בס"ד

Arp Spoof Warning

בתחילת הmain יש ArgumentParser שמאפשר לקבל מידע הסקריפט ולהעביר לו interface בקלות:

```
parser = argparse.ArgumentParser(description="Arpspoof Warning System. By Daniel Yochanan and Avi Feder")
parser.add_argument("-i", "--IFACE", type=str, metavar='', help="Interface you wish to use")
args = parser.parse_args()
interface = args.IFACE if (args.IFACE is not None) else conf.iface
```

לאחר מכן, יש פונקציה שמסניפה חבילות, ובמידה וחבילה מכילה ARP REPLAY, היא בודקת, באמצעות שני אינדיקטורים, האם יש התקפה. ובמידה ויש התקפה היא מפעילה פונקציה נוספת ע"מ להתמודד עם ההתקפה:

```
def check_spoofing(packet):
    if packet.haslayer(ARP) and packet[ARP].op == 2:
        if indicator1(packet) and indicator2(packet):
            print("You are under arp spoofing attack!")
            stop_attack(packet)
```

האינדיקטור הראשון, בודק האם הכתובת מק שמוצמדת לIP בחבילה שהוסנפה תואמת לכתובת מק שנקבל אם נעשה ARP REQUEST, לכתובת הIP הזאת:

```
def indicator1(packet):
    real_mac = get_mac(packet[ARP].psrc)
    response_mac = packet[ARP].hwsrc
    if not real_mac or not response_mac:
        return False
    if real_mac != response_mac:
        return True
    return False
```

האינדיקטור השני, בודק האם הכתובת IP הזאת הוצמדה בעבר לכתובת מק אחרת. ובמידה ולא שומר את הכתובת מק בסמיכות לIP בתוך מילון בשביל הבדיקות הבאות:

```
def indicator2(packet):
    if packet[ARP].psrc in ip_mac_dict:
        return False if (ip_mac_dict[packet[ARP].psrc] == packet[ARP].hwsrc) else True
    else:
        ip_mac_dict[packet[ARP].psrc] = packet[ARP].hwsrc
        return False
```

לאחר מכן, אם אכן זוהתה התקפה, מופעלת פונקציה להתמודדות עם ההתקפה.

הפונקציה שולחת לכתובת IP שמנסים לזייף לה את המק אלפי ARP REQUEST, וככה נצליח לקבל את הכתובת מק הנכונה:

```
def stop_attack(packet):
    while True:
        Ether(dst='ff:ff:ff:ff:ff:ff') / ARP(op=1, pdst=packet[ARP].psrc)
        sleep(0.01)
```