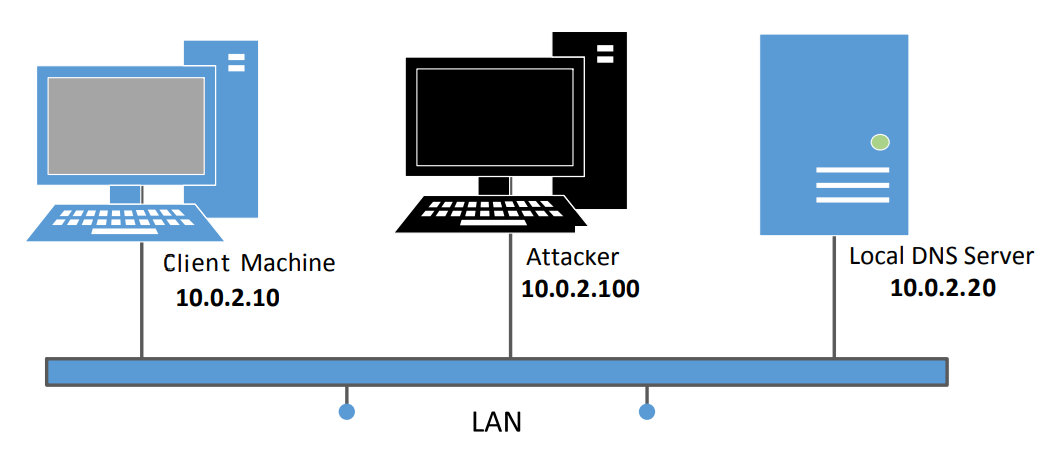
[לימוד מעבר לדרישות המעבדה](#_לימוד_מעבר_לדרישות)

# Task 2.1: Configure the User VM

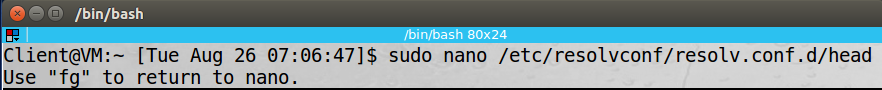
# 2.1.1 מבוא:

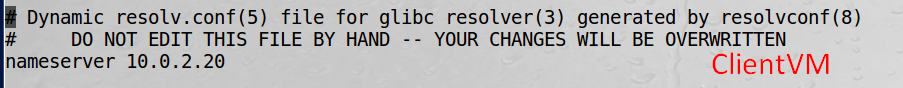
בחלק זה של המטלה אנחנו רוצים להגדיר את מכונת הClient כך ששרת הDNS שהיא משתמשת בו יהיה הServer שאנחנו הגדרנו, כדי שנוכל לבצע התקפות מאוחר יותר בלי להיכנס לכלא.

נצפה להגדיר כמה קבצים ולרשום את הServer בראש הרשומות וכך הוא יהיה השרת הראשי.

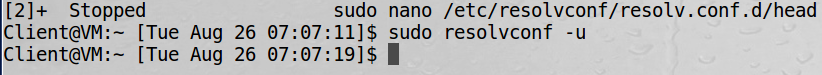
שרטוט הרשת: (שנשתמש בו לאורך המטלה)

# 2.1.2 ביצוע המשימה

נגדיר בClient את הServer כDNS Server ע"י הגדרתו כnameserver בשורה הראשונה:

נכנסים לקובץ:

רואים שהוספנו את הServer.

נריץ את הפקודה הבאה כדי לגרום לשינויים להתבצע:

אחרי הרצת הפקודה השרת אמור להיות מוגדר.

נבצע Dig לכתובת אינטרנט כלשהי (google.com) כדי לראות אם שרת הDNS מוגדר כמו שצריך:



אכן אפשר לראות שהClient השתמש בDNS Server של Server, אז ההגדרה הצליחה. אפשר לראות את זה גם בWireshark, שנדביק במטלה הבאה.

# 2.1.3 סיכום המשימה

בחלק זה של המטלה בסה"כ הגדרנו את הClient, כך שהוא יבקש בקשות DNS מהServer. עבד כמו שציפינו, לא מסובך בינתיים.

# Task 2.2: Set up a Local DNS Server

# 2.2.1 מבוא:

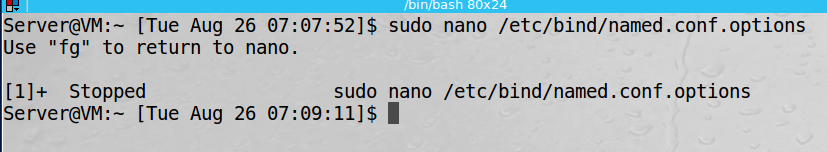
בחלק זה של המטלה אנחנו רוצים להגדיר את הServer כLocal Dns Server.

נשתמש בתוכנה מובנית בשם Bind9 כדי ליצור את השרת, ואז נגדיר דברים נוספים כמו לאן הCache יעשה Dump, או ביטול של הגנה בשם DNSSEC (שבעצם בודקת עם חתימות דיגיטליות את הבקשות)

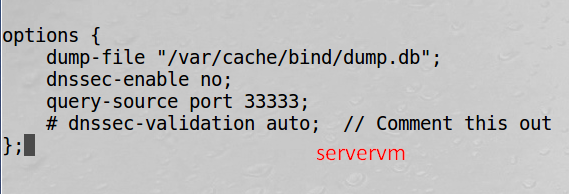
בשונה מהמטלה הקודמת, צריך לשנות את הZones, אז נעשה את זה. כלומר אנחנו צריכים להגדיר Forward Zone כך שברגע ששרת הDNS יפנה לkatzdvora.com הוא יפנה ישירות לIP של הAttacker, וכך נדע שההתקפה הצליחה ונעצור אותה.

לאחר מכן נריץ את הServer מחדש ונראה שהוא עובד.

# 2.2.2 ביצוע המשימה:

נתחיל בלפתוח את ההגדרות של Bind9:

ובתוכן נשנה את ההגדרות שאנחנו רוצים: לבטל DNSSEC, להגדיר מיקום לקובץ הCache, ולהגדיר פורט 33333 לבקשות יוצאות:



אפשר לראות ששינינו בOptions את ההגדרות שרצינו.

