Desc:

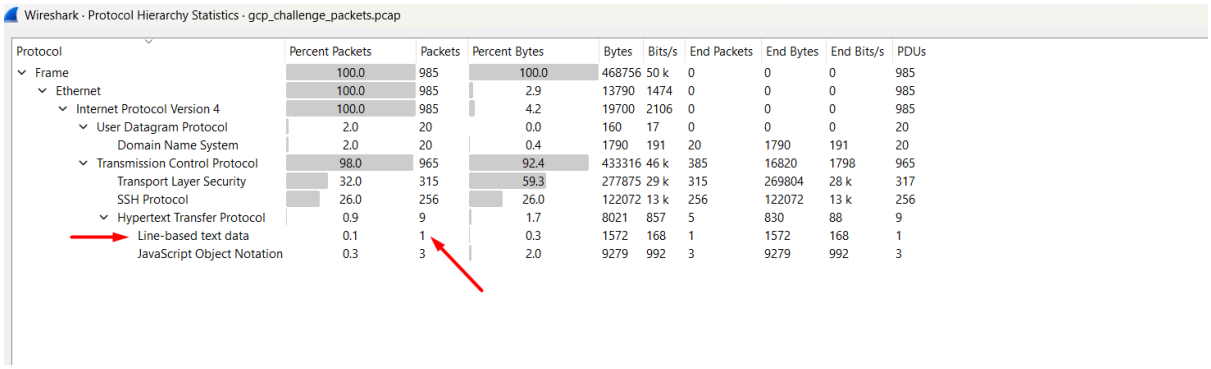
Lately, my dear friend Thomas is interested in DNS. He is trying to make a custom DNS server using his computer. He reads lot of articles about what is DNS to help him understand. After hours of work, he finally succeeds to make his own custom DNS server, The server that he made only have 1 domain stored in its database, but there is still some issue about the server. If we try to find out the IP address from the domain that is stored in the database of the custom server, there will be an error. Thomas asks me for help to retrieve information about the stored domain, but I guess you guys are more capable to do it. Here's the traffic that's captured when Thomas is trying to learn about DNS, maybe it can help you solve it. Oh yeah, 1 more thing, Thomas mentioned something about port 3534, hope it helps

format flag: COMPFEST14{IPAddress_ID}

Info yang diperoleh :

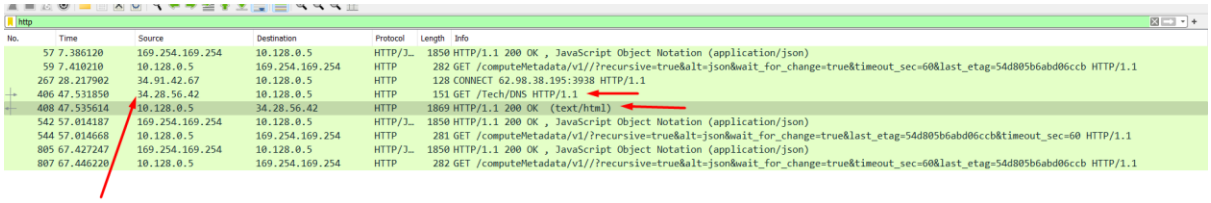
- Thomas membuat custom dns server di computer pribadinya
- Thomas berhasil membuat sebuah dns server dan menyimpan 1 buah domain di server tsb, namun terjadi error bila kita bertanya kepada DNS server terkait domain yg tersimpan di database dns server
- Thomas menyebutkan sesuatu terkait port 3534
- Kita memperoleh berkas pcapng yang difetch Ketika Thomas sedang membaca artikel terkait DNS
- format flag: COMPFEST14{IPAddress_ID}

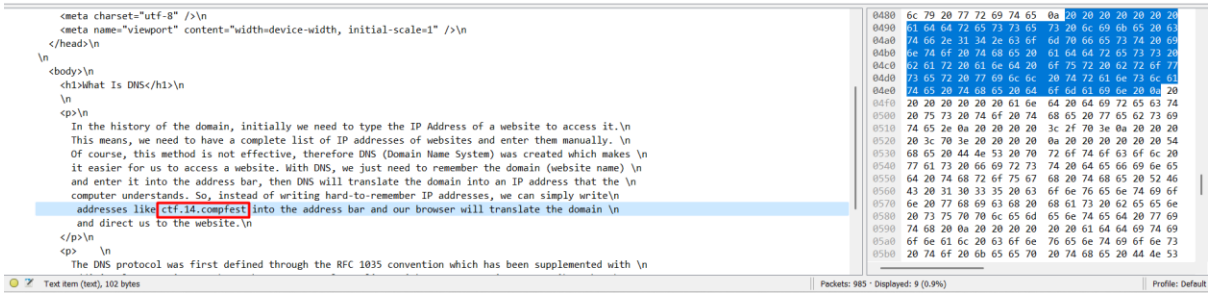
Mari kita buka berkas tsb dengan wireshark:



