DNSSEC (Spoof)

הקדמה – DNSSEC

נכון להיום, אודות ל- DNSSEC, לא ניתן לבצע DNS cache poisoning.

לפני שנדבר מה הוא בדיוק עושה, נזכיר בקצרה מה היה פעם:

התוקף היה יכול לשלוח שליחת שאילתת DNS סתמית לשרת NS, וע"י ניחוש ה-Trans. ID של התשובה, שהיה בד"כ אחד מהמספרים העוקבים של ה-Trans. ID של השאילתה, היה ניתן לעבוד על ה-NS ולהגיד כי URL *X* נמצא בכתובת *Y*.

ובחזרה ל-DNSSEC:

DNSSEC מתבסס על חתימה דיגיטלית. מה זה חתימה דיגיטלית?

- חתימה דיגיטלית היא דרך להעביר את המידע ע"י שימוש בהצפנה א-סימטרית בין שני גורמים.
  <u>אופן הפעולה:</u> הצד השולח לוקח את ה-DATA ומעביר אותו דרך פונק' HASH מסוימת ולאן מכן מצפין (=חותם) את הפלט של ה-HASH עם המפתח הפרטי שלו. לאחר כל התהליך הזה, המידע + החתימה נשלח לצד השני.
- כאשר הצד השני מקבל את החלק הזה, הוא מעביר בעצמו את ה-DATA באותה פונק' HASH, ובמקביל משתמש במפתח הציבורי של השולח ומפענח את החתימה. אם ה-HASHים שהתקבלו שווים – זאת אומרת שהמידע אמיתי ולא עבר שום שינוי.

הבעיה המתעוררת היא שניתן בעצם לעבוד על שרת DNS מקומי ולהגיד לו שאני שרת ה-ROOT (לדוגמא) ולשלוח לו מה המפתח הציבורי שלי – עכשיו נראה איך מתגברים על הבעיה הזאת.

כאשר משתמשים ב DNSSEC לכל תשובה *רגילה* מתווספת הודעת RRSIG, אשר מכילה את החתימה הדיגיטלית על התשובה *הרגילה* שאותו שרת NS שלח.

זה לא מספיק – כי צריך לדעת מה המפתח הציבורי המתאים של השולח (נכון לעכשיו תוקף עדיין יכול להתחזות לשרת NS ולהגיד הנה התשובה + החתימה + המפתח הציבורי שיפענח לך את זה.)

לכן יש עוד הודעה, שגם היא נשלחת למשתמש, שנקראת DNSKEY – הודעה זו מכילה את המפתח הציבורי המתאים של השרת השולח, והינה דוגמא לאיך שהיא נראית:

```
(1)                 (2)     (3)  (4)    (5)  (6) (7)
dskey.example.com. 86400 IN DNSKEY 256 3 5 ( AQOeiiR0GOMYkDshWoSKz9Xz
                                             fwJr1AYtsmx3TGkJaNXVbfi/
                                              2pHm822aJ5iI9BMzNXxeYCmZ
                                              DRD99WYwYqUSdjMmmAphXdvx
                                    (8)       egXd/M5+X7OrzKBaMbCVdFLU
                                              Uh6DhweJBjEVv5f2wwjM9Xzc
                                              nOf+EPbtG9DMBmADjFDc2w/r
                                              ljwvFw==
                                              ) ;  key id = 60485
                                                        (9)
```

הסבר מה יש כאן:

(1) מי שלח את ההודעה הזאת
(2) TTL
(3) Class
(4) סוג רשומת RR
(5) נעזוב את זה כרגע.
(6) אינטרנט פרוטוקול - היום זה 3, יכול להיות בעתיד זה ישתנה..
(7) איזה פונק' HASH השתמשנו ומהו אלגוריתם ההצפנה הא-סימטרי[1]
(8) BASE64 של המפתח הציבורי
(9) תגית זיהוי Key Tag (נבין עוד רגע מה זה)

[1] רשימת כל האפשרויות ניתן למצוא ב- RFC 4034

זה לא מספיק לנו מבחינת אבטחה – כי בינתיים שום דבר לא מונע מהתוקף לשלוח את שלושת ההודעות הללו ולהתחזות לשרת DNS. וכאן נכנסת לתמונה רשומת DS RR.
DS RR מכיל HASH של הודעת ה-DNSKEY וזה מאפשר לדעת שה-DNSKEY אינו מזויף!