בנוסף למה שעשינו, אנחנו צריכים להגדיר Forward Zone כך שברגע ששרת הDNS יפנה לkatzdvora.com הוא יפנה ישירות לIP של הAttacker, וכך נדע שההתקפה הצליחה ונעצור אותה:

תמונה שמכילה טקסט, גופן, קו, מידע

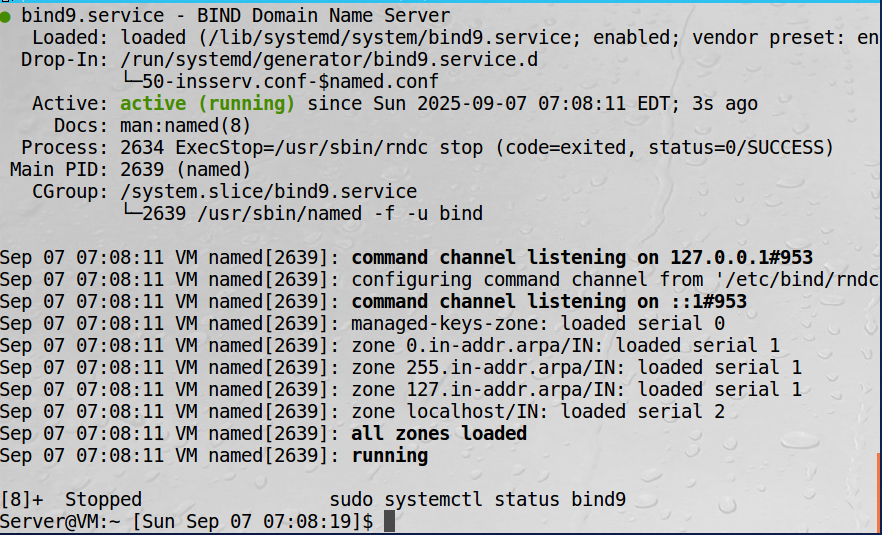
תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.נפתח את קובץ ההגדרה של bind9 בServer:

ובתוכו:

תמונה שמכילה טקסט, צילום מסך, גופן

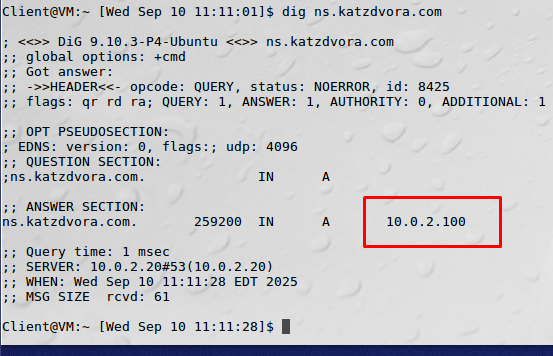
תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

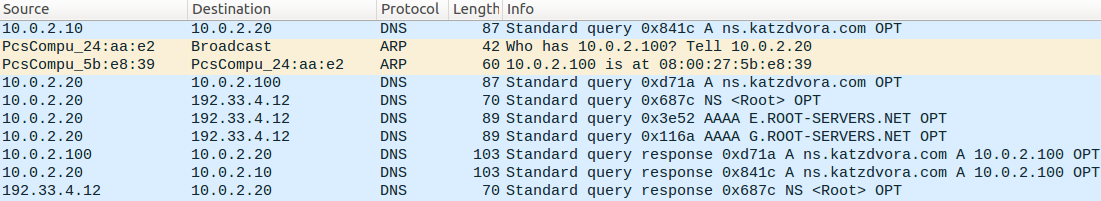
אפשר לראות שהגדרנו את הIP של katzdvora.com להיות הIP של הAttacker.

לאחר מכן נעשה Restart ואז נריץ את השרת ונראה שהוא עובד:

רואים שהServer עובד ומאזין. (עצרנו אותו אחרי בדיקות)

נבצע dig מהClient על ns.katzdvora.com:



אפשר לראות שהClient שולח בקשתDNS לServer ומקבל תגובה.

נראה בwireshark:

אפשר לראות שהServer פונה גם לAttacker וגם לDNS שמוגדר לו, כנראה כדי לחסוך זמן.

# 2.2.3 סיכום המשימה:

בחלק זה ראינו איך להגדיר Local DNS Server, איך לשנות בהגדרות לאיזה קובץ יישמר הCache Dump, ואיך לבטל הגנה בשם DNSSEC כדי שנוכל לתקוף מאוחר יותר.

ראינו גם את שרת הDNS עובד כששלחנו dig מהClient, וראינו גם שהForward Zone שהגדרנו לkatzdvora.com עובד (קיבלנו את 10.0.2.100 בתגובה).

# Task 2.3: Configure the Attacker VM

# 2.3.1 מבוא:

בחלק זה של המטלה אנחנו רוצים להגדיר את שרת הDNS במכונת הAttacker ולהגדיר שם שני Zones: אחד לkatzdvora.com ואחד מזויף לexample.com.

אחרי שאנחנו מגדירים נאפס את השרת.

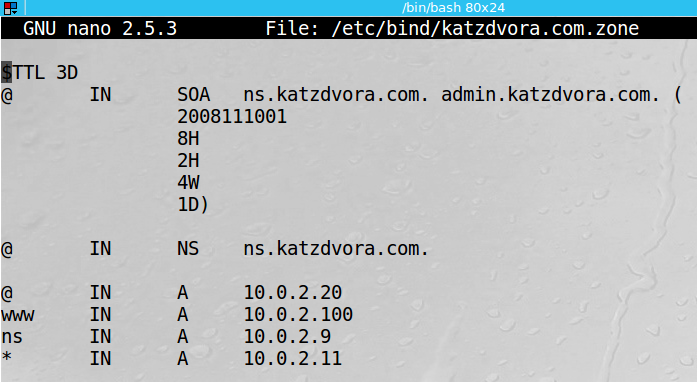
# 2.3.2 ביצוע המשימה:

נתחיל בלפתוח את ההגדרה של השרת בAttacker ולומר לו מאיפה לקחת את הZones:

הגדרנו שני zones כמו שרצינו, עכשיו אנחנו צריכים ליצור ולשנות את הקבצים בהתאם.