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	985	100.0	468756	50 k	0	0	0	985
Ethernet	100.0	985	2.9	13790	1474	0	0	0	985
Internet Protocol Version 4	100.0	985	4.2	19700	2106	0	0	0	985
User Datagram Protocol	2.0	20	0.0	160	17	0	0	0	20
Domain Name System	2.0	20	0.4	1790	191	20	1790	191	20
Transmission Control Protocol	98.0	965	92.4	433316	46 k	385	16820	1798	965
Transport Layer Security	32.0	315	59.3	277875	29 k	315	269804	28 k	317
SSH Protocol	26.0	256	26.0	122072	13 k	256	122072	13 k	256
Hypertext Transfer Protocol	0.9	9	1.7	8021	857	5	830	88	9
Line-based text data	0.1	1	0.3	1572	168	1	1572	168	1
JavaScript Object Notation	0.3	3	2.0	9279	992	3	9279	992	3

Berdasarkan deskripsi, diketahui bahwa Thomas memperoleh informasi untuk membuat DNS server berdasarkan artikel yang dibacanya melalui internet. Berdasarkan hirarki protocol yang ada di wireshark, diketahui bahwa ada 1 packet text data yang di capture menggunakan protocol http. Karena hanya 9 paket yang menggunakan protocol http, mari kita jadikan http sbg filter dan buka packetnya.





Setelah dibuka, ternyata kita memperoleh artikel yang dibaca oleh Thomas untuk membuat DNS server, pada artikel tersebut disampaikan bahwa salah satu contoh domain adalah ctf.14.compfest, hmm mari kita simpan informasi ini. Kemudian kita juga memperoleh alamat ip dari Thomas, yaitu 34.28.56.42

Informasi yang diperoleh sejauh ini :

- Thomas membuat custom dns server di computer pribadinya
- Thomas berhasil membuat sebuah dns server dan menyimpan 1 buah domain di server tsb, namun terjadi error bila kita bertanya kepada DNS server terkait domain yg tersimpan di database dns server
- Thomas menyebutkan sesuatu terkait port 3534
- Kita memperoleh berkas pcapng yang difetch Ketika Thomas sedang membaca artikel terkait DNS
- format flag: COMPFEST14{IPAddress_ID}
- Ip Thomas: 34.28.56.42
- Contoh domain: ctf.14.compfest

Baik, sebagaimana yang diketahui Thomas membuat custom DNS server di computer pribadinya. Sejauh ini kita telah memperoleh alamat ip (34.28.56.42) dan port (3534), mari kita coba lakukan “dig” ke ip tsb

```
┌──(ignite㉿IGNITE)-[~]
└─$ dig -p 3534 @34.28.56.42 www.google.com
```

```
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10734
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         273     IN      A       209.85.147.103
www.google.com.         273     IN      A       209.85.147.147
www.google.com.         273     IN      A       209.85.147.99
www.google.com.         273     IN      A       209.85.147.105
www.google.com.         273     IN      A       209.85.147.106
www.google.com.         273     IN      A       209.85.147.104

;; Query time: 490 msec
;; SERVER: 34.72.103.134#3534(34.72.103.134) (UDP)
;; WHEN: Wed Nov 02 16:59:42 WIB 2022
;; MSG SIZE  rcvd: 139
```

Oke kita berhasil memperoleh respon saat melakukan dig domain www.google.com. Oh yaa, berdasarkan artikel yang telah kita baca, artikel tsb menyebutkan ctf.14.compfest, mari kita gunakan domain tsb sebagai parameter

```
┌──(ignite㉿IGNITE)-[~]
└─$ dig -p 3534 @34.28.56.42 ctf.14.compfest
;; Warning: ID mismatch: expected ID 53901, got 4645
;; communications error to 34.72.103.134#3534: timed out
;; Warning: ID mismatch: expected ID 53901, got 17189
;; communications error to 34.72.103.134#3534: timed out
;; Warning: ID mismatch: expected ID 53901, got 17189
;; communications error to 34.72.103.134#3534: timed out
```