נעשה HASH על השדות הבאים:

```
digest = digest_algorithm( DNSKEY owner name | DNSKEY RDATA);

 "|" denotes concatenation

DNSKEY RDATA = Flags | Protocol | Algorithm | Public Key.
```

וככה נראית הודעה זו:

```
(1)                (2)     (3) (4)    (5)   (6) (7)          (8)
dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A
                                           98631FAD1A292118 )
```

הסבר מה יש כאן:
(1) מי שלח את ההודעה הזאת
(2) TTL
(3) Class
(4) סוג רשומת RR
(5) נועד לזהות על איזו הודעת DNSKEY עשו את ה-HASH בהודעה הזו (נשים לב שזה המשך לדוגמא הקודמת)
(6) איזה אלגוריתם הצפנה ופונק' HASH השתמשו בהודעת DNSKEY המתאימה.
(7) פונק' HASH שהשתמשו בה בהודעת DS הזאת.
(8) ה-HASH עצמו.

אבל עדיין! התוקף יכול לשלוח את 4 ההודעות הללו – עדיין לא התגברנו על הבעיה!
אבל זה לא נכון – ופה הקאץ'.
את ההודעה הזאת, **לא** שולח ה-NS, אלה ה***אבא*** שלו (מי שלפניו בהיררכית ה-DNS).
כלומר את ה-DS של שרת ה-authorize נקבל משרת ה-SLD שהפנה אותנו אליו.
את ה-DS של אותו שרת SLD נקבל מהאבא שלו וכן הלאה וכן הלאה עד שרתי ה-ROOT.
כאשר כל שרת בן נוצר הוא מודיע לאבא שהוא נוצר + מעביר לו את המפתח הציבור שהוא משתמש בו!
וכאשר מתקינים מערכת ההפעלה במחשב מסוים, מקבלים, בין היתר, את ה-DS של שרתי ה-ROOT!

כמו שאמרנו, את הדבר הזה לא ניתן לפרוץ היום(לצערנו 😉), ולכן נממש את התקיפה שהייתה אפשרית פעם.

אופן הפעולה[2]

הלקוח בוחר לאיזה אתרים הוא מונע גישה מהם – מה תהיה כתובת ה-IP המזויפת שלהם, מה כתובת ה-IP של שרת ה-NS המותקף.[3]
כמו כן, כדי לאפשר MITM, יש צורך לדעת גם מי ה-gateway ולכן גם אותו הלקוח מספק לנו[4].

לאחר שפירסרנו את המשתנים שהלקוח הכניס, אנו מריצים את הקוד התוקף.

הפונק' run [5]של המחלקה DNSSpoof מממשת את המתקפה. פונק' זו כוללת:

1. פתיחת UDP socket בפורט 53. נעשה בגלל שהמכונה אינה שרת DNS ופורט זה סגור אצלה באופן אוטומטי, ולכן כדי למנוע מצב שהמכונה שולחת באופן אוטומטי הודעת ICMP port-unreachable למותקף – פתחנו את הפורט הזה.
2. הרצה, בתהליך נפרד, את המתקפה ARP Spoof, כדי שיעביר אלינו, בין היתר, את שאילתות ה-DNS שהוא שולח ל-gateway.
3. הסנפת התעבורה – זיהוי שאילתות DNS בין הקורבן ל-gateway ומתן מענה עליהן.
4. כשאר הלקוח בוחר לסיים את המתקפה – סגירת הסוקט הפתוח וסיום התהליך אשר מבצע ARP Spoof.