הגדרת הקובץ לZone של Example.com בAttacker:

אפשר לראות שהגדרנו Zone עבור Example.com בAttacker בהצלחה.

נעבור להגדיר את הZone של katzdvora.com:

אותו סגנון כמו מקודם, יכלנו גם להשאיר שורה אחת עם כוכבית אבל לא היה לנו קריטי לשנות את זה.

בדקנו שהשרת עובד, לא נראה הוכחות.

# 2.3.3 סיכום המשימה:

בחלק זה הגדרנו את הDNS Server בAttacker עם הZones שלו, לא משהו מעניין במיוחד. לא נתקלנו בבעיות.

# Task 2.4: Testing the Setup

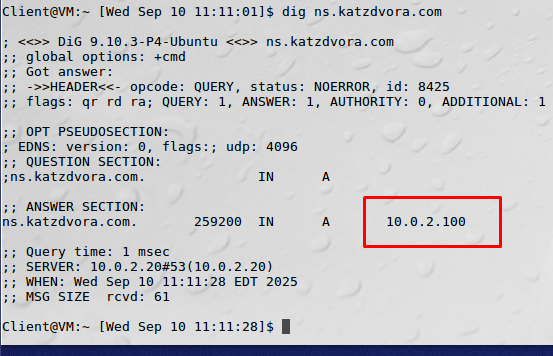
# 2.4.1מבוא:

בחלק זה של המטלה אנחנו רוצים לבדוק את שני הZones שהגדרנו בAttacker.

נרצה לשלוח פנייה מהClient לExample.com ולראות את התגובות שאנחנו מקבלים מServer. נעשה את זה פעם אחת דרך Server, שיבקש את זה דרך הDNS האמיתי, ופעם אחת דרך הAttacker, כמו שהיינו מקבלים אם הCache של הServer היה Poisoned.

בעצם אנחנו בודקים שכל ההגדרות עובדות ושאנחנו יכולים לעבור הלאה למתקפה.

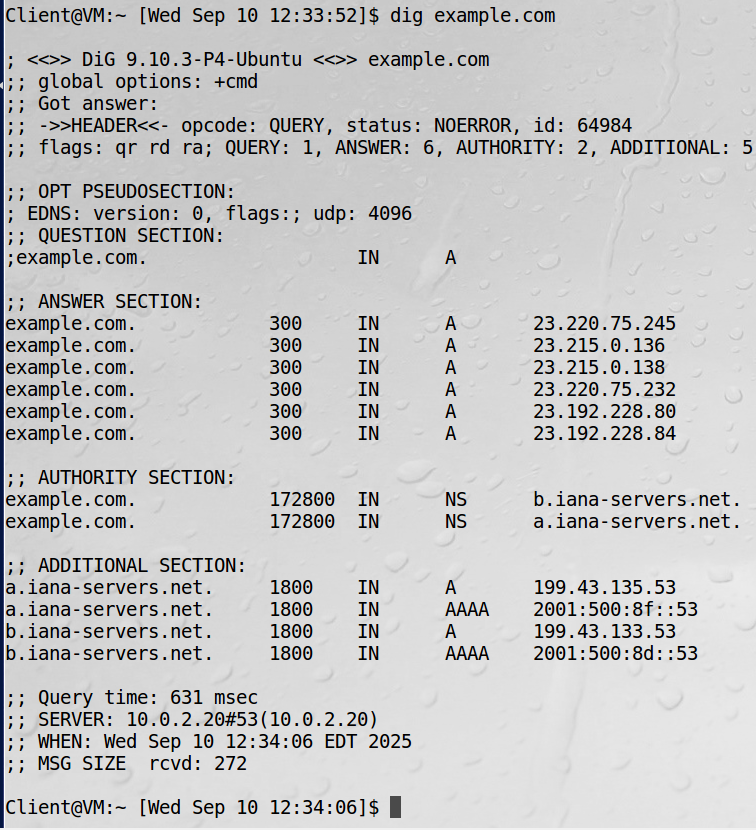
# 2.4.2 ביצוע המשימה:

נתחיל בלבצע Dig לkatzdvora.com כדי לראות שאכן עוברים דרך הAttacker (העתק הדבק ממקודם):

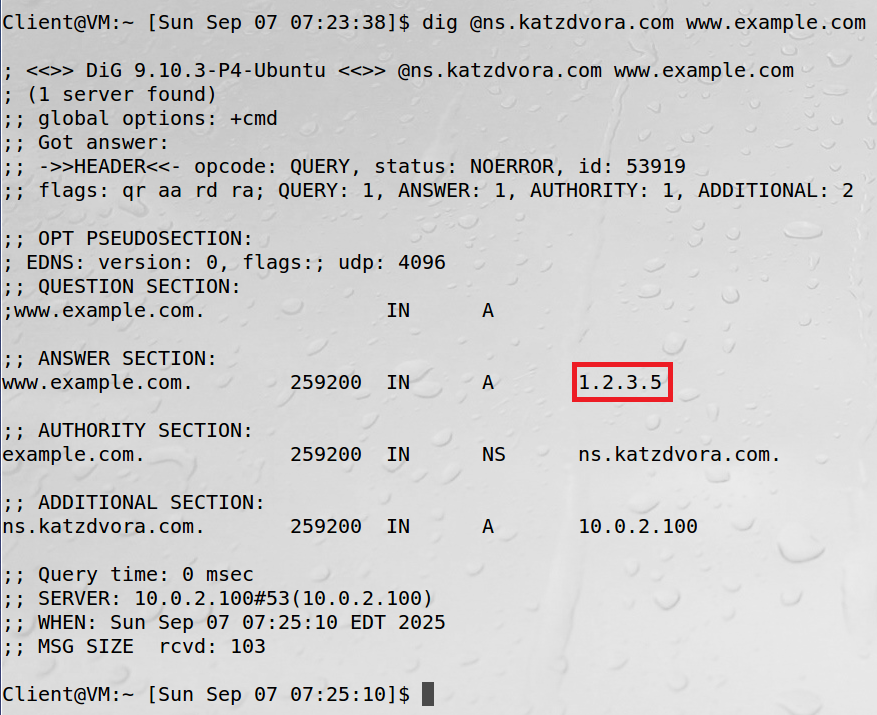
אפשר לראות שאנחנו פונים לServer ומקבלים את Attacker כתשובה, כמו שהגדרנו.