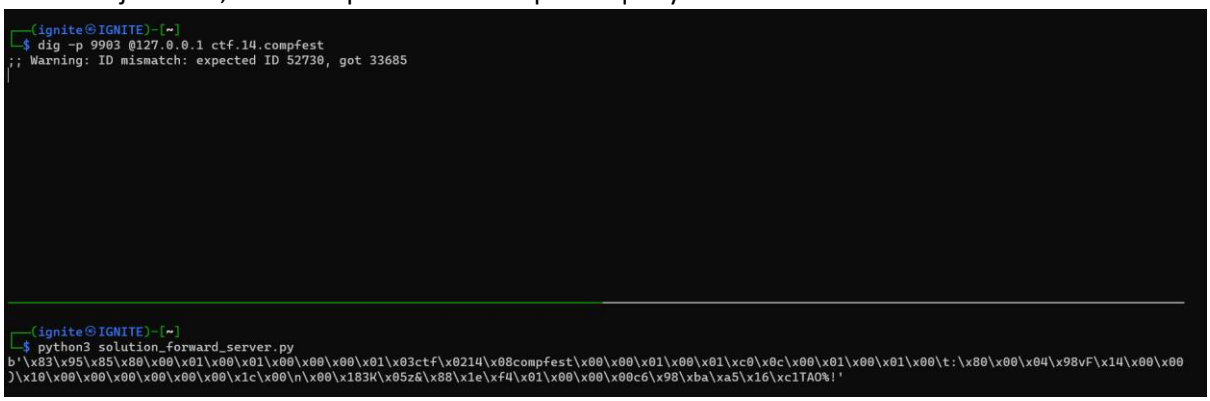
Kita memperoleh error ID mismatch. Sebagaimana yang diketahui, Thomas berhasil membuat sebuah dns server dan menyimpan 1 buah domain di server tsb, namun terjadi error bila kita bertanya kepada DNS server terkait domain yg tersimpan di database dns server, berarti domain yang tersimpan di database dns server adalah ctf.14.compfest, bagaimana cara kita untuk memperoleh IP Adress dan ID dari ctf.14.compfest ?

Kita bisa membuat sebuah forwarding server yang berguna untuk meng-intercept / mencegat response dari server, untuk memperoleh raw response

Berikut kode dari forwarding server:

```python
import socket

port = 9903
ip = ''
ip_server = '34.28.56.42' #ip server
port_server = 3534 # port server
BUFFER_SIZE = 2048
server_addr = (ip_server,port_server)

sc = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

sc.bind((ip, port))

while True:

    data_client, client_addr = sc.recvfrom(BUFFER_SIZE)
    sc.sendto(data_client, server_addr)
    data_server, ip_asdos = sc.recvfrom(BUFFER_SIZE)

    print(data_server)

    sc.sendto(data_server,client_addr)
```

setelah dijalankan, kita memperoleh raw response query :



response:

b'\x83\x95\x85\x80\x00\x01\x00\x01\x00\x00\x00\x01\x03ctf\x0214\x08compfest\x00\x00\x01\x00\x01\xc0\x0c\x00\x01\x00\x01\x00\t:\x80\x00\x04\x98vF\x14\x00\x00)\x10\x00\x00\x00\x00\x00\x00\x1c\x00\n\x00\x183K\x05z&\x88\x1e\xf4\x01\x00\x00\x00c6\x98\xba\xa5\x16\xc1TAO%!'

mari kita terjemahkan:

Berikut kode python untuk menerjemahkan raw DNS response query :

```
import struct

def get_transaction_id(data):
    data_msg= data[:2]
    id =""
    for byte in data_msg:
        id += hex(byte)[2:]

    id = int(id,16)
    return id


def get_flag(data):
    flag={}
    data_msg=data[2:4]

    data=struct.unpack('!H',data_msg)[0]

    qr = (data>>15)&int("1",2)
    opcode = (data>>11)&int("1111",2)
    AA = (data>>10)&int("1",2)
    TC = (data>>9)&int("1",2)
    RD = (data>>8)&int("1",2)
    RA = (data>>7)&int("1",2)
    Z = (data>>6)&int("1",2)
    AD = (data>>5)&int("1",2)
    CD = (data>>4)&int("1",2)
    RCODE = (data)&int("1111",2)

    flag['qr'] = qr
    flag['opcode'] = opcode
    flag['aa'] = AA
    flag['tc'] = TC
    flag['rd'] = RD
    flag['ra'] = RA
    flag['z'] = Z
    flag['ad'] = AD
    flag['cd'] = CD
    flag['rcode'] = RCODE

    return flag


def get_count(data):

    arr=[]
```

```
    for i in range(4,15,2):
        data_count = data[i:i+2]
        count = struct.unpack('!H',data_count)[0]
        arr.append(count)
    return arr

def get_domain(data):
    arr =[]
    data = data[12:]
    state = 0
    expectedlength = 0
    domainstring = ''
    domainparts = []
    x = 0
    y = 0
    for byte in data:
        if state == 1:
            if byte != 0:
                domainstring += chr(byte)
            x += 1
            if x == expectedlength:
                domainparts.append(domainstring)
                domainstring = ''
                state = 0
                x = 0
            if byte == 0:
                domainparts.append(domainstring)
                break
        else:
            state = 1
            expectedlength = byte
        y += 1
    questiontype = data[y:y+2]

    domain_name = ""
    for i in range(0,len(domainparts) - 1):
        if i == len(domainparts) -2:
            domain_name += domainparts[i]
        else :
            domain_name += domainparts[i] +"."
    qtype = struct.unpack('!h',questiontype)[0]
    q_row_3 = data[y+2:y+4]
    qclass = struct.unpack('!h',q_row_3)[0]
    arr.append(domain_name)
    arr.append(qtype)
    arr.append(qclass)
    arr.append(y+4+12+2)
    return arr
```

```python
def get_answer(data):
    arr=[]
    flag_1 = struct.unpack('!h',data[0:2])[0]
    flag_2 = struct.unpack('!h',data[2:4])[0]
    flag_3 = struct.unpack('!I',data[4:8])[0]
    flag_4 = struct.unpack('!h',data[8:10])[0]

    arr.append(flag_1)
    arr.append(flag_2)
    arr.append(flag_3)
    arr.append(flag_4)
    data = data[10:14]
    ip =""
    for byte in data:
        pid = hex(byte)[2:]
        pid = int(pid,16)
        ip += str(pid) + "."
    ip = ip[:-1]

    arr.append(ip)
    return arr



def response_parser(response_mesage_raw: bytearray) -> str:

    ID = get_transaction_id(response_mesage_raw)

    flag = get_flag(response_mesage_raw)
    QR = flag["qr"]
    OPCODE = flag["opcode"]
    AA = flag["aa"]
    TC = flag["tc"]
    RD = flag["rd"]
    RA = flag["ra"]
    AD = flag["ad"]
    CD = flag["cd"]
    RCODE = flag["rcode"]

    count_arr = get_count(response_mesage_raw)
    QDCOUNT=count_arr[0]
    ANCOUNT=count_arr[1]
    NSCOUNT=count_arr[2]
    ARCOUNT=count_arr[3]

    q_arr = get_domain(response_mesage_raw)
    QNAME=q_arr[0]
    QTYPE=q_arr[1]
```

```
    QCLASS=q_arr[2]
    q_length = q_arr[3]


    ans_arr = get_answer(response_mesage_raw[q_length:])
    TYPE=ans_arr[0]
    CLASS=ans_arr[1]
    TTL=ans_arr[2]
    RDLENGTH=ans_arr[3]
    IP=ans_arr[4]

    result = (
    '''
    [Response from DNS Server]\n
    ------------------------------------------------------------------------
\n
    HEADERS\n
    Request ID: {}\n
    QR: {} | OPCODE: {} | AA: {} | TC: {} | RD: {} | RA: {} | AD: {} | CD: {}
| RCODE: {}\n
    Question Count: {} | Answer Count: {} | Authority Count: {} | Additional
Count: {}\n
    ------------------------------------------------------------------------
\n
    QUESTION\n
    Domain Name: {} | QTYPE: {} | QCLASS: {}\n
    ------------------------------------------------------------------------
\n
    ANSWER
    TYPE: {} | CLASS: {} | TTL: {} | RDLENGTH: {}
    IP Address: {}
    ========================================================================
    ''').format(ID,QR,OPCODE,AA,TC,RD,RA,AD,CD,RCODE,QDCOUNT,ANCOUNT,NSCOUNT,A
RCOUNT,QNAME,QTYPE,QCLASS,TYPE,CLASS,TTL,RDLENGTH,IP)

    return result

def main():
    response =
b'\x83\x95\x85\x80\x00\x01\x00\x01\x00\x00\x00\x01\x03ctf\x0214\x08compfest\x0
0\x00\x01\x00\x01\xc0\x0c\x00\x01\x00\x01\x00\t:\x80\x00\x04\x98vF\x14\x00\x00
)\x10\x00\x00\x00\x00\x00\x00\x1c\x00\n\x00\x183K\x05z&\x88\x1e\xf4\x01\x00\x0
0\x00c6\x98\xba\xa5\x16\xc1TAO%!'
    print(response_parser(response))


if __name__ == "__main__":
    main()
```

Hasil yang diperoleh :

```
[Response from DNS Server]

    ------------------------------------------------------------------------

    HEADERS

    Request ID: 33685

    QR: 1 | OPCODE: 0 | AA: 1 | TC: 0 | RD: 1 | RA: 1 | AD: 0 | CD: 0 | RCODE:
0

    Question Count: 1 | Answer Count: 1 | Authority Count: 0 | Additional
Count: 1

    ------------------------------------------------------------------------

    QUESTION

    Domain Name: ctf.14.compfest | QTYPE: 1 | QCLASS: 1

    ------------------------------------------------------------------------

    ANSWER
    TYPE: 1 | CLASS: 1 | TTL: 604800 | RDLENGTH: 4
    IP Address: 152.118.70.20
    ========================================================================
```

Maka flag :

COMPFEST14{152.118.70.20_33685}