ועכשיו בפירוט – איך עשינו את זה בפועל:

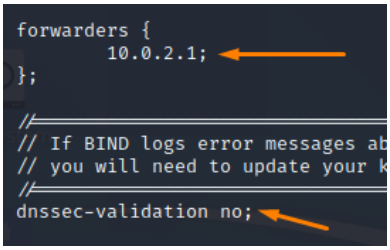
בפונק' dns_req[6] זיהינו שאילתות DNS אשר נשלחות משרת ה-DNS הקורבן אל ה-gateway.

עבור אותן הודעות, בפונק' process_spoof[7] חילקנו אותן לשני סוגים:

- עבור שאילתות DNS אשר שייכות ל-domain שאותו הלקוח רוצה לחסום - בנינו תשובה שיקרית ושלחנו אותה לקורבן.
- אבור שאילתות DNS אחרות – שלחנו אותן לשרת DNS ציבורי של גוגל, ואת התשובה החזרנו לקורבן.

צילומי מסך

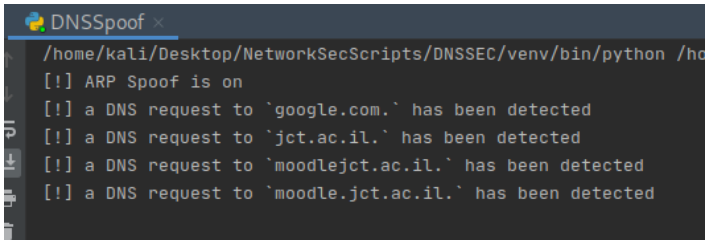
ראשית, יש לציין שקינפגנו את השרת כך שיעביר את כל ההודעות ה-DNS שלו אל ה-gateway, ושלא ישתמש ב-DNSSEC[8]:



ועכשיו – הוכחה שהתקיפה עובדת

שלחנו את הבקשות באמצעות הפקודה: $dig\ @127.0.0.1 < domain\ name >$

ככה זה נראה מהצד של התוקף:

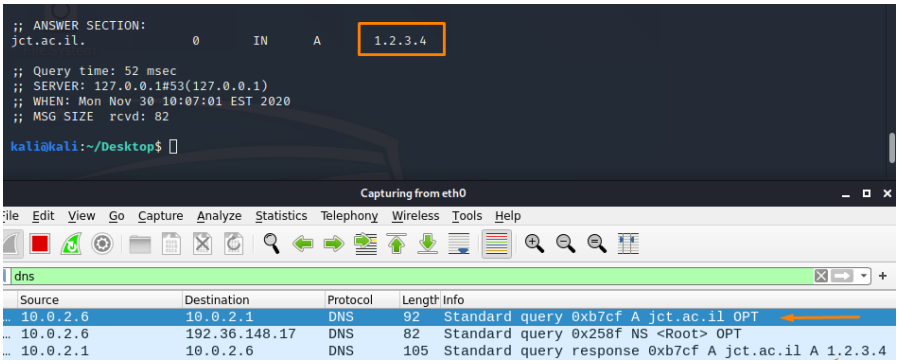


---

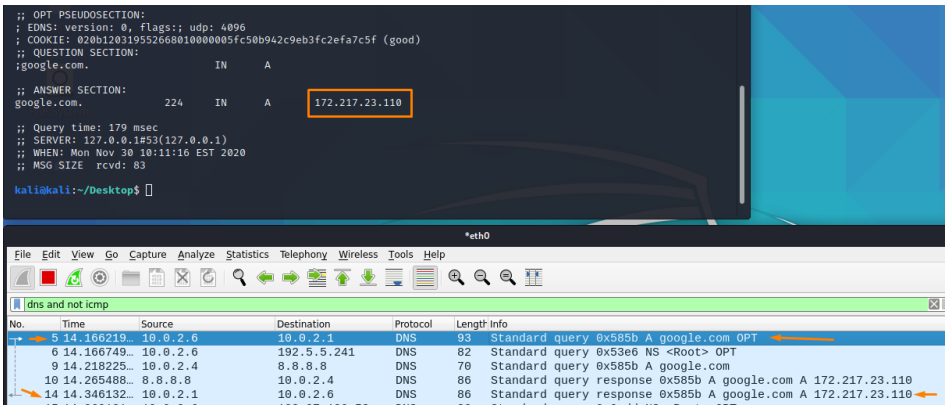
[2] השתמשנו בשרת DNS מסוג bind9 ב-Kali Linux. קישור לסרטון שמסביר איך לקנפג שרת זה ניתן למצוא כאן.
[3] אלו הן הפרמטרים אשר הסקריפט מקבל, אשר פרסורם נעשה בפונק' get_args בשורות 12-26.
[4] את ה-MITM אפשר לעשות עם ARP Spoof, ועם כלי זה גם עשינו. הסבר עליו ניתן למצוא בתרגיל מס' 2.
[5] שורות 95-110
[6] שורות 59-67
[7] שורות 69-93
[8] הקינפוג נעשה בקובץ /etc/bind/named.conf.options