עכשיו נרצה לראות מה ההתנהגות כשאנחנו פונים לExample.com בשני אופנים: פעם אחת דרך Server שייצא החוצה, ופעם אחת דרך Attacker שיענה בעצמו.

נראה פנייה בדרך הרגילה:



אפשר לראות שקיבלנו תגובה מהServer באמצעות השרתים שמוגדרים לו ולא דרך הAttacker, כי עדיין לא עשינו Poison.

נראה בקשת dig אחרת, שהולכת דרך Attacker:

אפשר לראות שאנחנו מקבלים תגובה מהAttacker עם הנתונים שהגדרנו בשרת שלו, אז הSetup עובד כמו שצריך.

# 2.4.3 סיכום המשימה

ראינו שההגדרות שלנו עובדות כמו שצריך, אם אנחנו שואלים על katzdvora.com אנחנו מקבלים את התשובה שקיבלנו מהForward.

אם אנחנו שואלים על Example.com אנחנו מקבלים תגובה מהשרתים האמיתיים, אלא אם כן אנחנו מגדירים בכוונה לפנות דרך katzdvora ואז מקבלים את מה שהגדרנו שם.

# Task 3.4: Construct DNS request

# 3.4.1 מבוא

במטלה זו אנחנו רוצים לשלוח בקשות DNS לגיטימיות לשרת כדי להכין את המתקפה הגדולה יותר.

נשתמש בscapy כדי לשלוח Packets לLocal DNS Server, ונראה שהוא מחזיר תשובה לאחר שהוא פונה לשרתים אחרים.

# 3.4.2 ביצוע המשימה:

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.ניצור את הקוד:

'

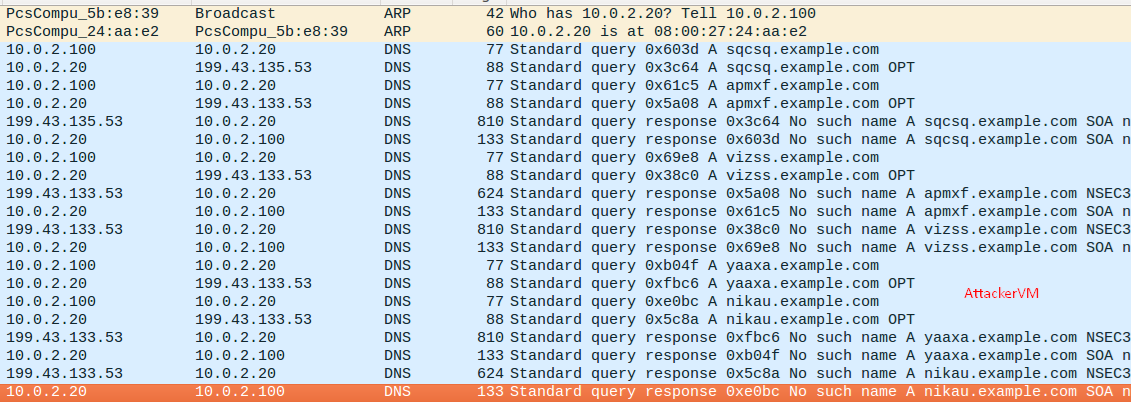
אנחנו מגדירים חמישה Subdomains רנדומליים עבור example.com, בונים Packet עבור DNS Query שיתאים, ושולחים.

Qdsec/qd: מה שאנחנו שולחים שאלה לגביו, במקרה הזה הsubdomain הרנדומלי.

אנחנו בוחרים פורטים רנדומליים לSource Port, וtransaction ID רנדומלי (למרות שהוא לא יעזור לנו בהתקפה עצמה).

תמונה שמכילה טקסט, צילום מסך, גופן, מידע

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.נריץ את הקוד בAttacker:

רואים שהתבצעה שליחה, עכשיו חסר לראות בwireshark שאכן Packets התקבלו:

אפשר לראות שהבקשות מתקבלות בServer ומקבלות תגובה בהתאם.

אנחנו מקבלים תגובה כזאת לכל אחת מחמשת הבקשות.

# 3.4.3 סיכום המשימה:

שלחנו בקשות DNS מזויפות לServer מAttacker וראינו שהן מתקבלות כמו שצריך ושהServer מחזיר עבורן תשובות (הבעיה שלנו היא שהתשובות לא מגיעות אלינו, בגלל Sport רנדומלי)

# Task 3.5: Spoof DNS Replies

# 3.5.1 מבוא:

בחלק זה של המטלה אנחנו רוצים לזייף DNS Reply Packets מהשרת האמיתי של Example.com אל הLocal DNS Server. המטרה שלנו היא למצוא את הIP של השרת האמיתי, ואז באמצעות שליחת Spoofed Packets לגרום לLocal DNS Server לקבל את המידע ולשמור אותו. בשביל זה נצטרך לקלוע לTransaction ID, ולא נצליח בחלק הזה של המטלה, אבל נראה שעובדת השליחה בWireshark.

# 3.5.2ביצוע המשימה:

תמונה שמכילה טקסט, צילום מסך, תוכנה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.ניצור את הקוד:

בקוד רואים זיוף של DNS Reply Packet, אבל בגלל שאנחנו עם tid רנדומלי ובגלל שאנחנו לא שולחים שאלה ההתקפה לא תצליח. אנחנו בסה"כ מכינים את ההתקפה לאח"כ.

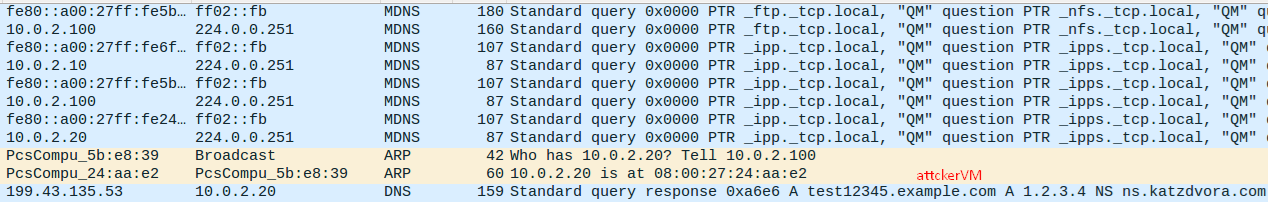
מבחינת השדות עצמם, מדובר באותו סגנון קוד כמו מקודם, אנחנו עונים תשובה לPacket וגם מגדירים את עצמנו כAuthority.

את הIP 199.7.91.13 לקחנו על ידי שליחת בקשה לשרת האמיתי של example.com.

תמונה שמכילה טקסט, צילום מסך, גופן, מידע

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.הרצה של הקוד:

רואים ששלחנו Packet.

נראה בWireshark שאכן נשלחה Packet ושהוא מזהה אותה כreply:

אפשר לראות שהPacket אכן נשלחת אבל בגלל שלא שלחנו request אנחנו מניחים שהLocal DNS Server זורק אותה.

# 3.5.3 סיכום המשימה:

בחלק זה של המטלה בסה"כ הגדרנו שליחה של Spoofed DNS Reply אחת, בלי משהו מעבר לזה.

זאת רק הכנה להתקפה האמיתית שנמצאת במטלה הבאה.

# Task 3.6 Launch the Kaminsky Attack.

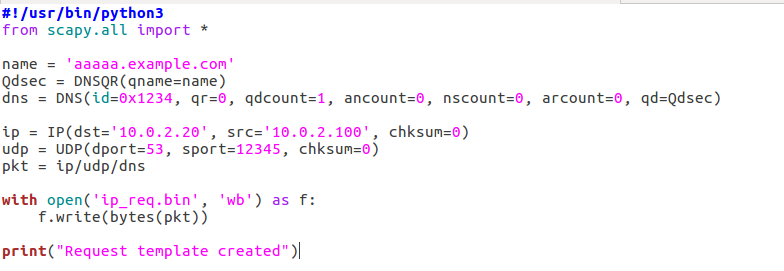
# 3.6.1 מבוא:

בחלק זה של המטלה אנחנו רוצים להצליח לבצע את Kaminsky Attack, כלומר לשלוח לLocal DNS Server בקשות עם subdomains שונים, ובזמן שהוא שואל את השרת האמיתי לשלוח לו תשובות מזויפות עד שאחת תהיה בדיוק עם הtransaction ID הנכון ותתקבל, וכך נהפוך להיות Authority עבור האתר.

ברגע שהצלחנו לעשות את זה, כל הClients שיבקשו מהLocal DNS Server יקבלו Spoofed Reply (יופנו אלינו) כי ביצענו Poison לCache של הLocal DNS Server.

נדע לעצור ברגע שנראה שיש פנייה אלינו לAttacker.

# 3.6.2 ביצוע המשימה:

נתחיל בליצור את הקוד בScapy עבור DNS Response:

אחרי שאנחנו יוצרים את הTemplate עבור הPacket אנחנו שומרים אותו בקובץ בינארי שנפתח אח"כ עם C.

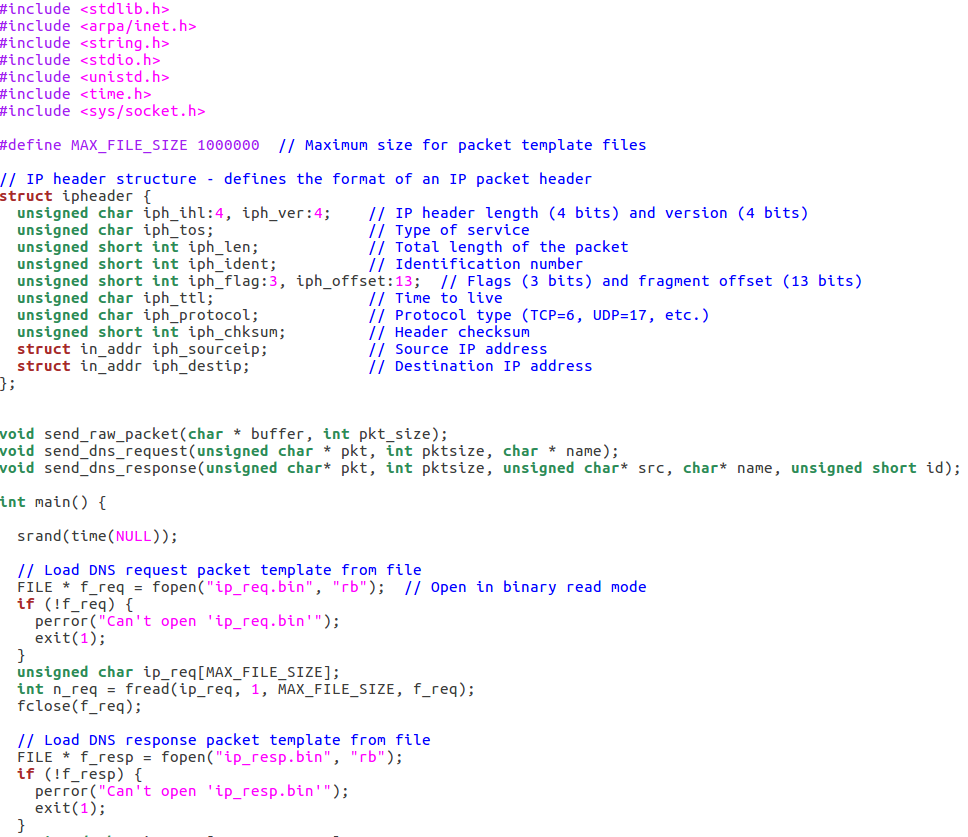
תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