ובצד המותקף,

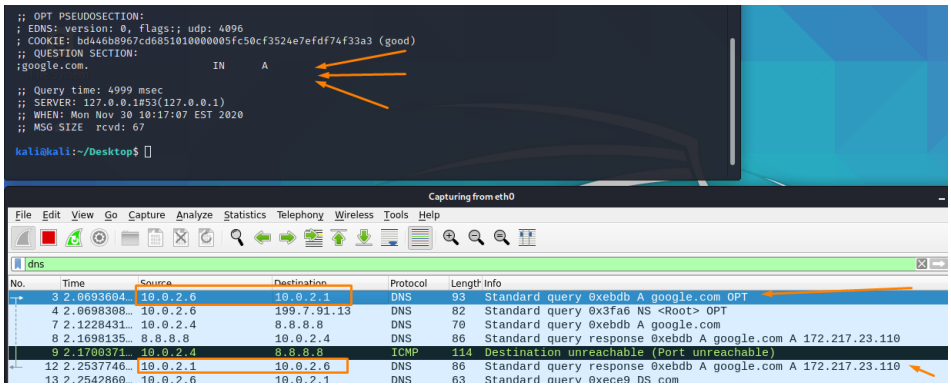
- עבור כתובת שהלקוח בחר לחסום[9]:

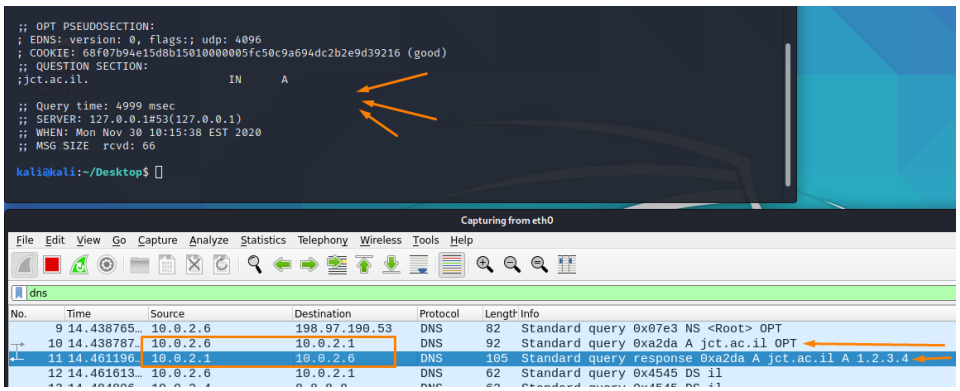


- עבור שאר האתרים, לדוגמא, גוגל:[10]



שלושת התמונות הבאות מהוות הוכחה שהתקיפה לא עובדת כאשר DNSSEC עובד:





---

[9] בדוגמא זו בחרנו לחסום את כל הכתובות שמכילות JCT – הפנינו אותן אל הכתובת 1.2.3.4

[10] מיותר לציין שניקינו לפני זה את ה-cache ע"י שימוש בפקודה *sudo rndc flush*