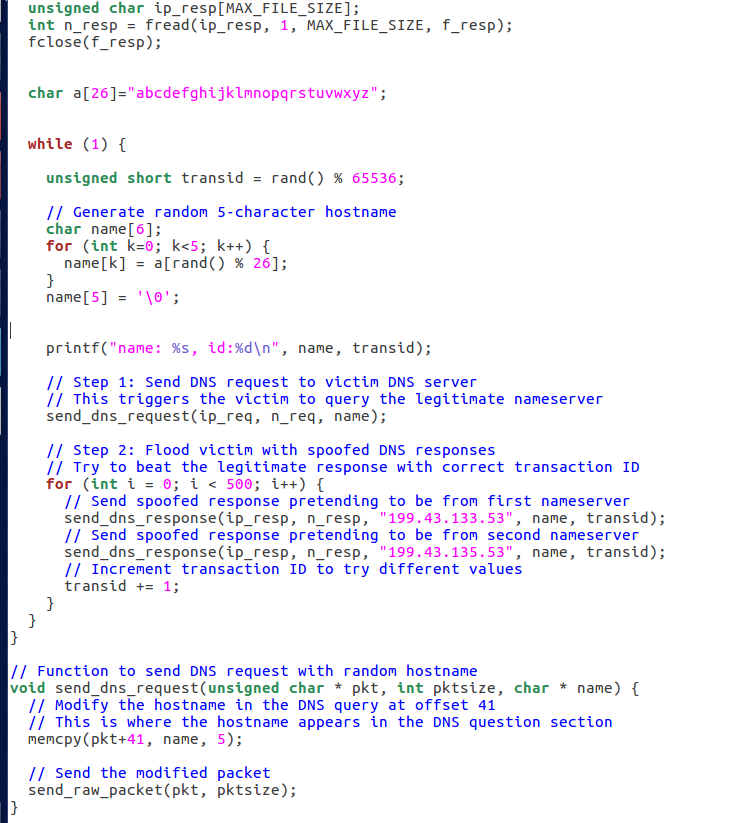
תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.ניצור קוד עבור DNS Reply Packet:

כאן אנחנו מגדירים את השדות שאנחנו צריכים בשביל לשלוח DNS Reply ושומרים אותו כקובץ בינארי גם.

אנחנו מגדירים בns את השרת שלנו כAuthoritative, וזאת המטרה שלנו בסוף -שהLocal DNS Server יקבל את הPacket לCache בצורה הזאת.

אחרי שיש לנו את הTemplates מScapy, אנחנו יכולים ליצור את הקוד בC (כי הוא מהיר יותר):

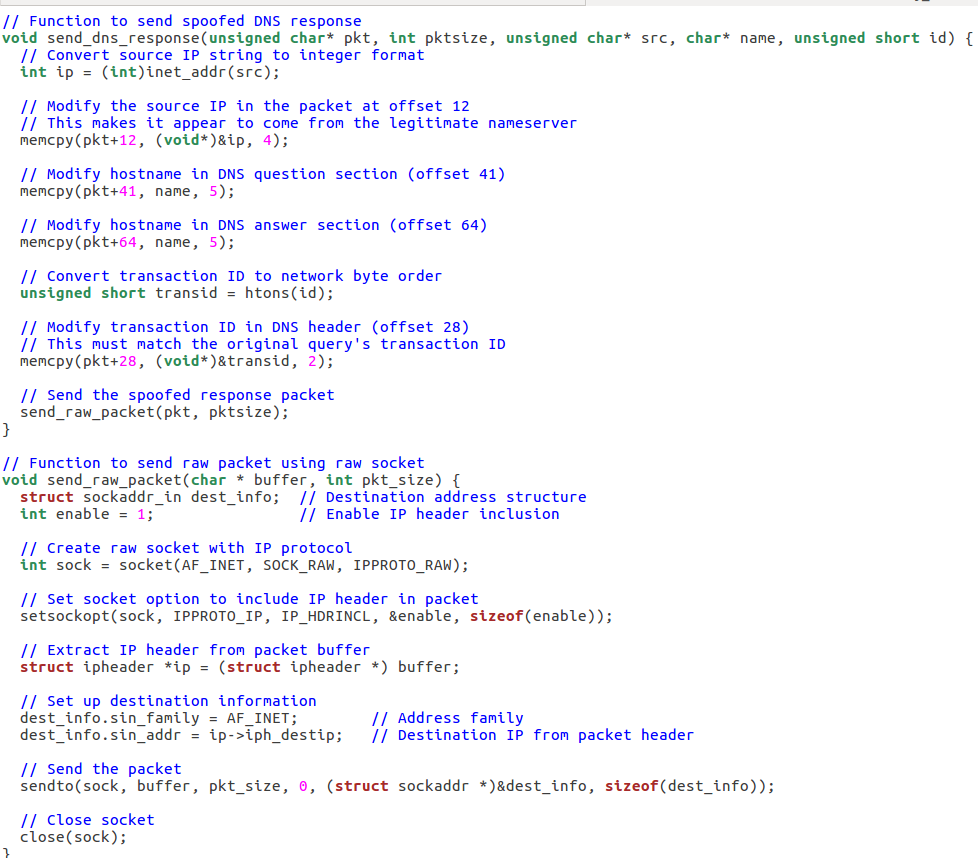


בחלק הזה של הקוד אנחנו פותחים את הקבצים של הrequest והresponse ושומרים אותם במשתנה, שאותו נשנה מאוחר יותר עם הפרטים הספציפיים שאנחנו רוצים.

עוד קוד:

החלק הראשון הוא ליצור בצורה רנדומלית שם באורך 5 אותיות שנשים כsubdomain כדי ליצור בקשות חדשות שלא נשמרות בCache של הLocal DNS Server.

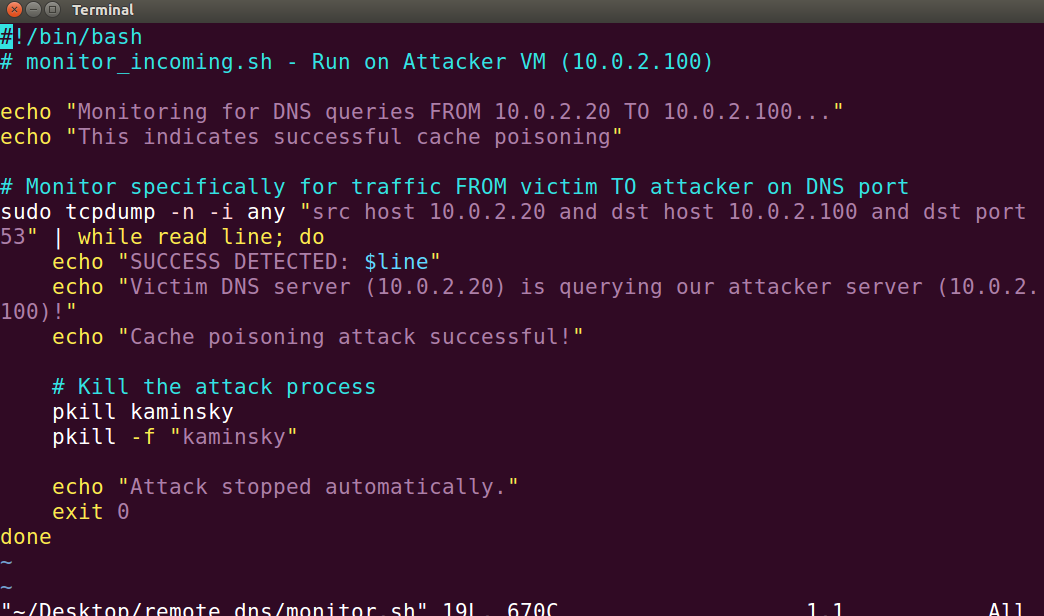
בStep 1 אנחנו שולחים את הRequest לLocal DNS Server כדי שישלח בקשה, ובזמן שהוא מחכה לתשובה מהServer ה"אמיתי" אנחנו בStep 2 שולחים 500 בקשות עם transId רנדומליים. (ערך התחלתי רנדומלי, ואז +500 בצעדים של 1)

עוד קוד:

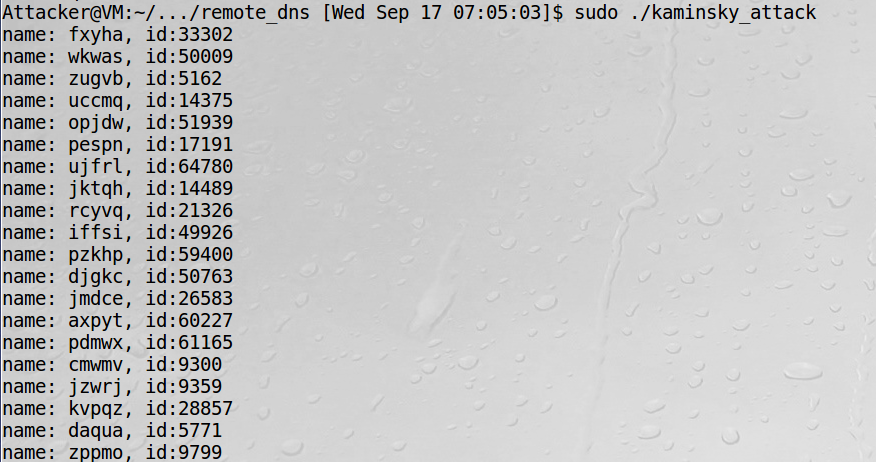
הפונקציה הראשונה היא כדי לשנות את הDNS Response לשדות שאנחנו רוצים, באמצעות memcpy וידיעת הOffset מההתחלה, אנחנו מגדירים את המשתנים כמו שאנחנו רוצים בIPHeader.

הפונקציה השנייה היא פשוט לשלוח את הPacket כRaw Packet עם Raw Socket, בגדול אנחנו אומרים שזה לשלוח את הPacket.

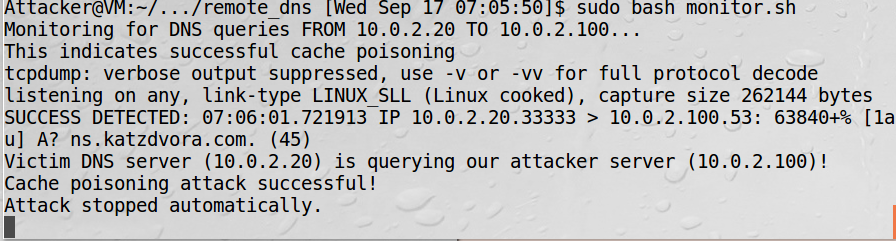
בנוסף לקוד, השתמשנו בBash Script שבודק אם אנחנו מקבלים הודעה מהServer לAttacker כדי שנדע מתי לעצור את המתקפה.



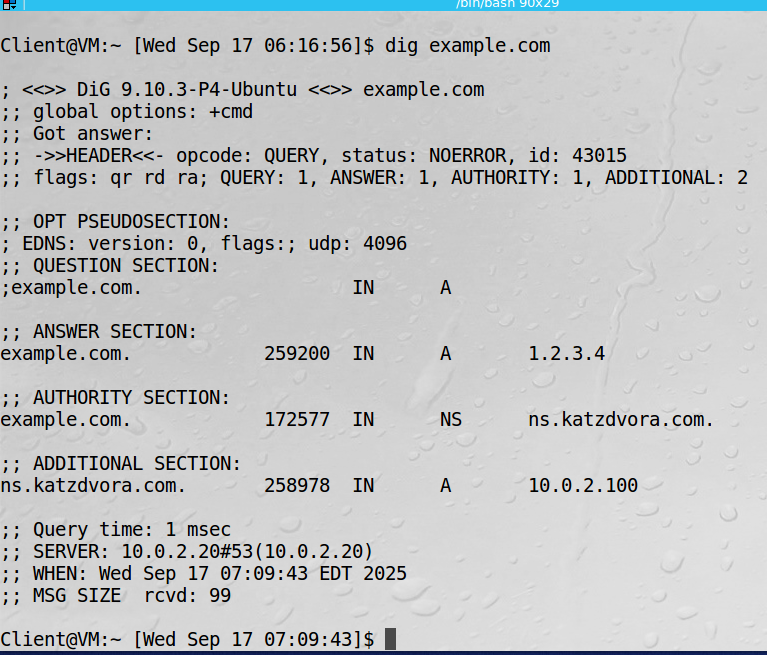
שימוש בScript הזה מאפשר לנו לעצור את ההתקפה בצורה אוטומטית ברגע שאנחנו מזהים Packet מהServer לAttacker.

הרצה של הקוד:

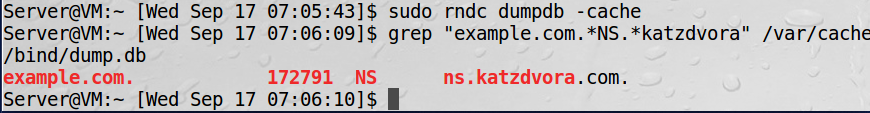
אפשר לראות שהשליחה מתבצעת עם שמות שונים ועם Transaction Ids שונים.

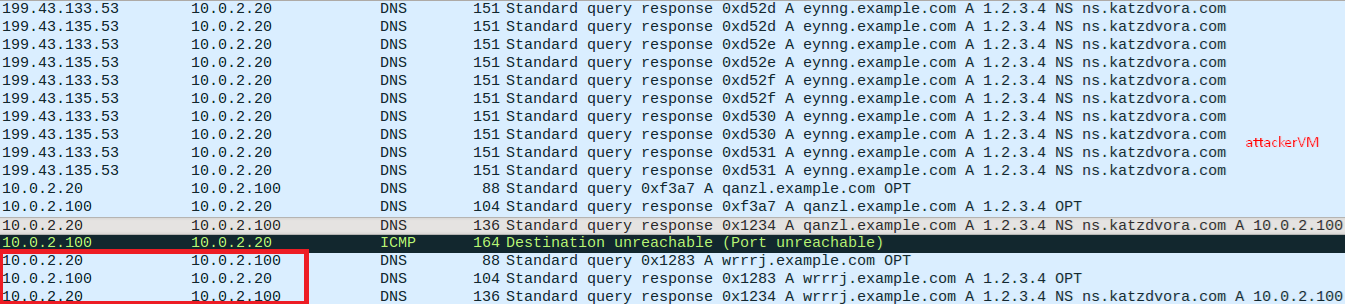
הרצה של הBash Script:

ברגע שההתקפה מצליחה אנחנו "הורגים" את התהליך של ההתקפה בצורה אוטומטית.

עכשיו כשאנחנו יודעים שההתקפה הצליחה, אנחנו יכולים לבדוק בClient:

רואים שאכן ההתקפה הצליחה והLocal DNS Server מפנה אלינו. (הראינו dig דרך attacker מוקדם יותר במעבדה – [2.4.2](#_2.4.2_ביצוע_המשימה:))

לא הוכחה, אבל אפשר לראות את זה בServer גם:

ולבסוף, wireshark:

אפשר לראות שהServer פנה לAttacker עם התשובה שExample.com עם Authority של ns.katzdvora.com, וזה מה שעוצר את הScript.

# 3.6.3 סיכום המשימה:

הצלחנו לבצע את ההתקפה, על ידי שילוב בין יצירת Packet בScapy ואז שליחה של הPackets האלה אחרי שינוי באמצעות C.

החלק הקשה בהתקפה הוא לנחש את הTransaction ID שיתאים בדיוק לServer Query, אבל בזכות קמינסקי אנחנו יכולים לשלוח לSubdomains לא קיימים כדי לא להיות תלויים בCache.

זה התאים לתיאוריה, אבל לקח זמן ליישם.

# Task 3.7: Result Verification

עשינו בתוך 3.6.

# סיכום המעבדה

רוב תחילת המעבדה היה קשור בSetup יותר מאשר משהו אחר:

בהתחלה הגדרנו את הLocal DNS Server בClient כך שיהיה הראשי בו הClient משתמש.

לאחר מכן הגדרנו את הLocal DNS Server עצמו – הגדרות כמו איפה יישמר הCache Dump, ביטול DNSSEC, ובדיקה שהForward Zone שהגדרנו עובד.

אח"כ הגדרנו את הDNS Server בAttacker עם הZones שהוא צריך בשביל ההתקפה.

ביצענו בדיקות אם ההגדרות עובדות כמו שצריך, וראינו שהן אכן עובדות – אם פונים לExample.com מקבלים תשובה אמיתית, ואם שואלים דרך השרת שהגדרנו מקבלים את מה שהגדרנו בZones.

בחלק השני של המטלה הגדרנו באמצעות Scapy בשביל ההתקפה האמיתיתDNS Request מזויפות, ולאחר מכן גם DNS Responses מזויפות.

לבסוף, ביצענו את ההתקפה על ידי שילוב בין Scapy (בניית Packets שנקרא אותם בC) לC ואז שליחה של הPackets האלה כדי לתפוס את הTransaction ID הנכון.

התהליך הוא לשלוח בקשה מזויפת ובזמן שהיא נשלחת לשלוח הרבה תשובות מזויפות עם Transaction IDs רנדומליים עד שמצליחים. עצרנו את ההתקפה ברגע שראינו שהLocal DNS Server פונה אלינו.

הכל התאים לתיאוריה, הבעיות שנתקלנו בהם היו איך לעבוד עם הקוד כמו שצריך וגם איך לעצור את ההתקפה בצורה אוטומטית בלי לנצל את העובדה שאנחנו בLAN עם הServer, אבל בסוף הסתדרנו.

# לימוד מעבר לדרישות המעבדה

[DNSSEC](https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions):

מדובר במנגנון הגנה שאנחנו ביטלנו ידנית במעבדה הזאת, כדי לבצע את ההתקפות.

הוא בעצם משתמש בחתימות דיגיטליות כדי לוודא שהמידע שהתקבל מהשרת הוא אכן המידע שהשרת שלח, וכך מונע spoof כמו שעשינו במטלה הזאת.

מדובר במנגנון של חתימות דיגיטליות עם chain of trust, כך שכל שרת DNS ששולח תשובה גם חותם על Hash של התשובה עם המפתח הפרטי שלו, והצד המקבל יכול לבדוק שהתשובה לא שונתה עם המפתח הציבורי.

בעצם הCertificates מתקבלים מהDNS Root Zone שנחשב לTrusted Party, ואתרים חדשים יוצרים את המפתחות בעצמם ומעלים אותם דרך הDomain Name Registrar שלהם, בעצם מי שאחראי על רישום האתר והDNS במערכת הכללית. הוא בעצמו קיבל Certificate מלמעלה, ואז הregistrar דוחף את המפתחות באמצעות secDNS לZone operator שלו וכך מפתחות חדשים נוצרים.

אם היינו מנסים לתקוף עם המנגנון הזה דלוק, לא היינו מצליחים כי הDNS Server היה בודק את החתימה של המידע ומשווה עם המקור הרשמי, רואה שאין התאמה וזורק את התשובה.